NBA Oracle

Matthew Beckler, Hongfei Wang
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
{mbeckler, hongfei}@cmu.edu

Michael Papamichael
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
mpapamic@cs.cmu.edu

Abstract

NBA Oracle uses Machine Learning methods to predict game outcomes and provide guidance for decision making in professional basketball. For game outcome prediction we were able to achieve up to 73% accuracy, using four standard binary classification algorithms: i) *Linear Regression*, ii) *Support Vector Machines*, iii) *Logistic Regression*, and iv) *Artificial Neural Networks*. For decision support we used i) *K-means Clustering* to predict optimal player positions with up to 75% accuracy and ii) *Outlier Detection* to identify outstanding players.

1 Introduction

The business of professional sports is a multi-billion dollar industry, with the NBA being one of its core constituents with over $3.5 billion dollars of revenue in the 2007-2008 season [3]. With many teams desperate for a winning season, the importance of well-informed coaching and player acquisition decisions [1] is critical. Moreover, professional sports betting, which is another multi-billion dollar industry on its own, greatly depends on accurate game outcome prediction [2];

In NBA Oracle we apply ML (Machine Learning) methods for i) predicting game outcomes and ii) providing guidance and advice for common decisions in the field of professional basketball. Specifically for game outcome prediction we experiment with four different ML binary classification techniques: i) *Linear Regression*, ii) *SVM (Support Vector Machines)*, iii) *Logistic Regression*, and iv) *ANN (Artificial Neural Networks)*. For decision support we apply i) *K-means Clustering* to infer optimal player positions and ii) *Outlier Detection* to identify outstanding players.

2 Problem Definition

NBA Oracle uses ML techniques for predicting the outcome of a game, choosing optimal player positions, and identifying outstanding players.

Game Outcome Prediction. Betting decisions can greatly benefit from accurate game outcome prediction, which involves predicting the winning team in a basketball game using data from previous seasons. In particular, the input to the four binary classification algorithms we used are team statistics, and the output is a binary value indicating the winning team. In an effort to improve accuracy we also tried augmenting our original data set (that only included cumulative data for each NBA season) with up-to-date team statistics that more accurately capture the current performance trends of each team. To attack this problem we used 4 different ML binary classification methods.

Inferring Optimal Player Positions. A basketball coach has a limited number of players and needs to maximize the performance of each player while on the court. Basketball has three main positions, that of “Center”, “Forward”, and “Guard”. If a player’s style and performance numbers are more characteristic of a different position, a coach might decide to let that player switch positions, to see

if they perform better. In this problem our task was to isolate the two most dominant features that best identify the position of a player and use those two features to predict the optimal position for a player, which was done through K-means clustering.

Identifying Outstanding Players. New player acquisitions and award nominations require selecting outstanding players by analyzing a large variety of player statistics. For this problem we first needed to find suitable performance metrics that successfully incorporate multiple features for each player; This is a tedious and subjective process that requires years of basketball experience, which made us look into existing NBA expert-derived metrics. For this problem we used two different player performance metrics, Approximate Value (AV) and Effectiveness, both developed by NBA expert analysts.Using these metrics, we experimented with outlier detection techniques to identify outstanding players.

3 Methodology

Our work for NBA Oracle consists of three major tasks. The first task was studying our original dataset and developing scripts for augmenting the dataset with additional statistics scraped from online sport databases. As part of our dataset preparation we spent a significant amount of time conditioning our dataset through cross-validation checks that helped us clean-up our stats, fixing erroneous or incomplete entries. To take advantage of the syntactic power of SQL, we inserted all of our data into a database, allowing us to be able to execute complicated queries. Once we finished with processing our data, the second task was performing game outcome prediction using 4 different ML techniques. For our experiments, we used two different versions of our dataset; one with our original downloaded dataset and one with the augmented dataset we generated through online database scraping and a series of SQL queries. Finally, the third task was to apply ML algorithms for decision support problems. We ran two sets of experiments; one for choosing optimal player positions through K-means Clustering and one for identifying outstanding players through outlier detection.

3.1 Data Set

The dataset used for this project was originally downloaded from the DatabaseBasketball[1] website. These raw data files contained cumulative stats of NBA players and teams, season-by-season. In order to do predictions on individual games, we needed to retrieve the game data from the website. This game data includes detailed stats for each individual player for each individual game. Due to the availability of consistent and reliable data, we focused primarily on seasons 1991-1992 through 1996-1997.

After we had obtained the raw data set, we needed to clean and standardize the raw data. We checked the completeness and validity of all the data files, and eliminated any CSV parsing errors or erroneous data values. Using the individual player-game data we were able to cross-validate the overall game scores for each team, to ensure that all player activity had been accounted for.

Having validated all our available data, we then proceeded to load the data from the plain text files into an SQL database using the SQLite single-file database engine and a few Python scripts. The flexibility of SQL queries allowed us to easily perform complex joins and merges between multiple tables. The database was accessible from both standard shell scripts and from within our Python scripts.

Now that we had complete, individual, and validated data stored in a powerful SQL database, we were able to distill all this data into the most useful form for our research. Individual player statistics were accumulated for each team over all games to produce cumulative statistics for each team, at each time-step. This cumulative team data provides us with the most current and accurate statistics for predicting the next game. For example, if we want to predict the outcome of a game that takes place at time-step k in the 1995-1996 season, we are now able to use not only the 1994-1995 season statistics, but also the current season's data up through time-step $k-1$.

Altogether, there are 14 basic individual player statistics, such as points scored, number of rebounds, blocks, and steals, as well as detailed shot statistics including number of field goals, free throws, and

[1] http://www.databasebasketball.com

three pointers attempted and made. Each team's statistics are accumulated player statistics, and include both offensive and defensive statistics, bringing the total to 28 features. A team's offensive stats would be the stats "earned" in favor if that team, while a team's defensive stats would include stats "earned" by their opponents. Adding in the wins and losses, we reach a total of 30 features per season. Usually we have both previous season and current season up-to-date data, giving us a grand total of 60 features to work with. Using the base statistics, we can compute a number of derived statistics that make comparison easier, such as normalizing the player stats to be "per game" or "per minute played". When comparing two teams, we found it useful to normalize the team statistics by taking the ratio of each team's numbers. This makes for an easier comparison when looking for the relative "advantage" that one team has over another.

3.2 Game Outcome Prediction

Linear Regression. Linear regression is usually one of the first classifiers introduced in machine learning. It is a form of regression analysis where a least-squares function is used to model the relationship between one or more independent variables (input features) and another variable, the dependent (output) variable. This function is a linear combination of the input features, weighted using linear values that are known as *regression coefficients*. Note that "linear" does not necessarily imply that the classification boundary will be a hyperplane, but rather that the degree of the regression coefficients is linear.

For our specific experiment here, we have many features which can be used for predicting the outcome of a game. For each of our input features, we associated it with a weight w_i for use in our regression equation. To simplify the computation, we have added an additional non-zero constant dimension to the data which allows us to automatically account for the w_0 offset. All together, we have the following linear regression model:

$$Y = w_0 + \sum_{i=1}^{n} w_i * x_i \tag{1}$$

The above formula gives us a very straightforward intuition for interpreting the results; the features that are assigned the largest weights will naturally be the most important in predicting the outcome of a game. Note that this includes both large positive and large negative numbers, as the sign of a weight only determines the correlation (either positive or negative) of the feature with the game outcome. The magnitude of each weight indicates the relative importance of that feature in determining the overall predicted outcome.

As the results for Y is a continuous number, we set up a threshold at 0.5, meaning

$$F = \begin{cases} 1 & \text{(win)}, \quad \text{if } Y >= 0.5 \\ 0 & \text{(lose)}, \quad \text{if } Y < 0.5 \end{cases}$$

Logistic Regression. Logistic regression is a discriminative classifier widely used in statistics and machine learning. It shares some similarities with linear regression, but uses a sigmoid function instead of a linear one. For the purposes of NBA Oracle we implemented a Matlab-based logistic regression classifier that uses the exponential logistic function:

$$Y = \frac{1}{1 + e^{-w^T x}}, \qquad \text{where } w \text{ corresponds to the feature weights vector}$$

In order to estimate the optimal weight vector w for our input features we used the Maximum Likelihood Estimation (MLE) method. In the case of logistic regression, MLE provides no closed form solution for calculating the weight values; however, since we are dealing with a concave function, we can use a gradient descent approach. In our logistic regression model the weight vector values converged in most cases after approximately 200-300 iterations using an epsilon threshold value of $\epsilon = 0.01$. As is the case with linear regression the magnitude of the calculated weights is a first-order indicator of the importance of each individual feature.

Support Vector Machines. While a linear or logistic classifier considers all points equally while minimizing the mean-squared error, the resultant classifier might perform poorly in the absence of

sufficient training data. One possible solution to increase the test accuracy is to instead focus on maximizing the margin of classification. Support Vector Machines (SVM) are one way to build a classifier that maximizes the margin, using the training points closest to the classification boundary as "support vectors". This allows SVM to focus on the points that really matter, namely the points that define the decision boundary, and ignore the "easy" points that lie far from the boundary.

Another feature of SVM methods is the use of a kernel to map the input feature space to a higher dimension, which can often be used to make the data points linearly-separable. Some of the most commonly used kernels include a basic linear kernel, a polynomial kernel, and the radial-basis function kernel, which is based on the Gaussian distribution. In the course of our experimentation, we tried many different kernel functions for SVM, but found that for our datasets, the linear kernel was the most successful. We used an open-source SVM implementation called SVM^{lite}, created by Thorsten Joachims at Cornell University [8].

Artificial Neural Networks. An *artificial neural network* (ANN) is often used as a classifier based on the idea of interconnected biological neurons, as in a central nervous system. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. It many cases, an ANN is an adaptive system that changes its structure based on external and internal information flows during the network training phase. Neural networks can be very non-linear statistical data modeling tools, able to model complex relationships between inputs and outputs, or to find interesting patterns in data records.

We had originally hoped that by using neural networks, we could dig out some of the hidden internal structure of the input data to learn the target function. We used some of the tools provided in Matlab's Neural Network Toolbox library. The network was implemented as a feed-forward back-propagation network. The first layer has weights coming from the input. Each subsequent layer has a weight for each value coming from the previous layer. All layers have biases. The last layer is the network output. Adaption is done with training, which updates weights with the specified analyzing and learning function from the Matlab library.

Specifically, we implemented a neural network with three layers– the input layer, the hidden layer and the output layer. The input layer has 64 nodes, one for each input feature in our data. The hidden layer has 20 hidden nodes each implementing a sigmoid transfer function. The output layer just has one node, using linear regression to produce a single output value. Like standard linear regression, the output here is also a continuous number which requires a threshold to produce a binary output. For this experiment, the threshold was set to be 0.5.

The main structure of the neural network is a feed-forward back-propagation network. Configuration of the network involved specifying the desired number of layers as well as the number of nodes per layer. We explored the configuration space in a non-exhaustive way, and tried to use our prior knowledge of both neural networks and the problem domain to guide our exploration. We eventually found that a neural network with a single hidden layer with 20 nodes worked the best for the data we were using. Given that it was not an exhaustive search, we cannot be certain that this is the best possible configuration. We did observe that as we added more hidden layers and more nodes per layer, that the time required for training increased dramatically, seeming exponential rather than linear. Despite the extra flexibility of the multiple hidden layers, we found that over-fitting was a significant problem, with the more complex networks performing no better than the single hidden layer configurations.

3.3 Decision Support

Player Position Inference. At the start of this experiment, we were unsure as to which characteristic or statistic of a player's performance would most strongly correlate with their position on the court. We chose to use K-means clustering for this task, because it is a method of clustering that does not require any supervision or correct classifications in order to find natural clusterings in the input data. Since there are three possible positions for each player, we naturally used K-means with $k = 3$. We expected that there would be a strong correlation between the player's position and at least one of the standard individual statistics, but we were unsure which one it would be. We also expected that the proper application of K-means clustering to our dataset would be able to identify this correlation.

The K-means algorithm partitions all of our data points into these three clusters, based on the standard Euclidean distance between the point and the center of each cluster. It is a two-step algorithm, and seeks to reduce the overall within-cluster sum of squares error. We want to find the cluster centers s that minimize:

$$\arg_s \min \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2$$

The K-means algorithm does not guarantee to find the global minimum error, but will certainly find a local minimum. Common solutions to this problem include random restarts and incorporating ideas from simulated annealing, such as allowing "uphill" state changes which temporarily increase the overall cost before finding a better solution. We did not investigate any such enhancements, and used the standard K-means *Assignment-Update* algorithm.

Outstanding Player Detection. For outstanding player detection we first generated two new features for each player that we used as our performance metric. Both features were developed by NBA expert analysts and are essentially linear combinations of a subset of the existing career statistics for each player. The first metric, called Approximate Value (AV), was developed by former NBA basketball player and coach Dean Oliver [9] and tries to capture long-term and consistent performance of a player for an entire season. The second metric called Efficiency captures shorter term performance characteristics of a player and can tolerate high variance in a player's performance; this metric will rank well players that might have performed exceptionally well for a few games but are not very consistent over all. Using these two features we created a scatter plot for all players from 1946 to 2001 after first eliminating players that had played less than 100 minutes in total and inconsistent entries. From this scatter plot we could easily identify the outliers by visual inspection.

4 Results

4.1 Game Outcome Prediction

For all of the game outcome prediction experiments, we did two main experiments. First, we used only the previous season's data to predict the current season's game outcomes. Once we had up-to-current data available, we re-ran all of our experiments using both the previous season and current season up-to-current date data records. For all experiments we used 100-fold cross validation.

Overall Results. For the results in the table below, we performed classification on both the "previous season only" (P) and "previous plus current up-to-date" (P+C) datasets, for seasons 1992-1993 through 1996-1997, for all four classifiers. Numbers provided are test set classification accuracy. Please see Figure 1 for a graphical representation of this data.

GAME OUTCOME PREDICTION ACCURACY RESULTS								
	Linear		Logistic		SVM		ANN	
	P	P+C	P	P+C	P	P+C	P	P+C
1992	0.6945	0.6955	0.6736	0.6800	0.6445	0.6527	0.6473	0.6309
1993	0.7110	0.7070	0.6910	0.7000	0.6680	0.6980	0.6601	0.6620
1994	0.6709	0.6836	0.6527	0.6736	0.6527	0.6655	0.6236	0.6400
1995	0.6882	0.6982	0.6773	0.6891	0.6500	0.6864	0.6415	0.6754
1996	0.7309	0.7200	0.6773	0.6955	0.6827	0.6927	0.6664	0.6595
Mean	0.6991	0.7009	0.6744	0.6876	0.6596	0.6791	0.6478	0.6536

Overall, we see that linear regression out-performed all other classifiers for all seasons under consideration. Logistic regression finished second, with SVM a close third. Neural networks finished last, possibly due to our exploratory tuning of the neural network, having never achieved classification accuracies as high as the other methods. When we look at the mean accuracies for each classifier, specifically trying to assess how useful the "up-to-date" current season data can be, we find an average increase in classification accuracy of approximately 1%. Another interesting feature of the data is that some seasons appear to be inherently more difficult to predict, such as the 1992-1993 and 1994-1995 seasons, which showed lower overall classification accuracies for all four classifiers.

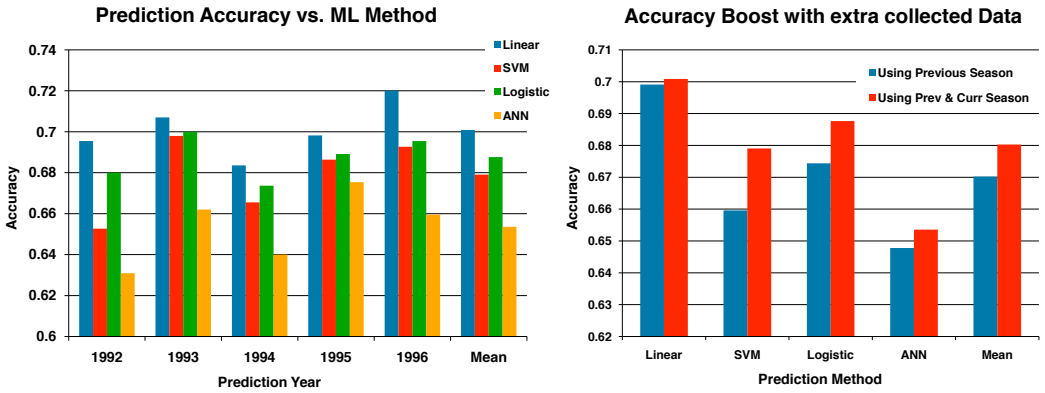


Figure 1: Prediction Accuracy and Accuracy Boost

Evaluation. To evaluate the game outcome prediction accuracy results of NBA Oracle we compared our accuracy with some other source of game outcome predictions, which are shown in the table below along with their associated accuracies:

Source:	Random	Majority Vote	Websites	Prior Work	Experts	NBA Oracle
Accuracy:	50%	62%	65%	70%	71%*	Up to 73%

Since game outcome prediction is a binary classification problem even randomly guessing can already achieve an average accuracy of 50%; we definitely want to perform better than this. To get a first-order feeling on accuracy results we experimented with a very naive majority vote classifier that always looked at all previous games between two teams and picked the one with the fewest losses as the winner. This algorithm achieved a modest accuracy of up to 62%. We then looked at the prediction accuracy guarantees listed on websites providing betting advice, which usually are in the range of 65% and we set this as our initial accuracy goal. Subsequently we looked at other similar projects that also try to predict game outcome prediction using similar ML methods and found that the best results reported an accuracy of 70%. Finally we looked at the prediction accuracy of well-established NBA experts, which is in the range of 71%. Surpassing the accuracy of NBA experts is quite hard for two reasons: i) NBA experts have more data available when they make their predictions (e.g. player injuries, importance of game, etc) and ii) NBA experts can refuse to predict hard games, which can significantly boost their accuracy. Nevertheless NBA Oracle's average accuracy was very close to that of NBA experts and in some cases even surpassed it, achieving an accuracy up to 73%. Overall game outcome prediction is a hard problem, governed by many sources of randomness, such as player injuries, player attitudes, team rivalries, subjective officiating and other non-deterministic factors.

Mutual Information. Although we used all available features for predicting game outcomes we were interested to identify the most dominant features that affect the outcome of a game. To achieve this we developed Matlab code that computes the mutual information of each feature with respect to the outcome of a game and tried it out on all 30 features of a team for all seasons from 1991 to 1997. The two most dominant features were the number of wins and losses that a team had in previous seasons. This was expected, since there is a strong correlation between the number of games that a team has won in previous years and the number of games a team will win in the future. The surprising result was the four next most dominant features, which were all defensive statistics. In particular in order of decreasing importance these were: defensive rebounds, points made by opposing team, number of blocks and assists made by opposing team. These results indicate that in order to win a game it is more important to prevent the opposing team from playing well and scoring points instead of solely focusing on playing well offensively. This result also validates the quote "Offense sells tickets; Defense wins championships." attributed to legendary coach Bear Bryant. Figure X shows the increase in accuracy for logistic regression as each feature is incrementally added one-by-one in increasing mutual information order.

4.2 Decision Support

K-Means clustering. When classifying into three classes, our base accuracy from random classification would be 33%. Running k-means (for $k = 3$) on the original 14 individual player statistics yielded classification accuracy of approximately 65%, nearly twice as accurate. We then looked into using fewer input features for the clustering in order to improve prediction accuracy. An exhaustive search was performed for all possible pairings of the 14 base statistics, trying to find the two most important features for predicting a player's position on the court. The two features were found to be the number of rebounds and steals, both normalized to each player's playing time. This result makes good intuitive sense, because players in the "center" position normally play close to the basket, a good position for making rebounds, but not a good place for getting steals. Those players who are "guards", however, usually play further away from the basket, where rebounds are more rare, but steals are much more likely. We can see this natural clustering in Figure 2.

The k-means code was implemented in Matlab, and does not take into account the true player classification during the training phase. We used the standard Euclidean distance as our distance function, and iterated the clustering algorithm until the total cost function stopped decreasing.

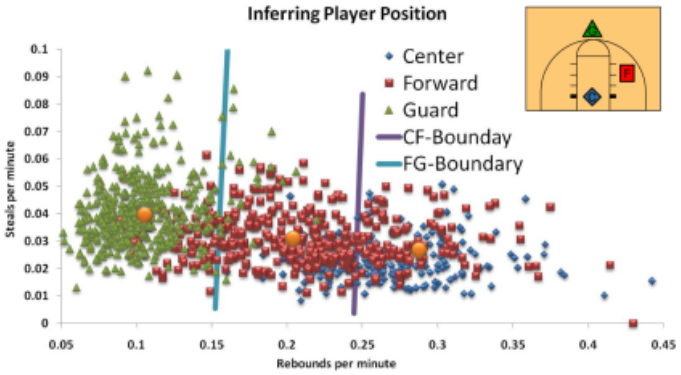


Figure 2: K-Means Clustering of Data

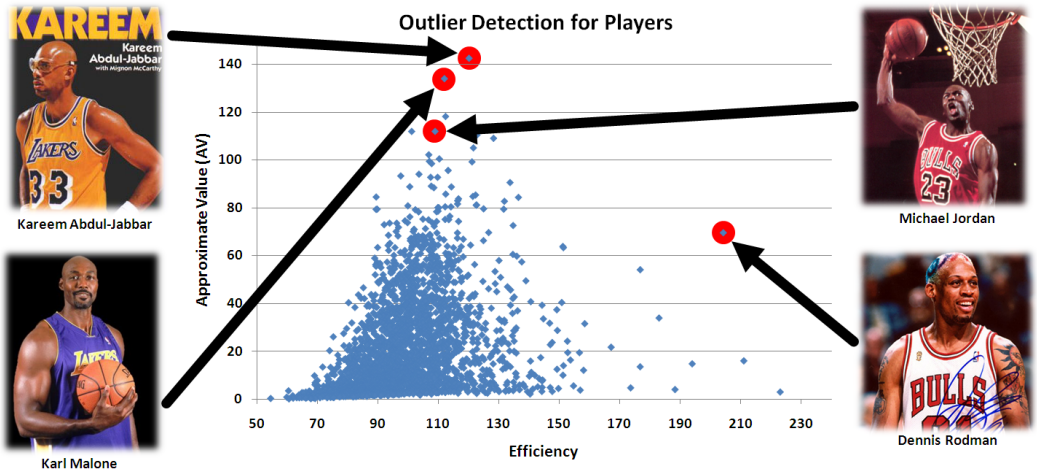
Feature Addition Accuracy Boost	
Added Feature	Accuracy
p_won	0.663
p_lost	0.670
pd_dreb	0.673
pd_pts	0.682
pd_blk	0.685
pd_asts	0.690

Table 1: Feature Addition Boost

Outlier Detection. The scatter plot used for outlier detection is presented in Figure 3 along with a few examples of some outstanding players we detected through visual inspection. Players that rank high on the vertical axis, that corresponds to the Approximate Value metric, are all-time famous exceptional basketball players, such as Kareem Abdul-Jabbar, Michael Jordan and Karl Malone. Players that ranked high on the horizontal axis have high Efficiency values and correspond to players that might have performed well in a few games, but overall were inconsistent and as a consequence not very well known. It is interesting to see that some players did not "fit" very well to the two chosen NBA expert-derived metrics, because their stats were drastically different than most other players. For instance Dennis Rodman consistently had an extraordinary amount of rebounds, but was very inconsistent in the amount of points he scored.

5 Conclusions

We applied a series of ML methods to predict game outcomes, choose optimal player positions and identify outstanding players to support betting, coaching, sponsorship and other decision in the field of professional basketball. For game outcome prediction our best accuracy results (up to 73%) were achieved using linear regression methods, which indicates that simple approaches seem to perform well for the specific problem domain. For player position inference our K-means algorithm was able to correctly classify players according to position 75% of the time. Finally for outstanding player detection our outlier detection methods, using the two chosen NBA expert-derived metrics, were able to pinpoint the majority of all-star players in NBA history.



References

[1] Colet, E. and Parker, J. *Advanced Scout: Data mining and knowledge discovery in NBA data.* Data Mining and Knowledge Discovery, Vol. 1, Num. 1, 1997, pp 121 – 125.

[2] McMurray, S. (1995). *Basketball's new high-tech guru.* U.S. News and World Report, December 11, 1995, pp 79 – 80.

[3] Howard, H. *The Explosion of the Business of Sports in the United States.* Nike Seminar, Spring 1998. http://www.unc.edu/ andrewsr/ints092/howards.html

[4] Babak, Hamadani. *Predicting the outcome of NFL games using machine learning.* Project Report for CS229, Stanford University. http://www.stanford.edu/class/cs229/proj2006/BabakHamadani-PredictingNFLGames.pdf

[5] Balla, Radha-Krishna. *Soccer Match Result Prediction using Neural Networks.* Project report for CS534. http://rk-pvt-projects.googlecode.com/svn/trunk/CS534_ML_SoccerResultPredictor/docs/CS534_ProjectReport.pdf

[6] Orendorff, David, and Johnson, Todd. *First-Order Probabilistic Models for Predicting the Winners of Professional Basketball Games.* Project report. http://www.ics.uci.edu/ dorendor/basket.pdf

[7] AccuScore, *The Leader in Sports Forecasting*, http://tinyurl.com/dhmsts

[8] T. Joachims, *Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning*, B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[9] Oliver, Dean. "Approximate Value Methods", *Journal of Basketball Studies - Methods Descriptions for the Scientific Study of Basketball*, http://www.powerbasketball.com/theywin2.html

[10] The MathWorks, *Neural Network Toolbox 6.0.2*, http://www.mathworks.com/products/neuralnet/