Computational Bracketology

Harrison Hall, John Sigman
Ph.D. Candidates in Engineering
Dartmouth College
{harrison.k.hall.th, john.b.sigman.th}@dartmouth.edu

March 8, 2013

1 Background

The NCAA Men's Basketball Tournament is a 64-team, single elimination tournament held every year that determine's the nation's national champion. Even with 6.45 million brackets, as in Figure 1, filled out on ESPN.com last year[1] the winner still failed to predict 12 of the games correctly[2]. While it is astronomically unlikely that anyone has or will ever picked a perfect bracket, a chance of 1 in 2^{63}, it is clear that the current augmented human predictions are not perfect. While some machine learning algorithms exists which are competitive with the brackets of professional sports analysts, these algorithms are designed to take into account only team-level statistics. While single-game, individual player statistics are available[3] current, published approaches tend to not evaluate the the importance of individual players or potential player match-ups.

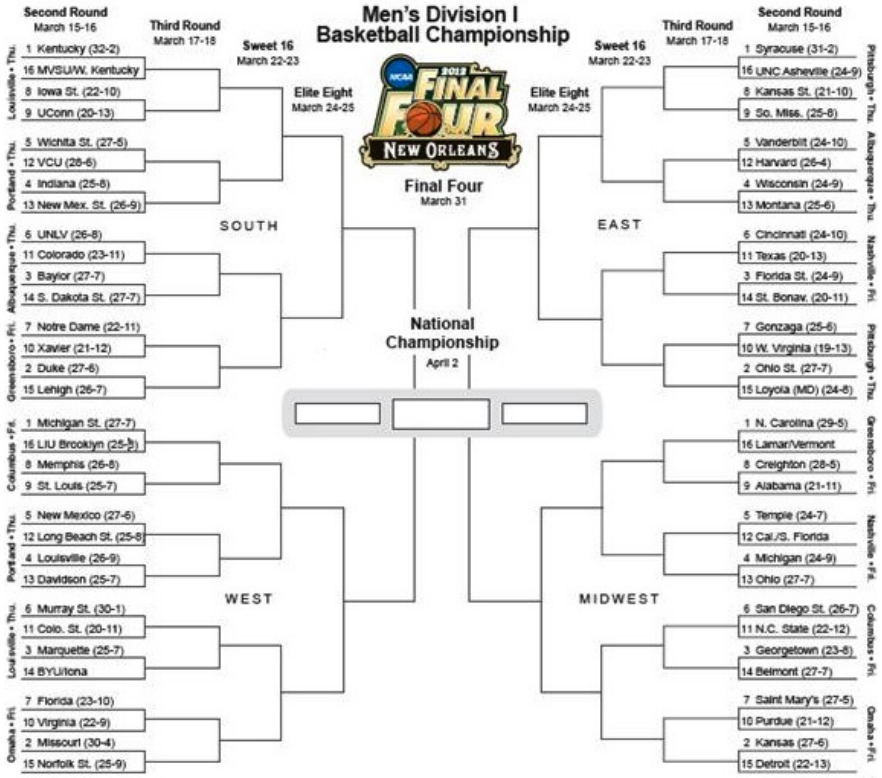


Figure 1: Blank 2012 NCAA Division I Men's Basketball Tournament bracket. Image courtesy of AP.

2 Dataset

Our dataset was scraped from the ESPN NCAA Men's Basketball website[3]. The set of information provided by ESPN can be seen in Table 1. They provide schedule data for all of the Division I teams that field men's basketball teams and their full schedules going back to the 2001-2002 season. For each game in those years ESPN provides game level box scores. In total we were able to scrape data for the 347 teams, excluding non-D1 opponents, corresponding to $21,366$ games.

ESPN also provides player statistics for every game as well as biographic data about each player. From the set of games described above we were able to extract $429,876$ player statistics with the fields described in Table 1b from the four seasons between 2008-2009 and 2011-2012. We were also able to pull biographic data as shown in Table 1e. However that data is not on a timeline so it is not possible to track variance in player weight, height, position, etc. across their college basketball careers.

Table 1: Fields provided by ESPN and their respective datatypes.

(a) Game

Field	Type
id	Integer
Game Time	Datetime
Site Arena	String
Site City	String
Site State	String
Site Arena	String
Home Team	FK(School)
Away Team	FK(School)
Home Team First Half Score	Integer
Away Team First Half Score	Integer
Home Team Second Half Score	Integer
Away Team Second Half Score	Integer
Home Team Final Score	Integer
Away Team Final Score	Integer
Number of Overtimes	Integer
Home Team Overtime Score	Integer
Away Team Overtime Score	Integer
Regular Season	Boolean
NCAA Tournament	Boolean

(b) Player Statline

Field	Type
Game	FK(Game)
Player	FK(Player)
School	FK(School)
Starter	Boolean
Points	Integer
Minutes	Integer
Field Goals	Integer
Field Goal Attempts	Integer
3-Point Goals	Integer
3-Point Goal Attempts	Integer
Free Throws	Integer
Free Throw Attempts	Integer
Offensive Rebounds	Integer
Defensive Rebounds	Integer
Assists	Integer
Steals	Integer
Blocks	Integer
Turn Overs	Integer
Personal Fouls	Integer

(c) School

Field	Type
id	Integer
Name	String
Mascot	String
Conference	FK(Conference)

(d) Conference

Field	Type
id	Integer
Name	String

(e) Player Biography

Field	Type
id	Integer
Name	String
Position	String
Birthday	Date
Hometown	String
Home State	String
Height (Feet)	Integer
Height (Inches)	Integer
Weight (Pounds)	Integer

This data is organized for ease of scraping from the source. Our model for predicting games in the NCAA tournament is single game comparisons based on historical data where each game is treated as an independent event. Thus we format this collected data such that each row contains the data used to characterize the each team's historical performance up to the date of the game. Table 2 shows the features that were generated from the scraped dataset. Note that Tables 2b and 2c were populated for each team.

Table 2: Fields formatted for classification in our algorithms

(a) Game Information

date	Date
Game Id	Integer
Home Team Id	Integer
Away Team Id	integer
Home Team Is Winner	Boolean
Game at Neutral Site	Boolean
Game in Postseason	Boolean
Game in NCAA	Boolean

(b) Historical Team Outcomes

Conference	Integer
RPI	Float
Points Scored	Integer
Poinst Against	Integer
Points +/-	Integer
Home Wins	Integer
Away Wins	Integer
Neutral Site Wins	Integer
Home Losses	Integer
Away Losses	Integer
Neutral Site Losses	Integer
Overtime Wins	Integer
Overtime Losses	Integer
Non-NCAA Postseason Wins	Integer
Non-NCAA Postseason Losses	Integer

(c) Cumulative Team Statistics

Field Goals Made	Integer	Number of Players 5'6"-6'	Integer
Field Goals Attempted	Integer	Number of Players 6'-6'6"	Integer
Threes Made	Integer	Number of Players 6'6"-7'	Integer
Threes Attempted	Integer	Number of Players 7'-7'6"	Integer
Free Throws Made	Integer	Number of Players 7'6"-8'	Integer
Free Throws Attempted	Integer	Number of Players Over 8'	Integer
Offensive Rebounds	Integer	Average Player Weight	Float
Defensive Rebound	Integer	Number of Players under 150 lbs	Integer
Assists	Integer	Number of Players 150-175 lbs	Integer
Steals	Integer	Number of Players 175-200 lbs	Integer
Blocks	Integer	Number of Players 200-225 lbs	Integer
Turnovers	Integer	Number of Players 225-250 lbs	Integer
Fouls	Integer	Number of Players 250-275 lbs	Integer
Average Player Points	Integer	Number of Players 275-300 lbs	Integer
Number of Starters	Integer	Number of Players Over 300 lbs	Integer
Average Age of Player	Float	Number of Centers	Integer
Number of Players Under 17	Integer	Number of Guards	Integer
Number of Players Aged 17	Integer	Number of Forwards	Integer
Number of Players Aged 18	Integer	Number of Utility Players	Integer
Number of Players Aged 19	Integer	Average Height of Centers	Float
Number of Players Aged 20	Integer	Average Weight of Centers	Float
Number of Players Aged 21	Integer	Average Height of Guards	Float
Number of Players Aged 22	Integer	Average Weight of Guards	Float
Number of Players Aged 23	Integer	Average Height of Forwards	Float
Number of Players Over 24	Integer	Average Weight of Forwards	Float
Average Height(In)	FLoat	Average Height of Utility Players	Float
Number of Players under 5'	Integer	Average Weight of Utility Players	Float
Number of Players 5'-5'6"	Integer		

3 Approach

3.1 RPI

The Ratings Percentage Index (RPI)[6] is an industry-standard statistic that comes from the following relations:

$$t_i, t_j \in \mathbb{T} = \{\text{Team}_1 \dots \text{Team}_m\} \tag{1}$$
$$n = \text{number of days in the season} \tag{2}$$
$$k \in [1 \dots n] \tag{3}$$
$$\mathbb{G} = \text{The set of all games played in a season} \tag{4}$$
$$\mathbb{O}_{i,k} = \text{The set of all teams } t_i \text{ played in the first } k-1 \text{ days of the season} \tag{5}$$
$$g_{i,j,k} \in \mathbb{G} \; s.t. \text{ the game played between } t_i \text{ and } t_j \text{ is on day } k \tag{6}$$
$$H^w_{i,j,k} = \text{Indicator function if } g_{i,j,k} \text{ exists and } i \text{ won at home} \tag{7}$$
$$H^l_{i,j,k} = \text{Indicator function if } g_{i,j,k} \text{ exists and } i \text{ won at home} \tag{8}$$
$$A^w_{i,j,k} = \text{Indicator function if } g_{i,j,k} \text{ exists and } i \text{ won away} \tag{9}$$
$$A^l_{i,j,k} = \text{Indicator function if } g_{i,j,k} \text{ exists and } i \text{ won away} \tag{10}$$
$$N^w_{i,j,k} = \text{Indicator function if } g_{i,j,k} \text{ exists and } i \text{ won at a neutral site} \tag{11}$$
$$N^l_{i,j,k} = \text{Indicator function if } g_{i,j,k} \text{ exists and } i \text{ won at a neutral site} \tag{12}$$
$$a = 0.25 \tag{13}$$
$$b = 0.5 \tag{14}$$
$$c = 0.25 \tag{15}$$
$$d, e, f \in \mathbb{R}[0 \dots 2] \tag{16}$$

$$RPI_{i,k} = a \cdot WP_{i,k} + b \cdot OWP_{i,k} + c \cdot OOWP_{i,k} \tag{17}$$
$$WP_{i,k} = \frac{\sum_{x=1}^{k-1} \sum_{y=1}^{m} d \cdot H^w_{i,y,x} + e \cdot A^w_{i,y,x} + f \cdot N^w_{i,y,x}}{\sum_{x=1}^{k-1} \sum_{y=1}^{m} d \cdot H^w_{i,y,x} + e \cdot A^w_{i,y,x} + f \cdot N^w_{i,y,x} + (2-d) \cdot H^l_{i,y,x} + (2-e) \cdot A^l_{i,y,x} + (2-f) \cdot N^l_{i,y,x}} \tag{18}$$
$$OWP_{i,k} = \frac{\sum_{o \in \mathbb{O}_{i,k}} WP_{o,k}}{||\mathbb{O}_{i,k}||}[1] \tag{19}$$
$$OOWP_{i,k} = \frac{\sum_{o \in \mathbb{O}_{i,k}} \sum_{p \in \mathbb{O}_{o,k}} WP_{p,k}}{\sum_{o \in \mathbb{O}_{i,k}} ||\mathbb{O}_{o,k}||} \tag{20}$$

RPI is perceived as a good way to rank teams and correct for the strength of an individual teams schedule. For NCAA Basketball, the wins and losses are weighted so that a win at home counts as $d = 0.6$, and a win on the road counts for $e = 1.4$ wins. Away losses count as $2 - e = 0.6$ and home losses count as $2 - d = 1.4$ losses. Neutral site games are counted as away games for both opponents, thus $d = f$. These weightings are to compensate for Home-Court Advantage, which we will discuss in our results.

3.2 Decision Trees

Decision trees attempt to reduce the entropy in the set of training examples at each node by selecting the classifier available that reduces the entropy the most. This requires a binary classifier to determine

[1] Typically, $OWP_{i,k}$ is calculated by omitting the meetings of team i with all of its opponents, however in this definition it was omitted for succinctness.

classification. In the case of our project the features used for the branching came from a strictly greater than comparison of the characteristics of the home versus away team features as listed in Tables 2. This gave us a binary classification upon which to build the trees. We used the implementation that we built for Homework 2.

3.3 Random Forests

Random forests build off of the structure of decision trees except there are k trees used and at each node in the tree a random subset of the available binary features are used for branching. The class that has the majority vote by the k trees in the forest is selected. We used the implementation that we built for Homework 2.

3.4 SVM

Our first attempt to generate a large set of weak classifiers to be used in AdaBoost was using Support Vector Machines. Our data was so noisy, however, that regardless of the slack, we could not get linear separation of classes, and there were no clear nonlinearities that could have warranted a nonlinear transformation. We then abandoned SVM in favor of the simpler k-nearest neighbors algorithm.

3.5 AdaBoost

Adaboost (Adaptive Boosting)[4], a meta, supervised learning algorithm that takes a set of trained "weak" classifiers, $h_t(x)$ and selectively weights a subset of them to generate a strong classifier. This subset of T classifiers is constructed by iteratively selecting the most accurate classifier and reweighting the samples that are classified incorrectly by the classifier as more important, while decreasing the importance of those samples correctly classified, for the next iteration. The accuracy of classification is computed as a sum of the weights of the misclassified points as seen in Equation 21.

$$\epsilon_j = \sum_i = 1^m W_t(i)(y_i == h_j(x_i)) \tag{21}$$

At each iteration AdaBoost calculates the weighted error,ϵ_j, of each of the T classifiers and selects the one with the least error. It then calculates α_t that is uses to weigth the most accurate classifier of this round and reweights all of the points in the training set. On initialization all of the weights of the sample points are set to $\frac{1}{m}$.

$$\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t} \tag{22}$$

$$W_{t+1}(i) = \frac{W_t(i)\exp(-\alpha_t y_i h_t(x_i))}{\sum_{k=1}^{m} W_t(k)\exp(-\alpha_t y_k h_t(x_k))} \tag{23}$$

Upon running T iterations we have a set of T classifiers each with weight $\alpha_1 \ldots \alpha_T$. To classify a test example we run the T classifiers on the test example and multiply their output by the corresponding α_t and take the sign of that to be the output class as seen in Equation (24)

$$H(x) = \text{sign}\left(\sum_{i=1}^{T} \alpha_i h_i(x)\right) \tag{24}$$

3.5.1 kNN

Because of the size and complexity of our data, we opted to use K-Nearest Neighbors as the classifier input to AdaBoost. Since we have 71 features per team per sample and we wanted to compare the same feature dimensions across the team we have $\sum_{i=1}^{71} 71 choose i$ possible feature groupings. This set is clearly too large to enumerate so we took two apporaches to solve this problem: enumeration for classifiers at the ends of the spectrum and random selection.

In the enumeration approach we used i from the set $\{1, 2, 70, 71\}$. Using these features to train AdaBoost took on the order of a day which was too long for the quick iteration that we needed for this project so we instead randomly created several thousand classifiers, using different choices of k and i. We ended up randomly selecting 2000 kNN classifiers with ks ranging from 1 to 21 counting by 4, and between 3 and 5 features selected from each team.

4 Results

Results are grouped into two sections, the optimization of RPI and game prediction using various methods.

4.1 RPI Optimization

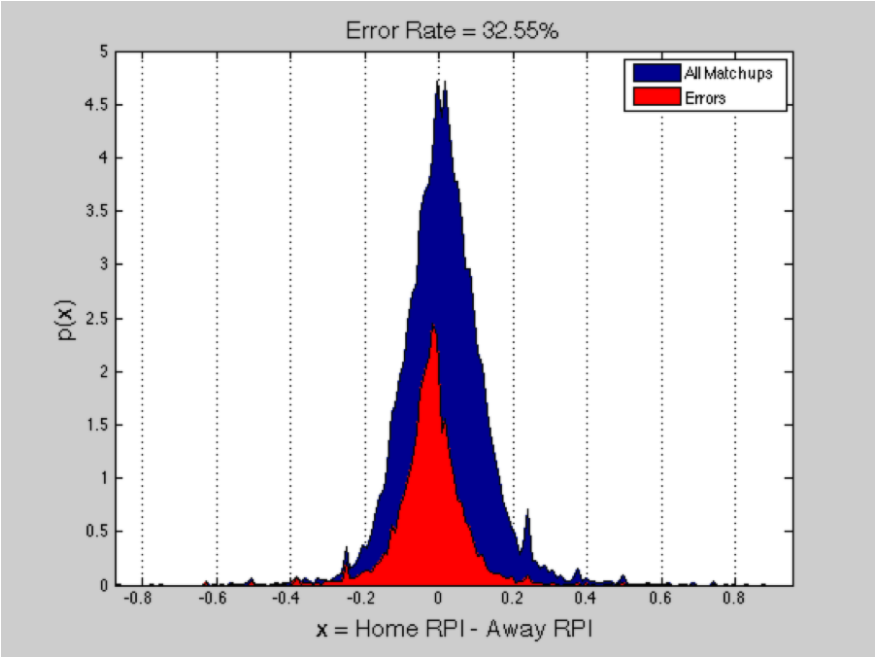


Figure 2: Unoptimized calculation of RPI using the NCAA standard of home wins, away losses, and neutral site losses being treated as 0.6 games while home losses, away wins, and neutral site wins are treated as 1.4 games.

Figure 2 is a probability density of all the matchups as a function of the difference in RPI score of the home and away teams before to the game. The x-axis is in order of increasing home team favor towards the right of the figure. The error rate can be derived from this figure by the ratio of the area under the red curve (total number of errors) to the area under the blue curve (total number of matchups). The overall error rate for the four seasons 2009-2012 was 31%.

We proposed to optimize the RPI with respect to predicted errors via batch gradient descent. The first step was to compute the partial derivatives for RPI. The equations below are the derivatives with respect to one team. Since the prediction is made by the difference in RPI score, the actual error function gradient is taken by the difference in these partial derivatives for the home and away teams in the matchup.

$$RPI(x) = (1 - B - C)WP(x) + B \cdot OWP(x) + C \cdot OOWP(x) \tag{25}$$

$$WP(x) = \frac{d \cdot H_w^x + e \cdot A_w^x + f \cdot N_w^x}{d \cdot H_w^x + e \cdot A_w^x + f \cdot N_w^x + (2-d) \cdot H_l^x + (2-e)A_l^x + (2-f)N_l^x} \tag{26}$$

$$W_{RPI}(x^i, x^j) = RPI(x^i) - RPI(x^j) \tag{27}$$

$$\begin{aligned} &(1 - B - C)\Bigg(\frac{d \cdot H_w^{x^i} + e \cdot A_w^{x^i} + f \cdot N_w^{x^i}}{2(H_l^{x^i} + A_l^{x^i} + N_l^{x^i}) + d(H_w^{x^i} - H_l^{x^i}) + e(A_w^{x^i} - A_l^{x^i}) + f(N_w^{x^i} - N_l^{x^i})} \\ = \quad &- \frac{d \cdot H_w^{x^j} + e \cdot A_w^{x^j} + f \cdot N_w^{x^j}}{2(H_l^{x^j} + A_l^{x^j} + N_l^{x^j}) + d(H_w^{x^j} - H_l^{x^j}) + e(A_w^{x^j} - A_l^{x^j}) + f(N_w^{x^j} - N_l^{x^j})}\Bigg) \\ &B\big(OWP(x^i) - OWP(x^j)\big) + B\big(OOWP(x^i) - OOWP(x^j)\big) \end{aligned} \tag{28}$$

$$\frac{dRPI}{dB} = OWP(x^i) - WP(x^j) \tag{29}$$

$$\frac{dRPI}{dC} = OOWP(x^i) - WP(x^j) \tag{30}$$

$$\begin{aligned} \frac{dRPI}{dd} = \quad &(1 - B - C)\Bigg(\frac{H_w^{x^i}\big(2(H_l^{x^i} + A_l^{x^i} + N_l^{x^i}) + d(H_w^{x^i} - H_l^{x^i}) + e(A_w^{x^i} - A_l^{x^i}) + f(N_w^{x^i} - N_l^{x^i})\big)}{\big(2(H_l^{x^i} + A_l^{x^i} + N_l^{x^i}) + d(H_w^{x^i} - H_l^{x^i}) + e(A_w^{x^i} - A_l^{x^i}) + f(N_w^{x^i} - N_l^{x^i})\big)^2} \\ &- \frac{\big(d \cdot H_w^{x^i} + e \cdot A_w^{x^i} + f \cdot N_w^{x^i}\big)(H_w^{x^i} - H_l^{x^i})}{\big(2(H_l^{x^i} + A_l^{x^i} + N_l^{x^i}) + d(H_w^{x^i} - H_l^{x^i}) + e(A_w^{x^i} - A_l^{x^i}) + f(N_w^{x^i} - N_l^{x^i})\big)^2} \\ &- \frac{H_w^{x^j}\big(2(H_l^{x^j} + A_l^{x^j} + N_l^{x^j}) + d(H_w^{x^j} - H_l^{x^j}) + e(A_w^{x^j} - A_l^{x^j}) + f(N_w^{x^j} - N_l^{x^j})\big)}{\big(2(H_l^{x^j} + A_l^{x^j} + N_l^{x^j}) + d(H_w^{x^j} - H_l^{x^j}) + e(A_w^{x^j} - A_l^{x^j}) + f(N_w^{x^j} - N_l^{x^j})\big)^2} \\ &- \frac{\big(d \cdot H_w^{x^j} + e \cdot A_w^{x^j} + f \cdot N_w^{x^j}\big)(H_w^{x^j} - H_l^{x^j})}{\big(2(H_l^{x^j} + A_l^{x^j} + N_l^{x^j}) + d(H_w^{x^j} - H_l^{x^j}) + e(A_w^{x^j} - A_l^{x^j}) + f(N_w^{x^j} - N_l^{x^j})\big)^2}\Bigg) \end{aligned} \tag{31}$$

$$\begin{aligned} \frac{dRPI}{de} = \quad &(1 - B - C)\Bigg(\frac{A_w^{x^i}\big(2(H_l^{x^i} + A_l^{x^i} + N_l^{x^i}) + d(H_w^{x^i} - H_l^{x^i}) + e(A_w^{x^i} - A_l^{x^i}) + f(N_w^{x^i} - N_l^{x^i})\big)}{\big(2(H_l^{x^i} + A_l^{x^i} + N_l^{x^i}) + d(H_w^{x^i} - H_l^{x^i}) + e(A_w^{x^i} - A_l^{x^i}) + f(N_w^{x^i} - N_l^{x^i})\big)^2} \\ &- \frac{\big(d \cdot H_w^{x^i} + e \cdot A_w^{x^i} + f \cdot N_w^{x^i}\big)(A_w^{x^i} - A_l^{x^i})}{\big(2(H_l^{x^i} + A_l^{x^i} + N_l^{x^i}) + d(H_w^{x^i} - H_l^{x^i}) + e(A_w^{x^i} - A_l^{x^i}) + f(N_w^{x^i} - N_l^{x^i})\big)^2} \\ &- \frac{A_w^{x^j}\big(2(H_l^{x^j} + A_l^{x^j} + N_l^{x^j}) + d(H_w^{x^j} - H_l^{x^j}) + e(A_w^{x^j} - A_l^{x^j}) + f(N_w^{x^j} - N_l^{x^j})\big)}{\big(2(H_l^{x^j} + A_l^{x^j} + N_l^{x^j}) + d(H_w^{x^j} - H_l^{x^j}) + e(A_w^{x^j} - A_l^{x^j}) + f(N_w^{x^j} - N_l^{x^j})\big)^2} \\ &- \frac{\big(d \cdot H_w^{x^j} + e \cdot A_w^{x^j} + f \cdot N_w^{x^j}\big)(A_w^{x^j} - A_l^{x^j})}{\big(2(H_l^{x^j} + A_l^{x^j} + N_l^{x^j}) + d(H_w^{x^j} - H_l^{x^j}) + e(A_w^{x^j} - A_l^{x^j}) + f(N_w^{x^j} - N_l^{x^j})\big)^2}\Bigg) \end{aligned} \tag{32}$$

$$\frac{dRPI}{df} = (1-B-C)\Bigg(\frac{N_w^{x^i}\left(2(H_l^{x^i}+A_l^{x^i}+N_l^{x^i})+d(H_w^{x^i}-H_l^{x^i})+e(A_w^{x^i}-A_l^{x^i})+f(N_w^{x^i}-N_l^{x^i})\right)}{\left(2(H_l^{x^i}+A_l^{x^i}+N_l^{x^i})+d(H_w^{x^i}-H_l^{x^i})+e(A_w^{x^i}-A_l^{x^i})+f(N_w^{x^i}-N_l^{x^i})\right)^2} - \frac{\left(d\cdot H_w^{x^i}+e\cdot A_w^{x^i}+f\cdot N_w^{x^i}\right)(N_w^{x^i}-N_l^{x^i})}{\left(2(H_l^{x^i}+A_l^{x^i}+N_l^{x^i})+d(H_w^{x^i}-H_l^{x^i})+e(A_w^{x^i}-A_l^{x^i})+f(N_w^{x^i}-N_l^{x^i})\right)^2} - \frac{N_w^{x^j}\left(2(H_l^{x^j}+A_l^{x^j}+N_l^{x^j})+d(H_w^{x^j}-H_l^{x^j})+e(A_w^{x^j}-A_l^{x^j})+f(N_w^{x^j}-N_l^{x^j})\right)}{\left(2(H_l^{x^j}+A_l^{x^j}+N_l^{x^j})+d(H_w^{x^j}-H_l^{x^j})+e(A_w^{x^j}-A_l^{x^j})+f(N_w^{x^j}-N_l^{x^j})\right)^2} - \frac{\left(d\cdot H_w^{x^j}+e\cdot A_w^{x^j}+f\cdot N_w^{x^j}\right)(N_w^{x^j}-N_l^{x^j})}{\left(2(H_l^{x^j}+A_l^{x^j}+N_l^{x^j})+d(H_w^{x^j}-H_l^{x^j})+e(A_w^{x^j}-A_l^{x^j})+f(N_w^{x^j}-N_l^{x^j})\right)^2}\Bigg) \tag{33}$$

We carried out the regression across four years, and saw only a marginal improvement in error rate, variables whileless than one percent.

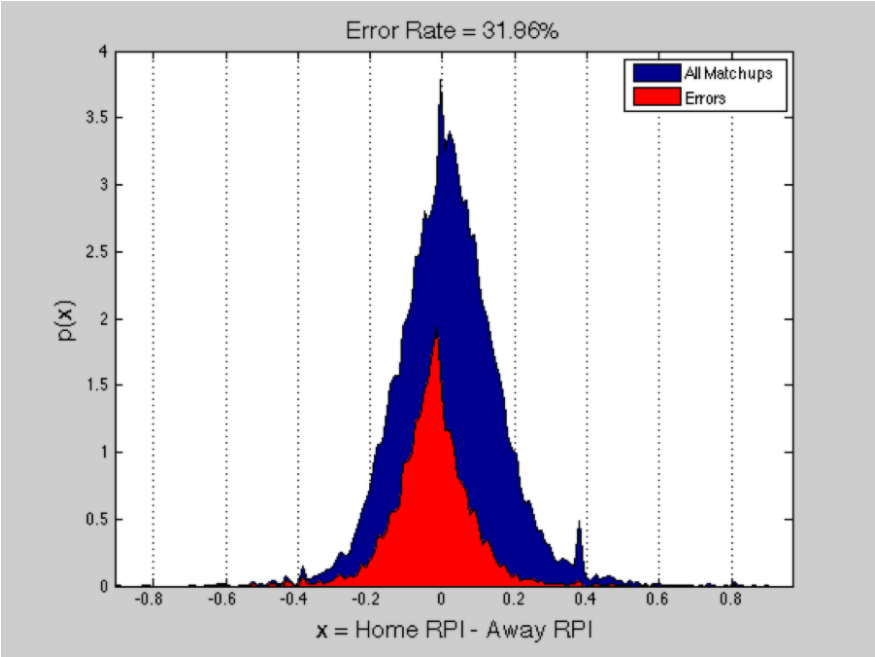


Figure 3: RPI after regression has been performed using the derivatives calculated in Equations25-33

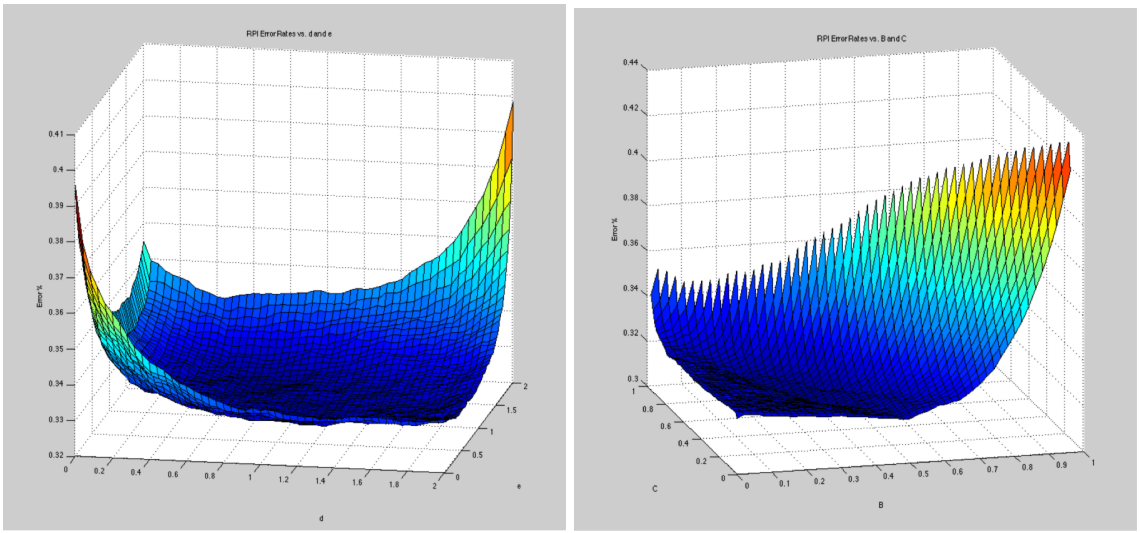
To discover why we couldnt improve the rate, we plotted several cross-sections of the error space as seen in Figure 4

It is important to note that in Figure 4 the error rate never falls below about 32% in the range. The local minimum is very wide and flat, except at the boundaries. For example, along the line $B + C = 1$, the error rate increases because, along this line, the RPI component for win percentage (WP) is null. These surface plots tell us that there is a very wide space of nearly optimal configurations in RPI, so there is not a strong justification for changing these weights. Because the NCAA has changed the coefficients in the past, we posit that they were similarly unjustified.

4.2 Game Prediction

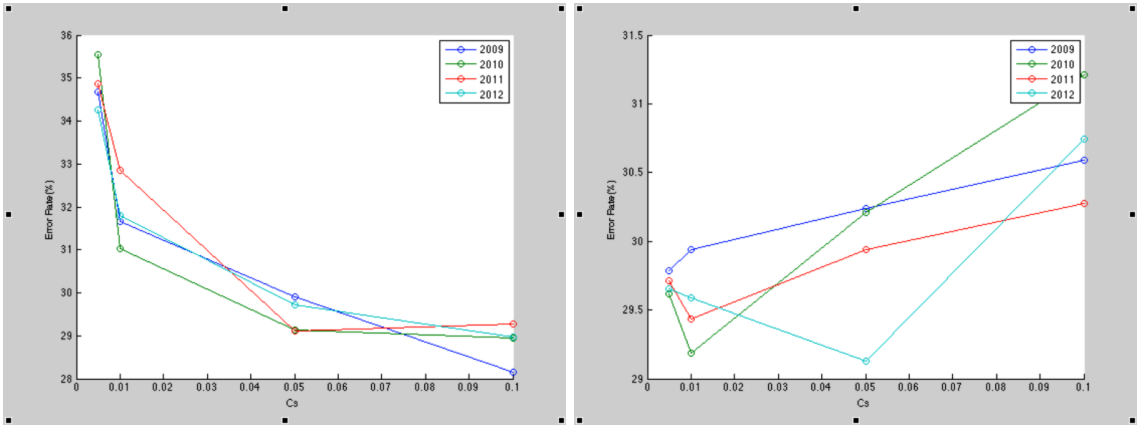
4.2.1 Crossvalidation Studies

The crossvalidation studies were done to understand how the classifiers performed against the regular season training data. Figures 5 and 6 show the results of the crossvalidation study. Decision trees and AdaBoost



(a) Error vs d and e. (b) Error vs B and C.

Figure 4: Plot of the error when using RPI to calculate the outcome of games in the regular season for while varying 2 variables and holding the others to NCAA standards.



(a) Decision Tree (b) Random Forest: forest size=11, pool size=5

Figure 5: Plots of the decison tree and random forest crossvalidation training error for different single seasons as a training set over the remaining three years as a test set.

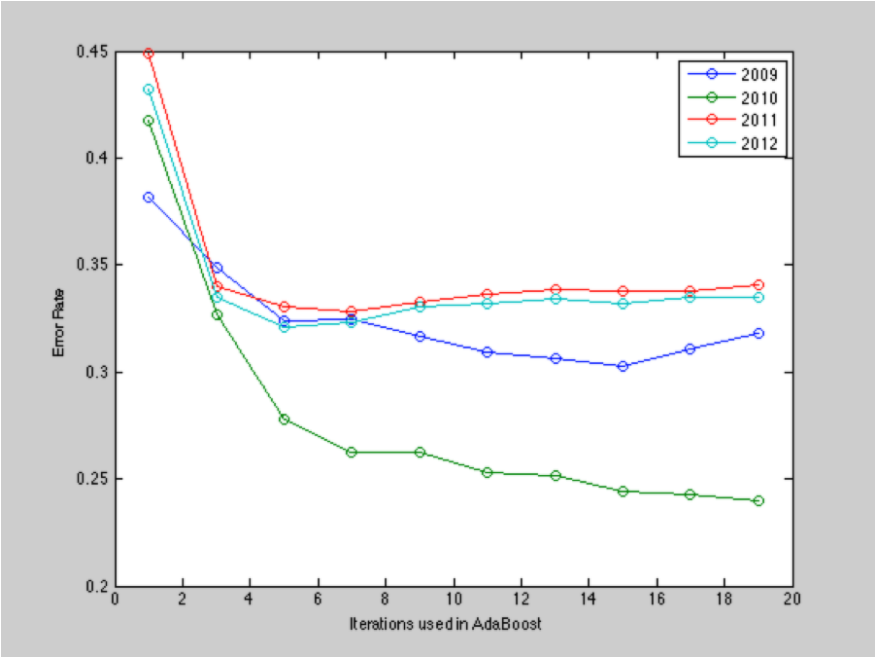


Figure 6: The crossvalidation training error when using AdaBoost predictions for different single seasons as a training set over various numbers of iterations of Adaboost classifier selection. There is a strictly decreasing error rate as the number of weak classifiers included in AdaBoost increases.

showed excellent fit even when using one season of data and crossvalidating against the other three. Oddly random forests performed more poorly than decision trees in the cross validation study.

We chose this scheme for crossvalidation because crossvalidating past seasons with future seasons seemed like epistomological failing. Back and forward projecting a single season without knowledge of the others seemed a more reasonable approach.

4.2.2 NCAA Tournament Predictions

Using the same system for predicting game-by-game expectation, we simulated NCAA tournaments for the years 2009-2012, taking the first-round 64 participants and evolving the tournament. Because the games played later in the tournament are wholly dependent on earlier matches, predictions made by chance will yield extremely low scores. This forward propagation of errors makes any error rate higher than 33.85% better than chance as calculated using Equations 34-40. After simulating the full tournament, we compared

it to our ground truth historical tournament.

$$N : \text{random variable of number games predicted correctly in a Tournament} \tag{34}$$

$$G : \begin{array}{c}\text{random variable that a particular team will win a game in a particular} \\ \text{round assuming all teams are equally strong, i.e. unbiased coin flip}\end{array} \tag{35}$$

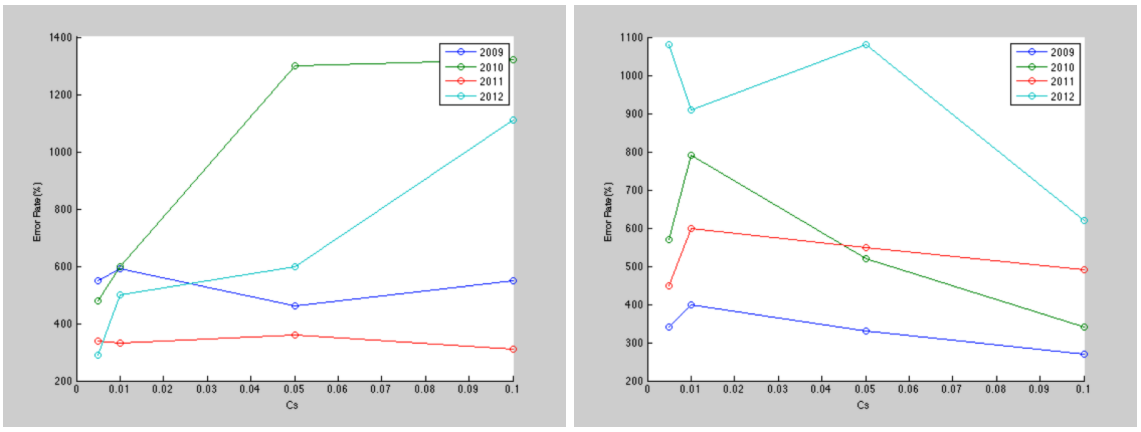
$$E[N] = \sum_{r=1}^{6} \sum_{g=1}^{2^{r-1}} P(G) \tag{36}$$

$$= \sum_{r=1}^{6} \sum_{g=1}^{2^{r-1}} (\frac{1}{2})^{7-r} \tag{37}$$

$$= 21.328 \tag{38}$$

$$\text{error rate} = \frac{E[N]}{63} \tag{39}$$

$$= 0.3385416 \tag{40}$$



(a) Decision Tree

(b) Random Forest: forest size=11, pool size=5

Figure 7: Plots of the same-year tournament errors when using decision tree predictions for different single seasons as a training set over various numbers of iterations. There is significant overfit of the classifier against the Tournament test sets.

All of the algorithms tested showed signs of overfit when tested against the tournament data. This can most clearly be seen in Figure 8 where the error rate of each AdaBoosted classifier set grows when utilizing more than one classifier. In retrospect this makes sense due to our large input classifier space. We have essentially found a set of classifiers, from a very large set of classifiers that fit the training set very, very well. Another approach to this would be to train the weak classifiers and AdaBoost on different training sets so as to remove overfit, however

It is also reasonable that tournament play is significantly different from the regular season. In Division I play there are 347 teams only 63 of which make the tournament. We included all teams in the trainin set which now seems like a poor choice. In the future we would remove matchups that did not include selected teams, though retain all of the cumulative data related to those teams, as our training set.

5 Future Work

The most interesting advancement may come from the selection of features. The group of Shu Michelle, Gediminas Bertasius, and Jason Wardy[5] showed excellent results with significantly fewer features by in-

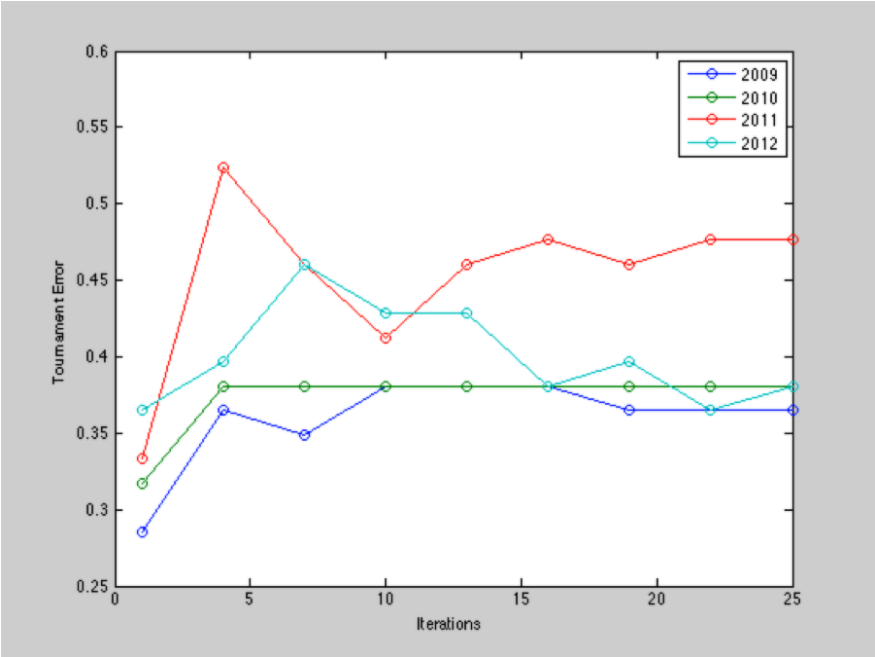


Figure 8: The same-year tournament errors when using AdaBoost predictions for different single seasons as a training set over various numbers of iterations. There is significant overfit of the classifier against the Tournament test sets.

cluding statistics about the opponents of the teams of interest. We would like to see if altering our feature set had similar performance increases.

6 Outside Resources

We used several outside resources broken down into the categories of acquiring our data and processing our data. The rest was completed in entirety by John Sigman and Harrison Hall.

6.1 Scraping Data

To scrape the data from ESPN we used the following packages:

- Django v1.4.3
- Unipath v0.2.1
- beautifulsoup4 v4.1.3
- distribute v0.6.19
- html5lib v0.95
- psycopg2 v2.4.6
- python-dateutil v1.5
- requests v1.1.0
- wsgiref v0.1.2

- yolk v0.4.3

6.2 Machine Learning Analysis

A function, mfcsvread.m, for reading .csv files into a matlab struct was taken from the Mathworks file server. The boilerplate source code for decision trees and random forests was provided by the CS 174 staff.

References

[1] K. Chong-Adler. (2012, Mar.) Espns tournament challenge sets bracket record with 6.45 million entries. [Online]. Available: http://frontrow.espn.go.com/2012/03/espns-tournament-challenge-sets-bracket-record-with-6-45-million-entries/

[2] CincyFan007. 1. [Online]. Available: http://games.espn.go.com/tournament-challenge-bracket/en/entry?entryID=970323

[3] ESPN. Ncaa-mens college basketball. [Online]. Available: http://espn.go.com/mens-college-basketball/

[4] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[5] S. Michelle, G. Bertasius, and J. Wardy. Predictive applications of ensemble methods: Nba betting. [Online]. Available: http://www.cs.dartmouth.edu/~lorenzo/teaching/cs174/Projects/FinalizedWriteups//michelle.w.shu.html

[6] B. T. West, "A simple and flexible rating method for predicting success in the ncaa basketball tournament: Updated results from 2007," *Journal of Quantitative Analysis in Sports*, vol. 4, no. 2, Apr. 2008.