Predicting NCAA March Madness Using Support Vector Machine

Zihao Xu & Xiaotong Gui
March 12, 2017

Abstract

Numerous models have been developed by statisticians to yield the optimal prediction accuracy: linear regression, classification trees, logistic regression, etc. In our approach to predict the 2017 March Madness winners, we seek to find a model that will best fit our data and give the most accurate outcome. To evaluate accuracy, the value of "r-squared "(also referred as coefficient of determination) is employed to indicate goodness of fit of a certain model. In the following steps we respectively run through three linear regression models but all of them gives a low r-squared value. This further motivates us to find a more suitable model. We finally arrive at Support Vector Machine, a binary classifier and determine that it yields the best accuracy among the four by r-squared testing. The next question of our research is: how to avoid overfitting and how to choose metrics features that would be most valuable for the prediction? To solve this problem, we examine the mean r squared of all the combinations of features of size 3 or less. Our analysis finally gives the most promising combination: AdjT_x-y" (Adjusted Tempo: possession per 40 minutes), "W_x-y" (Number of total wins), "PT_ratio_x-y" (Point ratio, total points earned by team divided by total point earned by opponents). The final prediction of 2017 is based on the combination of these three features and we believe together they yield the optimal predictability.

Getting the Data

We used three sets of data for our prediction:

1. Kenpom Ranking Statistics (2002 - 2017): **http://kenpom.com/index.php**
2. Sports Reference: Advanced School Statistics (2002-2017): **http://www.sports-reference.com/cbb/seasons/2017-advanced-school-stats.html**
3. 32 Game Results Among the Top 64 Teams (2002 to 2016): **http://www.sports-reference.com/cbb/postseason/2016-ncaa.html**

The first two datasets combined give us a set of metrics, which we will refer to as "features", of each NCAA team from 2002 to 2017; the third dataset records the actual results for the first 64 games played in each year from 2002 to 2016. In our analysis, we will strive to determine out of the 15 metrics we obtained from Kenpom and Sports Reference Advanced School data, which combination(s) of features will yield the highest accuracy in predicting the results, as determined by the values of "r-squared" (goodness of fit of a certain model). To this end, we also need the results (1. The result of who winned; 2. The point differences of the competing teams.) of actual games played to both train and test our models, which is indicated in dataset 3, the first 64 NCAA games played in each year.

To get and clean the data, we use the "Rvest" package under Rstudio to scrap data from the websites and merge all the information to what we refer to as "master" data for each year (link: https://github.com/zihaoxu/Statsketball-Tournament/tree/master/Master_Data). In this data frame, each row represents an individual game with team names and differences in metrics (for example: "AdjEM_x-y = AdjEM_x = AdjEM_y"). The following analyses are performed on the merged "master data sets" with "NAs" (missing data points) dropped.

A Quick Method Summary:

Our core idea is to use available rankings and metrics to predict game results. By construction, the two columns in our "master data" that are indicative of wins and losses are "point_diff_x-y" and "results". The "point_diff_x-y" is obtained by (score_x - score_y) * 2 and a positive point different

shows the result of team X won while a negative number means team Y won. Therefore, we took two difference approaches to make the predictions: 1. Using linear models to predict "point difference"; 2. Using binary classification to predict "X" (team X wins) or "Y" (team Y wins). We seek to choose from the two approaches, and to find a model that will produce the most accurate predictions. The accuracy of model is determined by the **coefficient of determination**, or r-squared auto-generated by the available statistics model: a higher value of r-squared indicates a more accurate prediction.

Our first approach employs three Linear Regression Models: Least Absolute Shrinkage and Selection Operator (LASSO), Ordinary Least Squares(OLS) and Random Forest to predict the point difference of each game. We split the data into 85-15 percent, train our models using the training set and test each model using r-squared with 5-fold cross validation (see :https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation). However, none of these three models produced a satisfying r-squared:

```
Index(['AdjEM_x-y'], dtype='object')
RF :
Coefficient of determination on training set: 0.84551505585
Average coefficient of determination using 5-fold crossvalidation: -0.182984197634


LASSO :
Coefficient of determination on training set: 0.435309553766
Average coefficient of determination using 5-fold crossvalidation: 0.277600997966


LINEAR :
Coefficient of determination on training set: 0.435309553766
Average coefficient of determination using 5-fold crossvalidation: 0.392700575457
```

R-squared with 5-fold Cross Validation of LASSO, LINEAR and RF

As indicated by the results, none of the regression models does a good job in predicting the point difference. Therefore, we switched our approach and attempted to predict the game result directly.

With some online research, we arrived at the model of Support Vector Machine (SVM). SVM introduces the idea of "hyperplane classifiers" which is based on constructing hyperplanes in a multidimensional space that separates cases of different class labels. In our prediction, we will use SVM to take the difference between teams' metrics and classify each game into a result of either an "X" (team X wins) or a "Y" (team Y wins). Note that all the codes of analysis and testing are written in Python.

A Detailed Analysis of Support Vector Machine:

First of all, we look to determine which of the 15 features yield the highest predictability, while trying to limit the number of features we use (since we understand that using too many features tend to overfit the model and increase r-squared even when the additional metrics actually do not add additional predictability). Therefore, we perform SVM using all combinations of features with size no larger than 3 (all single feature, combinations of two/three features) to determine which combination(s) give the highest accuracy. The specific method is as follows: in 100 repetitions, we split the merged "master" data set (including all games from 2002-2016) into a training set to fit the model (85% of the data) and a testing set (15% of the data) to see how accurate our model is. After the 100 iterations, we calculate the mean value of r-squared to determine which set of features would produce the most accurate result:

```
Average confidence w/ 100 repetitions for  ['PT_Ratio_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.723742690058
Average confidence w/ 100 repetitions for  ['OPPO_pt_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.733426900585
Average confidence w/ 100 repetitions for  ['TM_pt_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.738514619883
Average confidence w/ 100 repetitions for  ['SOS_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.736485380117
Average confidence w/ 100 repetitions for  ['SRS_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.738871345029
Average confidence w/ 100 repetitions for  ['W_L_ratio_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.723298245614
Average confidence w/ 100 repetitions for  ['L_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.682043859649
Average confidence w/ 100 repetitions for  ['W_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.767540935673
Average confidence w/ 100 repetitions for  ['luck_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.725040935673
Average confidence w/ 100 repetitions for  ['AdjT_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.710950292398
Average confidence w/ 100 repetitions for  ['AdjD_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.720301169591
Average confidence w/ 100 repetitions for  ['AdjO_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.697122807018
Average confidence w/ 100 repetitions for  ['AdjEM_x-y', 'True_S_x-y', 'seed_x-y'] w/ 5-fold CV : 0.739669590643
```

Confidence (R-squared) with 5-fold Cross Validation of SVM on some combination(s) features

Eventually, we ranked all the possible combinations of feature with size no larger than 3. The result indicated that the combination of differences in “AdjT_x-y” (Adjusted Tempo: possession per 40 minutes), “W_x-y” (Number of total wins), “PT_ratio_x-y” (Point ratio, total points earned by team divided by total point earned by opponents) have the highest predictability (average r-squared of 0.791 with 5-fold cross-validation and 100 repetitions). We will use this combination to predict the 2017 March Madness result:

```
X = np.array(d['master'][['AdjT_x-y', 'W_x-y', 'PT_Ratio_x-y']])  # 0.79051754386
y = np.array(d['master']['result'])

r_squared_lst = []
for i in range(100):
    X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.15)

    clf = svm.SVC()
    clf.fit(X_train, y_train)

    cv = KFold(X_train.shape[0], 10, shuffle = True)
    r_squared = cross_val_score(clf, X_train, y_train, cv = cv)
    r_squared = np.mean(r_squared)
    r_squared_lst.append(r_squared)
print ("Average confidence w/ 100 repetitions for ", "AdjT_x-y', 'W_x-y', 'PT_Ratio_x-y", "w/ 5-fold CV :", \
       sum(r_squared_lst)/len(r_squared_lst))
```

```
Average confidence w/ 100 repetitions for  AdjT_x-y', 'W_x-y', 'PT_Ratio_x-y w/ 5-fold CV : 0.79635
```

Confidence (R-squared) with 5-fold Cross Validation of SVM on [‘AdjT_x-y’, ‘PT_ratio’, W_x-y‘’]

Testing and Prediction

Before we predict the 2017 result, we will test our model on previous games played from 2002-2016. The testing shows that our method obtained an accuracy rate of approximately …….
(SHOW SCREENSHOT)

This decent accuracy gives us confidence to predict the 2017 March Madness. Run the code again and the prediction results are given as the following:

	team_X	team_Y	AdjT_x-y	W_x-y	PT_Ratio_x-y	predicted_winner
0	Villanova	Mt St.Mary's/New Orleans	-2.8	12	0.240862	Villanova
1	Wisconsin	Virginia Tech	-5.0	3	0.119829	Wisconsin
2	Virginia	UNC Wilmington	-11.3	-7	0.060613	Virginia
3	Florida	East Tenn	-1.2	-3	0.023809	Florida
4	SMU	Providence/USC	-2.8	9	0.190788	SMU
5	Baylor	New Mexico St.	-4.5	-3	-0.014510	Baylor
6	South Carolina	Marquette	-1.8	3	0.017575	South Carolina
7	Duke	Troy	-0.1	6	0.060891	Duke
8	Gonzaga	South Dakota St.	3.0	14	0.326169	Gonzaga
9	Northwestern	Vanderbilt	-0.4	4	0.046927	Northwestern
10	Notre Dame	Princeton	4.0	3	-0.044197	Notre Dame
11	West Virginia	Bucknell	0.2	0	0.105352	West Virginia
12	Maryland	Xavier	-0.1	3	0.047794	Maryland
13	Florida St.	FGCU	5.6	-1	0.002262	Florida St.
14	St. Mary's	VCU	-8.8	2	0.142709	St. Mary's
15	Arizona	North Dakota	-5.5	8	0.059500	Arizona
16	Kansas	NC Central/UC Davis	3.9	3	-0.042072	Kansas
17	Miami	Michigan St.	-3.7	2	0.040101	Miami
18	Iowa St.	Nevada	-1.5	-5	-0.003964	Iowa St.
19	Purdue	Vermont	4.6	-4	-0.001252	Vermont
20	Creighton	Rhode Island	4.7	2	0.001281	Creighton
21	Oregon	Iona	-3.8	7	0.155256	Oregon
22	Michigan	Oklahoma St.	-7.4	3	0.041954	Michigan
23	Louisville	Jacksonville St.	4.4	4	0.146634	Louisville
24	North Carolina	Texas Southern	2.2	4	0.166365	North Carolina
25	Arkansas	Seton Hall	2.1	4	0.043398	Arkansas
26	Minnesota	Middle Tennessee	5.1	-6	-0.095385	Middle Tennessee
27	Butler	Winthrop	-5.6	-3	-0.020231	Butler
28	Cincinnati	Kansas St./Wake Fst	-2.4	9	0.168334	Cincinnati
29	UCLA	Kent St.	4.9	7	0.135906	UCLA
30	Dayton	Wichita St.	-0.1	-6	-0.164809	Wichita St.
31	Kentucky	Northern Kentucky	4.9	4	0.132508	Kentucky

Final Result:

Accuracy rate = (32-5)/32 = 84.375%

Sources:

http://kenpom.com/index.php
http://www.espn.com/mens-college-basketball/statistics/team/_/stat/scoring-per-game/sort/avgPoints
https://courses.cs.washington.edu/courses/cse140/13wi/projects/jarrison-report.pdf
http://www.sports-reference.com/cbb/seasons/2017-advanced-school-stats.html

https://www.nytimes.com/2015/03/22/opinion/sunday/making-march-madness-easy.html?_r=1
https://www.degruyter.com/view/j/jqas.2015.11.issue-1/jqas-2014-0058/jqas-2014-0058.xml?format=INT