Levi Franklin

Lefrankl - #05548939

CS229 Final Project

12/08/14

Predicting March Madness: Winning the Office Pool

I. Introduction

Each year at the end of the NCAA basketball season a massive tournament, called "March Madness", is played to determine the NCAA champion. The tournament consists of 64 of the "best" teams during the regular season competing to win the designation of champion. When the teams are announced, they are placed into 4 different regions and seeded 1-16 based on their performance during the regular season. During March Madness, people all over the nation compete to fill out a bracket predicting the outcome of every game of this tournament. People look at many metrics such as the end of season rankings, seeds, team records, and many others. My goal in this project is to take a large historical dataset of basketball season results and attempt to predict the outcome of the March Madness tournament.

II. Dataset

I have acquired a large dataset of historical data for NCAA division 1 basketball seasons and their respective tournament results. This dataset was acquired from Kaggle and was compiled by Kenneth Massey. The dataset provides several files:

- Seasons – A list of every season, its dates, and an identifier for the rest of the files.
- Teams – A list of all the teams and an identifier.
- Regular Season Results – A list of every game that was played, the winning and losing team as well as their scores, and other information like date.
- Tourney Results – A list of all the games that were played in each tournament and their results.
- Tourney Seeds – A list of each team in the tournaments and what seed they were (1-16).
- Tourney Slots – A description of the format of each tournament, informing which teams played each other and in what order.

This data is very comprehensive and is much of the same data that sports analysts and layman alike use in making their own predictions. The data spans 18 seasons from 1995 – 2013. The goal is to process this data in a way that an analyst could not by utilizing the machine learning techniques we have used in class.

III. Features and Preprocessing

One of the primary difficulties of this project lies in extracting features from the large dataset. While the data is interesting in its original format, it is not very useful for machine learning problems. I wrote a C# application to assemble all of the data into a useable form, and then process it to extract features. After looking at the data for an extended period of time, I decided on the following features to begin my investigation:

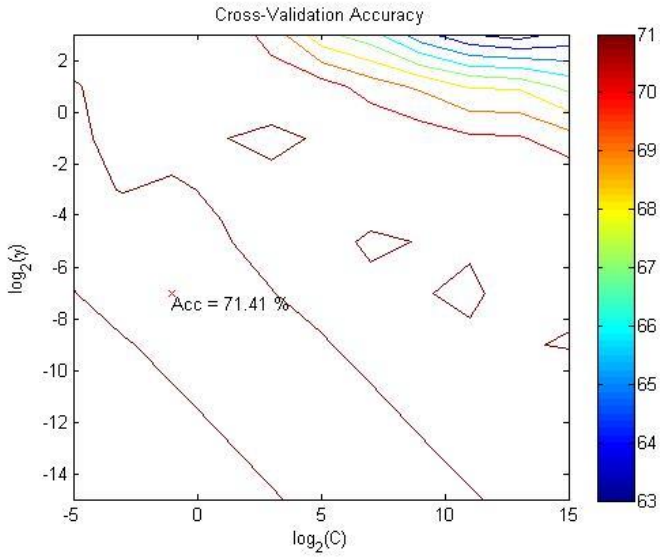
- Team 1 margin of victory over team 2 in regular season
- Difference between team 1 and team 2 seeds
- Difference between margin of Victory for Team 1 over common opponents and margin of victory for Team 2 over common opponents
- Difference between Team 1 wins and losses for season and Team 2 wines and losses for the season
- Teams performance during last year's tournament (number of games played)
- Difference between overall margin of victory during season for team 1 and team 2

These features are similar to things that an analyst might look at in determining the strength of a team, though some are hard to calculate without using computers. An interesting feature is the seed, because it is not a result of the team's performance. This feature comes from a group of analysts looking at the teams, and determining who they think is best based on information similar to our other features. Therefore, I think it is also interesting to remove that feature from our training, despite it being a good indicator of success, and see how our model performs.

IV. Models

Once I had preprocessed the data I was left with a very large training set. I had 1223 games that occurred during all of the tournaments throughout the 18 seasons. Because every game used a comparison of the winning team to the losing and I made every feature just a comparison between two teams, I made a copy of each row and inverted the features and labels so that we would have some negative training examples as well as positive. This gave 2446 total rows that I used to train.

My initial experiment was to build an SVM model. I considered this problem to be a classification problem asking "Does the first team win or not?" To achieve this I used libSVM to train over my 6 features. Initially, my result was ok, but not great. I was getting much better results than a 50/50 guess at winners but was not scoring very well compared to the Kaggle leaderboard for this problem. To combat this, I began doing cross validation of my model. The kernel I used was a radial basis function kernel, as they suggested in the libSVM documentation. I used cross validation to train the parameters of my model by maximizing cross validation accuracy and achieved an accuracy of ~70%. This resulted in a testing accuracy of around 64%



I next decided to try logistic regression as it might give a better matching. I implemented logistic regression in matlab and went about training my model. I used stochastic gradient ascent with the update rule discussed in class. After experimenting with various learning rates I was able to get it to converge. This resulted in similar results to the SVM.

After thinking more about the issue, I realized that assigning each game a class of win or lose leaves out some valuable information. A game where a team wins by one point is very different than a blow out by 20 points. After thinking about this I decided to consider this as a regression problem. Instead of assigning a class of win or lose, I set each training label to be the margin of victory of the game.

I decided to use a similar technique as in my classification modelling by using libSVM regression. I again used cross validation, though this time to minimize the mean squared error. I was able to get the mean square error down to around 130. Unfortunately, this also had similar results to the classification problem.

Lastly, I implemented linear regression to see if it offered any advantages. Instead of using gradient descent I decided to use the normal equation to save myself development time. The thetas resulting from this computation provided me some insight into what features were most prominent in my models and helped me understand why achieving high accuracy is difficult in this problem.

V. Results

First, I compare my results to the results that you would get by assigning each team a 50% chance in every game. Intuitively, this is the approach that one would take without any

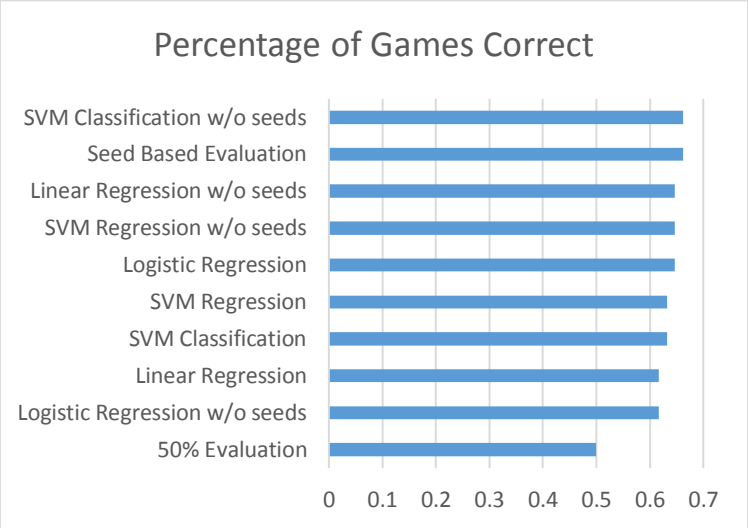
prior knowledge about the teams. This obviously results in a very poor accuracy as only 50% of games will be chosen correctly.

Second, I compare my results to the results that you would get by basing teams' success on their seed. Any time a team plays another team with differing seeds, the higher seed wins. This is the approach many people take when filling out march madness brackets and it is a reasonable approach. Because these seeds are chosen by experts looking at very similar data to what I am, I hope to get similar results to this approach or slightly better. Additionally, I want to get similar results to this metric without taking seeds into account in my model.

Lastly, because this started as a Kaggle competition, there is a leaderboard with lots of information about how other students scored. Many of these students are probably very skilled and put in much more time than a class project as $15,000 was on the line, so I do not expect to have results on par with the leaders. However, I expected to be in the range of some of the medium competitors.

Method	**Correct Games**	**Percentage**
50% Evaluation	32	50.0%
Seed Based Evaluation	45	66.2%
SVM Classification	43	63.2%
SVM Classification w/o seeds	45	66.2%
Logistic Regression	44	64.7%
Logistic Regression w/o seeds	42	61.7%
SVM Regression	43	63.2%
SVM Regression w/o seeds	44	64.7%
Linear Regression	42	61.7%
Linear Regression w/o seeds	44	64.7%

Method	**Kaggle Score**
50% Evaluation	0.693
Seed Based Evaluation	0.600
SVM Classification	0.597
SVM Classification w/o seeds	0.622
Logistic Regression	0.621
Logistic Regression w/o seeds	0.658
SVM Regression	0.608
SVM Regression w/o seeds	0.634
Linear Regression	0.592
Linear Regression w/o seeds	0.621



VI. Discussion and Conclusions

As can be seen from the results above, all models did nearly as well as or better than the seed based evaluation. Every model did significantly better than the 50% evaluation method. This shows that these features are effective at predicting winners of games when trained using machine learning. Even more interesting is that in many cases the models performed better without the seeds than when using seeds as a feature. I believe this shows that these models can predict march madness based on data alone, independent of any panel's opinion of the team. Furthermore, this system is a very good way of ranking the teams and could be used for seeding them very similarly to the panel.

I think it is also prudent to compare these models to the Kaggle scores. Kaggle mentions the average score on its leaderboard of 0.576. These scores are calculated using a log loss function. My best log loss model score is 0.592 for linear regression. This would place me at number 111 place out of 248 submissions. I think this is a respectable number in that they had more time and could use outside data, which I did not have to take advantage of.

Overall, I think these results are very successful. After taking a large dataset of historical data on games, I was able to get a 66% success rate for predicting tournament game winners. I think this is better than the average person could do by just looking at teams. It is interesting to see how the inclusion or exclusion of features impacts the results for various machine learning algorithms. Though the problem of predicting march madness tournaments seems to be too random for machine learning to do extremely well, these models can definitely provide insight into how a tournament will progress.

VI. References

[1] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm