Contact information

Isaac Hess, Jacob Jashinsky, Alec Otterson

Kaggle.com: Historical NCAA data

Introduction

We chose to use the data provided through Kaggle for a machine learning competition to predict the bracket of the 2018 NCAA basketball national championship. This problem is interesting because there are a lot of variables that go into sports competitions, including a lot of statistical data and random variables such as the human aspect. The challenge of putting an entire bracket together is particularly challenging, as things can change even from game to game. There is a large repository of data that can lend to helping predict likely outcomes, though, and it has become a popular pastime for people and groups to try their hands at predicting the likely final bracket.

We decided to use our algorithm to take in the teams' previous game stats to predict the next match up. We then took out data and algorithm to predict the semi-finals outcomes. We then used those stats to predict the final.

Data Preparation

We were able to download our needed data from Kaggle.com easily enough, in csv form. We had to do some preprocessing of the data in order to get it into a format that we could use. Since the data is made up of two teams per game, we were uncertain how to go about trying to isolate a particular team and use their data against another team that was not a part of that game. We decided to use the data for a particular team from their last game and then take the data of the team they are playing and subtract one from the other. This gave us a dataset with a comparison of each team and whether that first team beat the second team or not. In order to achieve this, we had to split the original dataset by each team, determine which number game in the dataset the game was for each team, and then pull the data for each team and merge it back into a final dataset for the current game.

Another issue we faced was the first team listed was always the winner, thus the distribution of classes was always the same or homogenous in other words. To overcome this we randomly sampled half of the data and switched the teams and made a new column that indicated if the team won. This was quite a coding challenge at first but after much collaboration we were able to obtain the desired dataset. Our final data has 20 columns and 79,553 rows.

Mining / learning from the data

We started off by running the data through a bunch of algorithms to see which ones would perform the best. We tried decision trees, neural networks, bagging classifier with kneighbors, bagging classifier with neural networks, ada boost, gradient boosting, random forest, support vector machines, KNN, and naive bayes. Support vector machines algorithms were very difficult to really fine tune due to our large dataset. But after using a smaller data sample we learned that the SVM did not fare much better than any other algorithm.

Out of these, the neural network with 100 hidden layers, logistic activation, adam solver, learning rate of 0.001, max iterations of 400, momentum of 0.9, and early stopping of max 10 iterations with no change was the best.

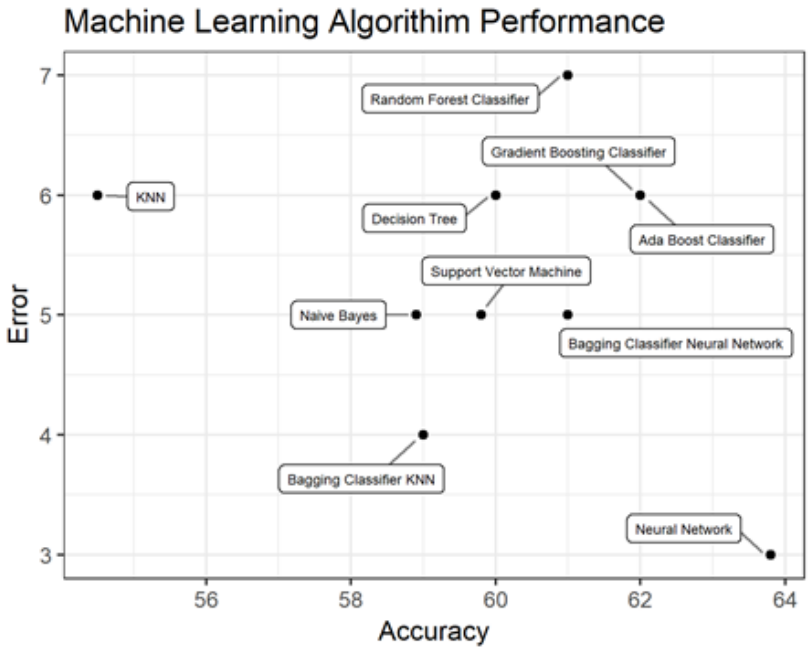
One other neural network that performed well consisted of 5 hidden layers, 1000 iterations, adam solver, the logistic activation function, learning rate of 0.00, and momentum of 0.9.

We also attempted to train the same algorithms on the data before the differences were made on the features. It seems that the results tended to be a bit worse off but not by much, if any. We also considered dropping the team_id columns but that also decreased accuracy.

Results

We ran the various machine learning algorithms and received the following results.

Classifier	Accuracy	Error	True positive	True negative	False negative	False positive
Decision Tree	60	6	61.5	63.7	38.5	36.3
Neural Network	63.8	3	65	62.6	35	37.4
Bagging KNN	59	4	61	60.9	39	39.1
Ada Boost Classifier	62	6	63.5	62.7	36.5	37.3
Gradient Boosting Classifier	62	6	64	63.5	36	36.5
Random Forest Classifier	61	7	62.9	62.9	37.1	37.1
Support Vector Machine	59.8	5	60	59	40	41
Bagging Neural Network	61	5	57.1	66.6	42.9	33.4
Naive Bayes	58.9	5	58.2	59	41.8	41
KNN	54.5	6	54.5	55	45.5	45



We selected the Neural Network as our algorithm of choice due to it having the best accuracy overall as well as the lowest error amount. We then attempted predictions of the Final Four games and the Championship game.

Semi Finals:

Prediction: Michigan State will beat Texas Tech

Result: Texas Tech won (Incorrect)

Prediction: Virginia will beat Auburn

Result: Virginia won (Correct)

Finals:

Prediction: Virginia will beat Texas Tech

Result: Virginia won (Correct)

Total Accuracy: ⅔

The results of our predictions with the Neural Network of the final three games of this season was 66%, which is very similar to our test results of the various games that we

tested from the data set 64%.

Conclusions (including business takeaways and action items)

We learned that a neural network using stats data from the previous games of the teams in question could predict the winner with ok accuracy. These results could probably be improved by adding more data. Sports are sort of interesting in that there are a ton of variables involved leading up to a game and within the game itself, so it'd be a monumental task to achieve accuracy beyond a certain point. The accuracy our predictor ended up with, given the data we gave it, is honestly pretty decent considering how complex the system is.

Our results provide a quick and simple way to make conjectures about NCAA game outcomes. This could be used for gambling purposes. A prediction rate better than 50% means profit. And 64% accuracy is relatively good. Businesses could use the model to predict what merchandise they need to produce to optimize their processes that way. For example, if I'm in the business of t-shirts representing whoever wins, I could produce more shirts focused on the team that is more likely to win and not spend so much time and money producing shirts for the predicted loser, and it would work out to my benefit ⅔ of the time.

It is pretty interesting that our model can predict the winning team 2/3 of the time just based on the stats of the last game. This isn't extensive data by any means, but apparently it is sufficient to give us a prediction rate much greater than a simple coin toss.

Gambling in general could be seen as unethical, and this definitely lends itself to that past-time. Our data as it is is pretty bland and un-controversial. In a future iteration, however, there is a real probability of including team and location information that may identify the race of the players and coaches or the cultural and demographic environment of locations. This could lead to certain types of players being avoided by teams based on race or players avoiding certain locations that because of demographic data. These would be topics to think about as that sort of data is incorporated.

Lessons Learned

Considering the variability of teammates and opponents I think we learned an effective way of predicting games result with a decent level of accuracy. We also learned

how to create all new features for a dataset. This was unique to some learning problems because we could not use the current game data because the prediction would become trivial. We had to create a new features to train on.

We would approach it from the same angle but add more features to the dataframe. We could add an aggregation of 3 or more games, not just the previous game. We think that including more data about actual team members could be beneficial because teammates change to different teams all the time and teams are never quite the same. We could also look at including the stats of the previous opponents. Maybe the team did well last game because they were facing an easier opponent, or if the last game was really close.