March Machine Learning Mania 2017, 2nd Place Winner's Interview: Scott Kellert

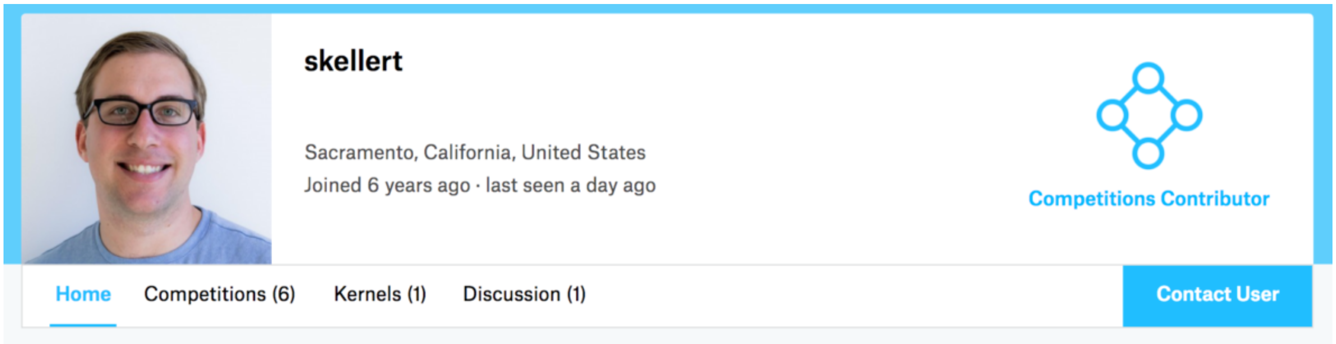
Kaggle Team (http://blog.kaggle.com/author/kaggleteam/) | 04.28.2017

Kaggle's annual March Machine Learning Mania competition (https://www.kaggle.com/c/march-machine-learning-mania-2017) returned once again to challenge Kagglers to predict the outcomes of the 2017 NCAA Men's Basketball tournament. This year, 442 teams competed to forecast outcomes of all possible match-ups. In this winner's interview, Kaggler Scott Kellert (https://www.kaggle.com/skellert) describes how he came in second place by calculating team quality statistics to account for opponent strength for each game. Ultimately, he discovered his final linear regression model beat out a more complex neural network ensemble.

The basics

What was your background prior to entering this challenge?

I work as a data scientist at Nielsen doing Marketing ROI Attribution for digital ads. I got my Bachelors in Industrial Engineering from Northwestern University and my Masters in Analytics at the University of San Francisco.


skellert

Sacramento, California, United States
Joined 6 years ago · last seen a day ago

(https://www.kaggle.com/skellert)

SCOTT KELLERT (HTTPS://WWW.KAGGLE.COM/SKELLERT) ON KAGGLE.

Do you have any prior experience or domain knowledge that helped you succeed in this competition?

I have no specific training or work experience in the field of sports analytics. However, as a die hard Oakland A's fan, Moneyball is a near religious text for me. I have always tried to read up on the most cutting edge sports analytics trends. I have done side projects in the past to prep for fantasy football and baseball leagues as well as analysis on the existence of clutch hitters in baseball (there aren't) and ongoing work on an all encompassing player value metric for the NHL.

How did you get started competing on Kaggle?

I started on Kaggle with the 2015 edition of the March Machine Learning Madness competition. I entered with two of my peers from grad school for our Machine Learning final project. While I have dipped my toes in a few other competitions, the March Madness competitions are the only ones I have pursued seriously and I have done so each year since.

Let's get technical

Before I get to answering these questions, I would like to direct people to the website that I built using the results from this competition. While this competition was geared towards producing a probability for every possible match up, I built the website to be used as a guide for filling out a bracket. It also contains many of the outputs from my analysis in an easy to digest format. I hope you enjoy. (NB I am not a web developer. This site does go down occasionally and there are small bugs. I am always trying to improve though so feel free to reach out if you have some feedback.)

www.pascalstriangleoffense.com (http://www.pascalstriangleoffense.com/)

Did any past research or previous competitions inform your approach?

With this being the third year in a row that I have participated in this competition, I was able to reuse a great deal of the work from prior years. However, I make a point to tweak and improve my model in significant ways each year.

What preprocessing and feature engineering did you do?

Preprocessing and feature engineering is the most important part of my process for this competition, and I believe it probably is for many competitors as well. Unlike some other Kaggle competitions, the training data does not come in a format that allows for any algorithms to be applied without preprocessing. Each row is a game box score which is information you will not have yet at the time of prediction, and it contains no information about the teams' prior performance.

While there are many services that will provide analytically driven statistics on team quality (most notably KenPom), I set a goal to perform all the calculations myself. In college basketball the concept of adjusting team statistics for opponent strength is crucial. Teams play most of their games within their own conference and these conferences vary wildly in skill. Therefore, a team can produce inflated stats in a bad conference or deflated stats in a great conference. Adjusting these statistics for opponent strength will make a big impact on the quality of the predictions.

For example, we could describe a team's offense by taking its average points scored across the season. Applying this to the 2015-2016 season, the top 5 offensive teams in the country would be Oakland, The Citadel, Marshall, North Florida, and Omaha. None of these teams even made the tournament. Applying the opponent adjustment algorithm reveals that North Carolina was, in fact, the best offense in the country. The Tar Heels made it all the way to the final.

The algorithm for applying the adjustment itself is relatively simple but is computationally expensive. The idea is that for every statistic we want to adjust we'll give each team a relative score, meaning it has no units or direct interpretability. Every team starts with a score of 0 which reflects our lack of knowledge about the system before the optimization begins. The next step is to generate a score for every team and every game. This score is a reflection of how well that team did with respect to the given statistic in the given game. For point differential, which is the metric I use for overall team quality, the score is calculated using the pythagorean expectation for the game.

$$PythagoreanExpectation = \frac{Points^9_{For}}{Points^9_{For} + Points^9_{Against}}$$

The rest of the statistics are absolute, meaning that unlike point differential there is no against portion of the formula. For example a team's ability to produce blocks, ignoring the blocks they allow. In this case I produce a score using the p value produced from the normal distribution.

$$StatScore = NormalCDF(\frac{Stat_{game} - Stat_{LeagueAverage}}{StdDev_{Stat}})$$

After game scores are calculated for every game, a team's overall statistic score (which started at 0) is updated to be the average of the sum of their game score and their opponent's overall score across all the games in the season. In the case of the absolute stats, the opponent's score is their score for preventing that stat, not producing. Then the whole process is repeated until the scores converge. These scores typically end up being distributed between -1 and 1. The interpretation is that a team with a score of 0 would be expected to tie an average team in the case of point differential or produce an average amount of a given statistic against a team that is average in preventing it. As the score diverges from 0 there is not as simple an interpretation but it can be taken to mean absolute ability.

One additional wrinkle that I added this year was to create an opponent adjusted score for each game. In the past I created one score for the whole season, but this introduced data leakage in the training set. A team's score was impacted by every game from the regular season but also used to predict outcomes from games in that season. By calculating these scores by game, I had a training set where I could predict regular season outcomes based on statistics from all the games from that season except for the game being predicted. This created a minimal but significant improvement to my results.

What supervised learning methods did you use?

Going into the competition I was committed to using Neural Networks as my driving algorithm. In the past I have tried many algorithms and ensembling techniques but Logistic Regression has always won out (I find this to be true a surprising amount of the time across all data science projects). After testing many parameters, I found that my Neural Net worked best with a single relatively small hidden layer and that it had high variance. This realization sparked the idea that I should be bagging my Neural Nets, and sure enough this finally allowed me to surpass Logistic Regression.

However, a later discovery that I will cover below caused Linear Regression to easily beat my excessively complex Neural Net Ensemble. Linear Regression was my final model.

What was your most important insight into the data?

As teased above, my most important insight was to predict continuous point spreads instead of binary wins/losses. I always knew that this was the better approach, but I hadn't thought of a good way to convert those results into probabilities as the competition requires. In previous years I had tried to use regressors to predict the pythagorean expectations, described above, but these never performed as well as classifier solutions. This year it occurred to me that I could use the concept of the prediction interval from Linear Regression to produce probabilities. I simply calculated the standard error of my point spread predictions (typically around 10.55) and used the normal CDF to produce a probability. This approach performed significantly better than my classifier.

Which tools did you use?

I do almost all my work in ipython Jupyter Notebooks. I find the notebooks to be the easiest way to quickly iterate on both code as well as algorithms. Sklearn, numpy, scipy, and pandas are my drivers within python. Sklearn drives all my ML algorithms, scipy covers my statistical distributions, and numpy/pandas cover all my data engineering.

I will occasionally export data to R if I want to visualize something. This is mostly because I have much more experience with ggplot than with matplotlib.

How did you spend your time on this competition?

Over the course of the three years that I have worked on this competition, the feature engineering task of creating the opponent adjustment algorithm was the most work intensive piece. However, over the last two years, I have only had to tweak that code which gives me a lot more time to work on the machine learning component. I spent significant time this year improving my cross validation approach for testing new estimators and playing with Neural Nets that I did not end up using.

What was the run time for both training and prediction of your winning solution?

Most of the time is committed to running the opponent adjustment optimization. For all 15 seasons of data to be adjusted takes around two hours. Once that process is complete, training and applying the Linear Regression takes less than a minute.

Words of wisdom:

What have you taken away from this competition?

Small tweaks can make a big improvement in the leaderboard. My approach did not change all that much between this year and last year, but the small tweaks that I discussed above took me from finishing around the 60th percentile in 2015 and 2016 to second place this year. If you didn't do well this year, you could be one minor adjustment away from placing next year!

Do you have any advice for those just getting started in data science?

- Learn how to write production worthy code. It doesn't really matter how good your algorithm is if no one can put it to use.

- Interpretable results are frequently more important than the most accurate results. Logistic and Linear Regression may be old and boring but they are still frequently the most accurate and much more interpretable than other estimators.

Bio

Scott Kellert (https://www.kaggle.com/skellert) is a data scientist at Nielsen doing Marketing ROI Attribution for digital ads. He has his Bachelors in Industrial Engineering from Northwestern University and my Masters in Analytics at the University of San Francisco.

LINEAR REGRESSION (HTTP://BLOG.KAGGLE.COM/TAG/LINEAR-REGRESSION/)

MARCH MANIA (HTTP://BLOG.KAGGLE.COM/TAG/MARCH-MANIA/)

[SPORTS ANALYTICS (HTTP://BLOG.KAGGLE.COM/TAG/SPORTS-ANALYTICS/)](http://blog.kaggle.com/tag/sports-analytics/)

What's your dream travel experience?
Win up to a $2,500 scholarship to study or volunteer in a country of your choice.
Scholar TRIPS
INTERNATIONAL TRAVEL SCHOLARSHIPS
Enter today

0 Comments No Free Hunch 1 Login

♡ Recommend 1 Share Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

✉ Subscribe Add Disqus to your siteAdd DisqusAdd Privacy DISQUS

(https://www.facebook.com/kaggle) (https://twitter.com/kaggle)