My First Kaggle Competition - Football Match Probability Prediction

This blog post will cover the main takeaways from my first kaggle competition. It will be broken down into the following sections:

* A brief overview of the competition, my approach, and my score.
* Things I wished I'd known before starting.
* Importance of understanding the dataset + thoughtful feature engineering.
* The models I used and their performance.
* Places to improve and my next steps.

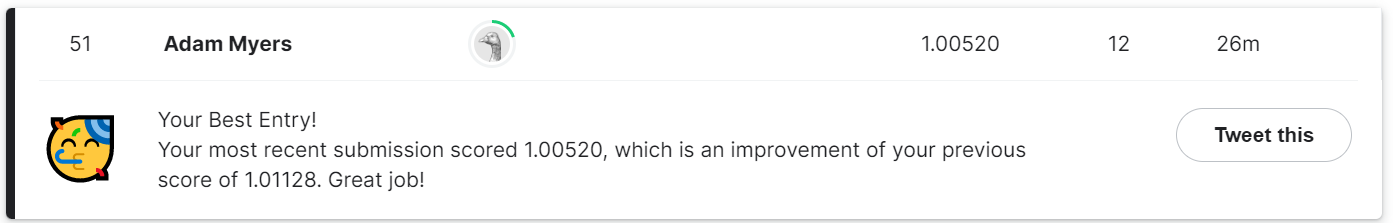
**A brief overview of the competition, my approach, and my score**

The aim of [this competition](https://www.kaggle.com/competitions/football-match-probability-prediction/overview) was to predict the match outcome of 189 games of football using statistics from the team’s last ten games.

My general approach was to extract short, medium, and long term averages of different features included in or engineered from the original data set. I used feature importance from a decision tree forest to see if these signals had good predictive power or not before ensembling a few different kinds of models for a final submission.

The predictions were judged using a log-loss score so overconfidence was punished. I scored a respectable 1.00520. For context, the leader scored 0.98834, and the bookies 0.9730. My real aim for this competition was to get used to the end-to-end process of predictive modelling and to this means it was a success.

Here’s the screengrab which, at the time of writing, put me in 51st out of 288 position:



**Things I wished I'd known before starting**

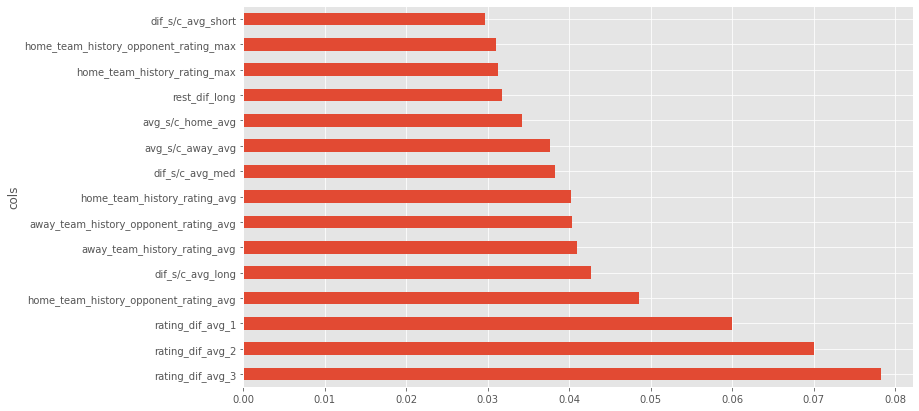
I’ll put these in a list for ease:

1. Once testing is complete, train a final model with the whole data set provided - not just the training split. This hampered my progress quite a lot before I noticed but it was a valuable learning point nonetheless.
2. Don’t be afraid to do some hard coding. By this I mean looping through variable names caused complications and lead to mistakes. If in doubt just copy and paste a few times to be safe.
3. Use box and whisker plots or histograms after each newly added feature to check things went as planned.

**Importance of understanding the dataset + thoughtful feature engineering**

Most of my gains didn’t come from the actual model I chose - in fact I hardly optimised these at all - they came from deriving useful metrics from the data and getting rid of useless ones.

By looking at how each feature split in a tree improves the model's predictive power, feature importance can be gauged. Here’s a bar chart of my top fifteen features:



Most of the variables are fairly self explanatory; however, I'll add that ‘dif’ refers to a difference between the home and the away team; ‘1’,’2’,’3’ and ‘long’,’med’,’short’ refer to the respective time average taken; and ‘s/c’ is the difference between scored and conceded goals. 

This shows how explicitly feeding the model differences really helps it categorise games. I was most surprised to see how ‘rest\_dif\_long’, which is the long term average rest differences between games between the home and away team, is judged as fairly important by the decision trees.

Team ratings were provided by [Ocotosport](https://www.octosport.io/), and again the relative difference between the averages proved most useful.

I didn't apply any explicit time series transforms to this data but I'd be interested to see how decomposing ratings trends into signal components with a fourier transform of some kind may provide extra signal - having said this, extra complexity always comes at a price.

**The models I used and their performance**

My final submission was an ensemble of a logistic regression model; a random forest mode; a light gradient boosted model; and a two layer neural network. Okay this was overkill but it was more for learning the strengths of different models - ensembling was just an easy extra kick up the leaderboard.

Surprisingly simple a logistic model alone would have scored very well. This speaks a lot for simple models and is definitely a good take away: start simple, the rest is limiting returns.

The shallow neural net extrapolated really well out of its training batch. I used fast.ai’s tabular model, which incorporates lots of under the hood tricks to work well on tabular data. Their high level API helped to split categorical and continuous features; make the entity embeddings for categorical features; and find a good learning rate for a learning rate varied fitting cycle.

**Places to improve and my next steps**

The glaring area for further research is running a grid search on my tree based models. To do this I think I'll have to rent a better GPU as the free version on [Paperspace](https://www.paperspace.com/) simply took too long.

Having said this, for a first attempt, I'm happy with my performance as I picked up the main points about making a good model. Optimising little things is the aim of the game of Kaggle competitions but for me it's about learning as much as I can.

My plan now is to learn some more about natural language processing models and do a deep dive into the maths behind.