False Analysis for Scrabble Rating Prediction Project

In order to predict the rating of a scrabble player, we used an XGBoost model. In this note we will analyze the errors of the model and try to understand what causes them and how to improve to model based on that information.

train and validation RMSE:

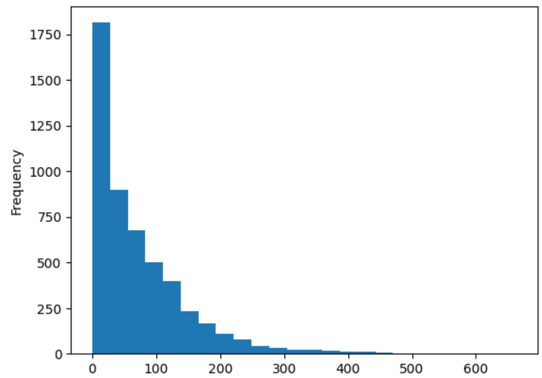
set	RMSE score
validation	105.045
train	85.112

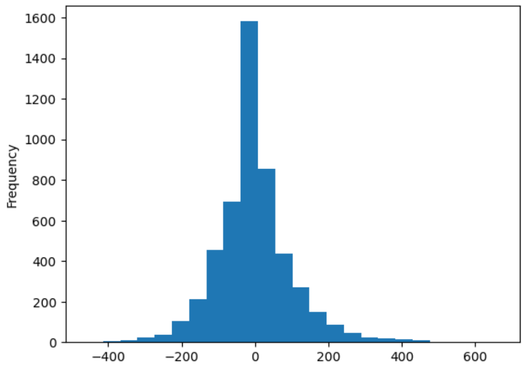
we define error and abs_error as:

$error = prediction - rating$

$abs\_error = |prediction - rating|$

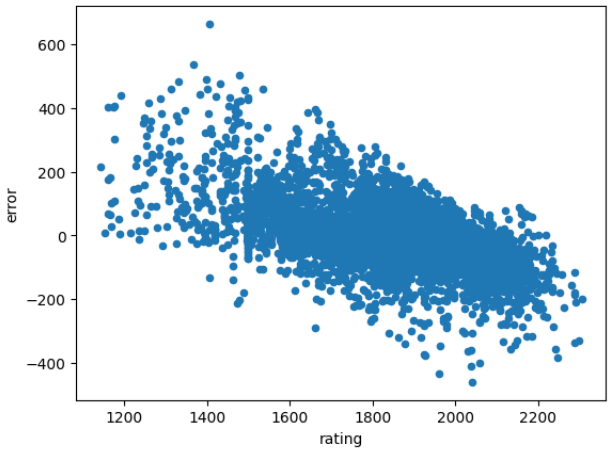
Behavior of the error and absolute error on the validation set:



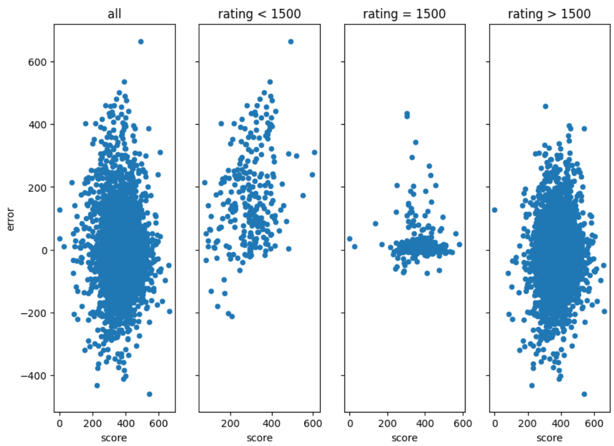


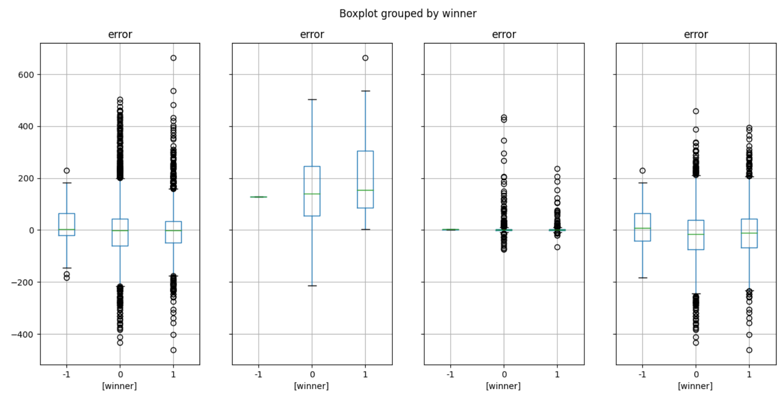
We can see they behave as expected, abs_error frequencies are decreasing and error is normal with mean zero.

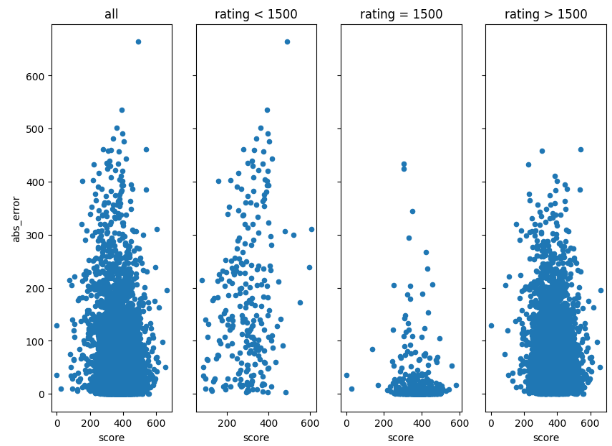
In the next graph we scatter the points (rating, error), we can see that it is slightly more likely to predict higher rating for low rating points that lower to predict lower rating for high rating points.

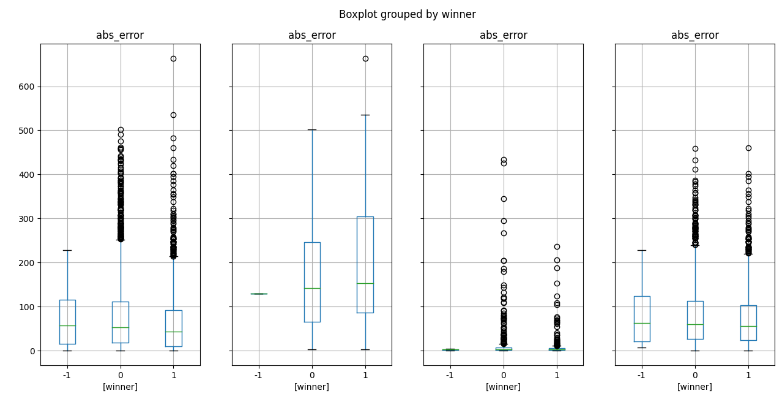


Now we will try to see how the error and absolute error behaves corresponding to different features. For each feature there are 8 graphs, 4 for absolute error and 4 for error, the firsts are on the whole data, seconds for data with rating < 1500, thirds for ratings = 1500, and forths for ratings > 1500. That's because there are many players with 1500 ratings, because that's the initial rating. The graphs look similar, so I will add one example for numerical values and one for catacorial: (in first row is the error, in the second abs_error)









We can see that when rating < 1500, prediction tends to be higher than real rating
Also, when rating = 1500 errors tend to be smaller.
When looking on data with low ratings and high error, I noticed that there was a difference between the rating and the bot rating.

Let’s see that in numbers for all the data in val set:

condition	examples meeting condition
True	5041
error > 200	195
error > 200 and (bot_rating - rating) ≤ 150	1
error > 100	636
error > 100 and (bot_rating - rating) ≤ 150	31
error > 100 and (bot_rating - rating) ≤ 200	67

So we tried to run a model without the ‘bot_rating’ feature and got **88.877 RMSE** which is worse.

Now lets see how many examples with high error are there when rating=1500 and when not:

condition	examples meeting condition
rating = 1500	753
abs_error > 100	1337
rating = 1500 and abs_error > 100	24
rating ≠ 1500 and abs_error > 100	1313

So most errors happened when rating is not 1500, and we tend to be right on examples with rating=1500.
Idea: maybe we can try to classify whether a player is new (meaning rating=1500), and if it says not, try to predict using a regressor that was fit without 1500 examples.
First, lets see what is the RMSE of the rating≠1500 data in the old model: **93.134**
In the new model, which was trained without rating=1500 data: **90.478**.
This is slightly better!

Let’s check classifier’s performence:

	validation	train
accuracy	0.991	0.998
precision	0.980	0.998
recall	0.959	9.994
F1	9.969	9.996

Looks like it works well.

Using that classifier, given the data X, we can predict as follows:
$mask \ = \ classifier(X)$
$regression \ = \ regressor(X)$
$final\_prediction \ = \ mask \cdot 1500 + (1 - mask) \cdot regression$

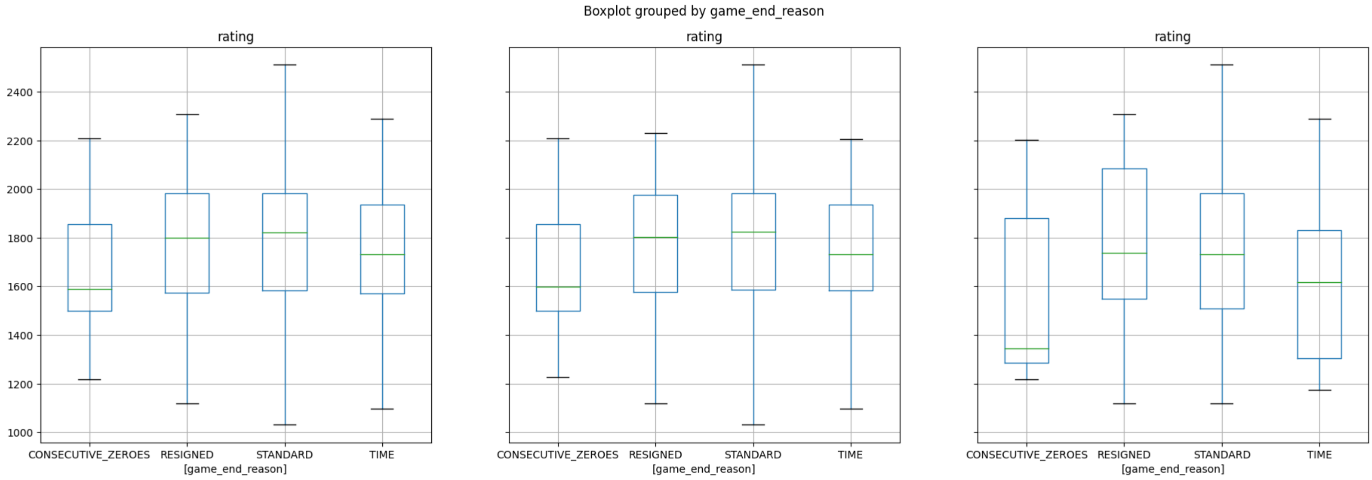
Now the RMSE on the train is **85.582** which is better!
Unfortunately, when cross validating we get:

set	RMSE score
validation	105.905
train	83.62

which is slightly worse.
Another idea was to use 2 regressors, one for examples which the clasiifier said yes on them, and were fit using all examples, and the other as before. So now prediction will be:
$mask \ = \ classifier(X)$
$regression_1 \ = \ regressor_1(X)$
$regression_2 \ = \ regressor_2(X)$
$final\_prediction \ = \ mask \cdot regression_1 + (1 - mask) \cdot regression_2$
The results are not better:

set	RMSE score
validation	105.903
train	84.466

Now let's try to draw graphs like before, only this time we will draw the rating mask splited by 3 groups: all, abs_error≤200, abs_error>200:



We can see that when ans_error is bigger, rating tends to be lower. The rest of the graphs show similar things.

To conclude, we saw that the model works well on examples with rating of 1500, and that it tends to predict higher rating for examples with low rating. Also we saw that for example with low rating and high error, bot_rating tends to be higher than rating. We have also suggested some methods to solve these problems and tested them, but the results weren't better.