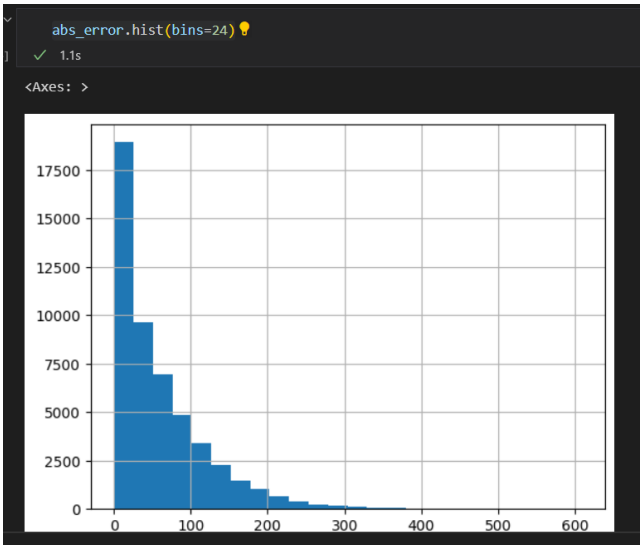
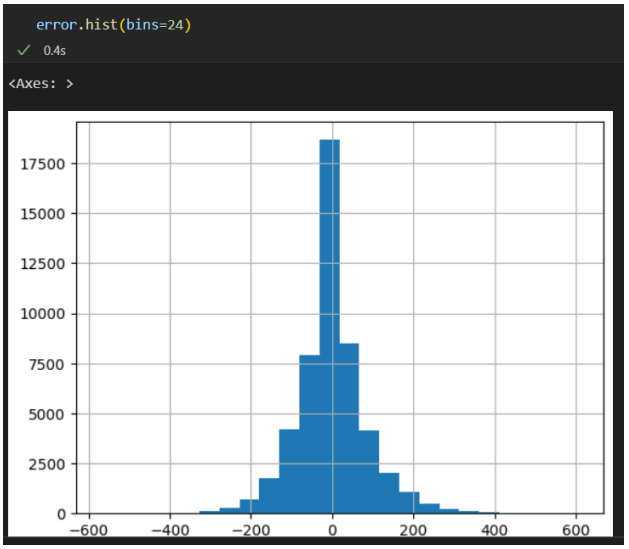
False Analysis for Scrabble Rating Prediction Project

train and validation RMSE;

```
test_score    -105.045024
train_score    -85.112928
```

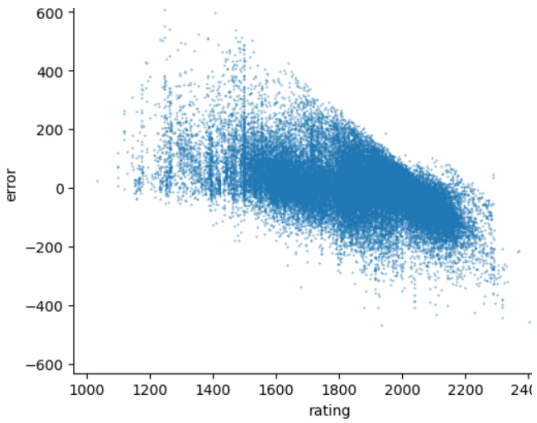
Behavior of the error and absolute error on the train set:





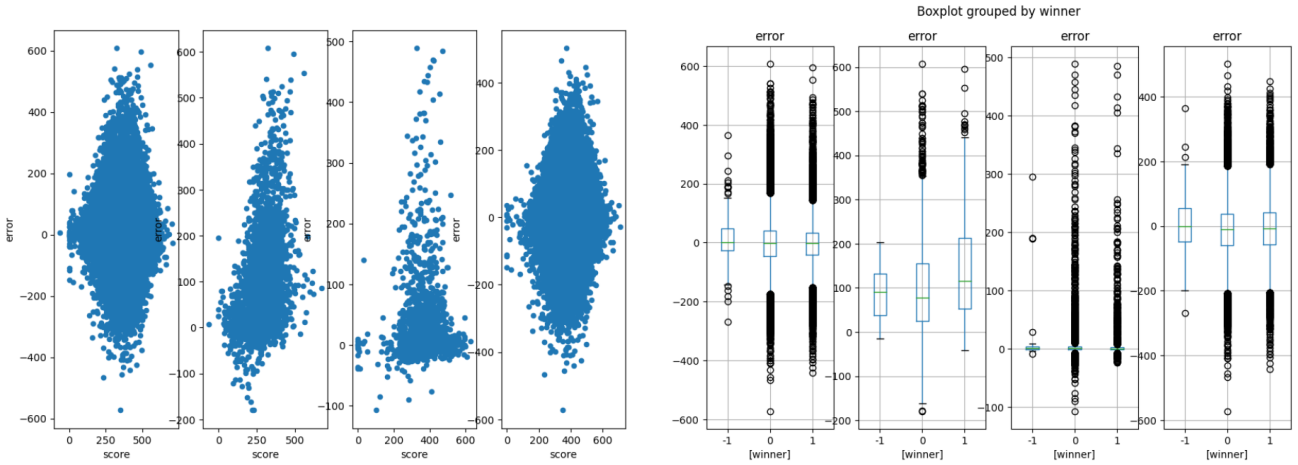
we can see they behave as expected, abs_error incidence is decreasing and error is kinda normal with mean zero.

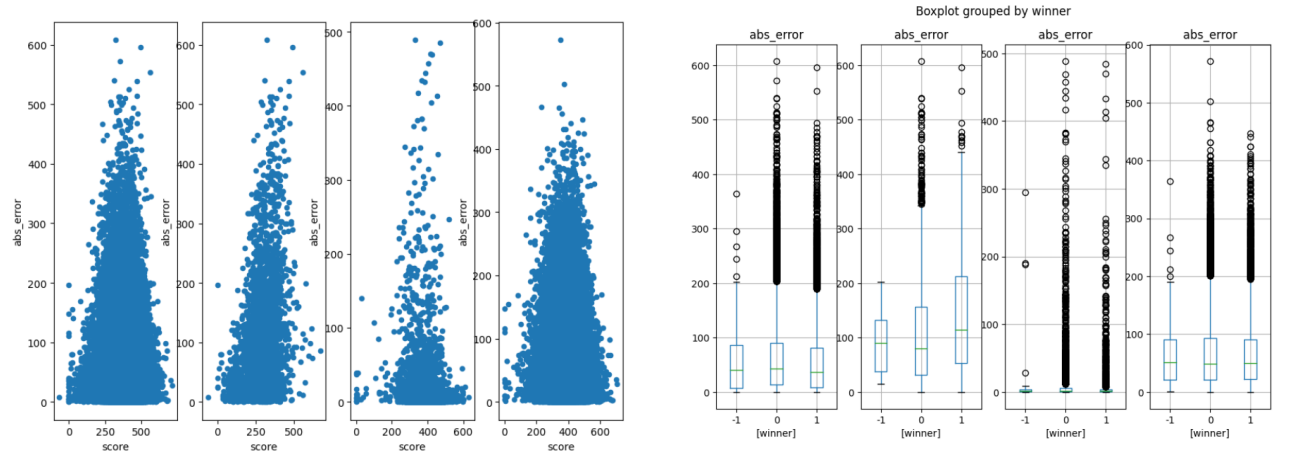
In the next graph we scatter the points (rating, error), we can see that it is slightly more likely to predict higher rating for low rating points that lower to predict lower rating for high rating points.



Now we 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:

```
total:                                          50410
prediction higher by 200:                       1192
prediction higher by 200 and bot_rating-rating<=150: 9
prediction higher by 100:                       5075
prediction higher by 100 and bot_rating-rating<=150: 288
prediction higher by 100 and bot_rating-rating<=200: 642
```

So I tried to run a model without the 'bot_rating' feature and got 88.877 which is worse.

Now lets see how much big error are there when rating=1500 and when not:

```
rating=1500:                    7575
abs_error>100:                  10210
rating=1500 & ans_error>100:    179
rating!=1500 & ans_error>100:   10031
```

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 wether a player is new (meaning rating=1500), and if it says not, try to predict using a regressor that was fitted 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 performence:

```
test_accuracy       0.991073
train_accuracy      0.998874
test_precision      0.980707
train_precision     0.998013
test_recall         0.959472
train_recall        0.994488
test_f1             0.969972
train_f1            0.996247
```

looking good,
Now, given the data X, we can predict as follows:

```
mask = classifier,predict(X)
regression = regressor.predict(X)
final_prediction = mask * 1500 + (1 - mask) * regression
```

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

```
test_score    -105.905307
train_score    -83.629168
```

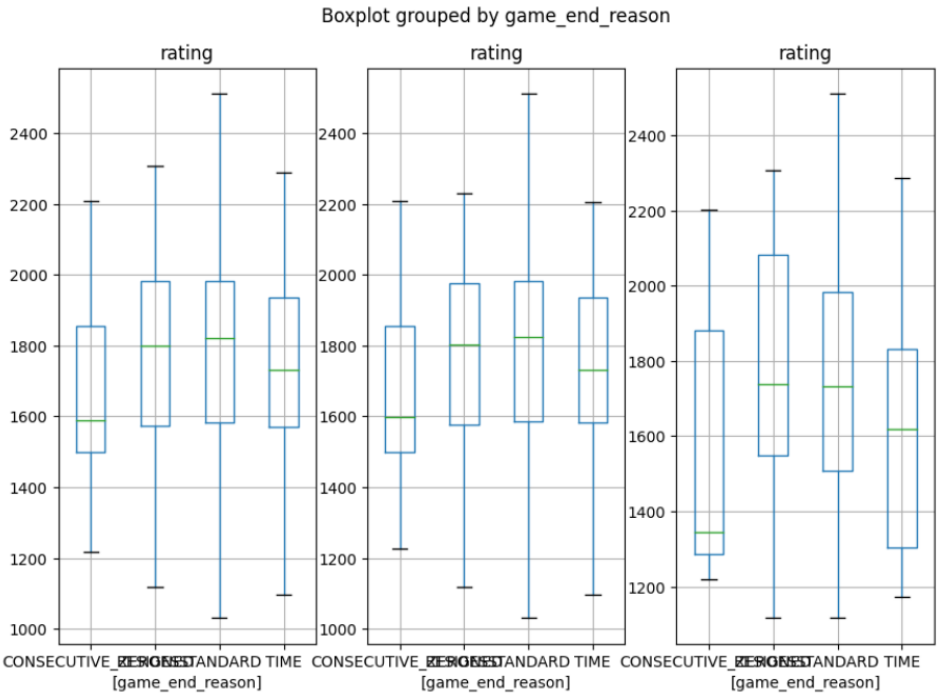
which is slightly worse.
Another idea was to use 2 regressors, one for examples which the clasiifier said yes on them, and were fit using all examples, and the other as before. So now prediction will be:

```
mask = classifier,predict(X)
regression1 = regressor1.predict(X)
regression2 = regressor2.predict(X)
final_prediction = mask * regression1 + (1 - mask) * regression2
```

The results are not better:

```
test_score    -105.903389
train_score    -84.466893
```

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. Rest of the graphs shows similar things.
Lastly I looked on those graphs again but for a validation set, results were similiar.