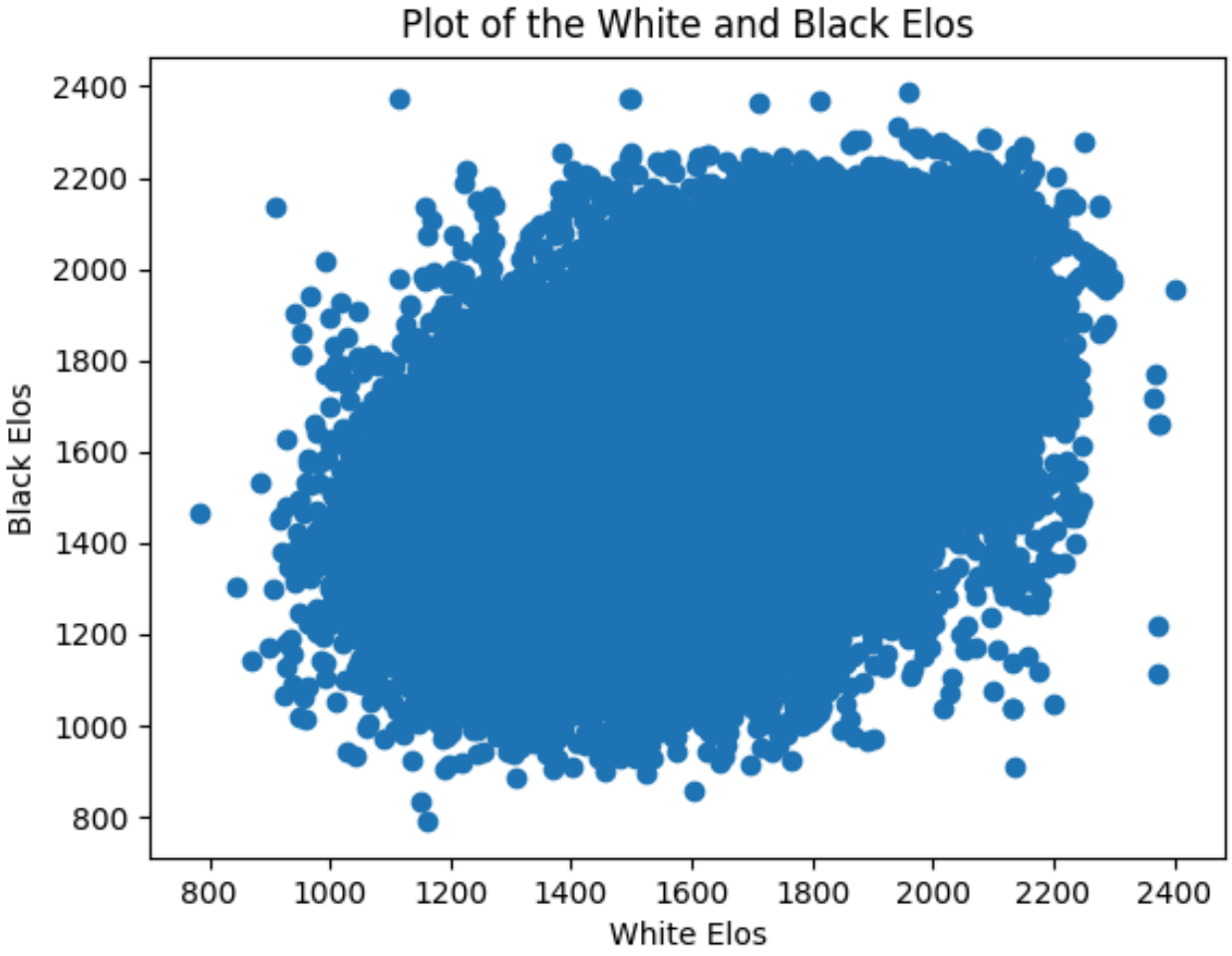
The files that are being examined for machine learning, being “Lichess 2013-01_str_list.csv” and “Lichess 2013-01_int_array.csv” are both the output of the Lichess Multiprocessing program. Sample rows of the files are for str_list and int_array, respectively:

Date	White Name	Black Name	Game Type	Time Control	Opening	Loss Type
2012.12.31	Naitero_Nagasaki	800	None	60+1	B12	Normal
2012.12.31	nichiren1967	Naitero_Nagasaki	None	60+1	C00	Normal

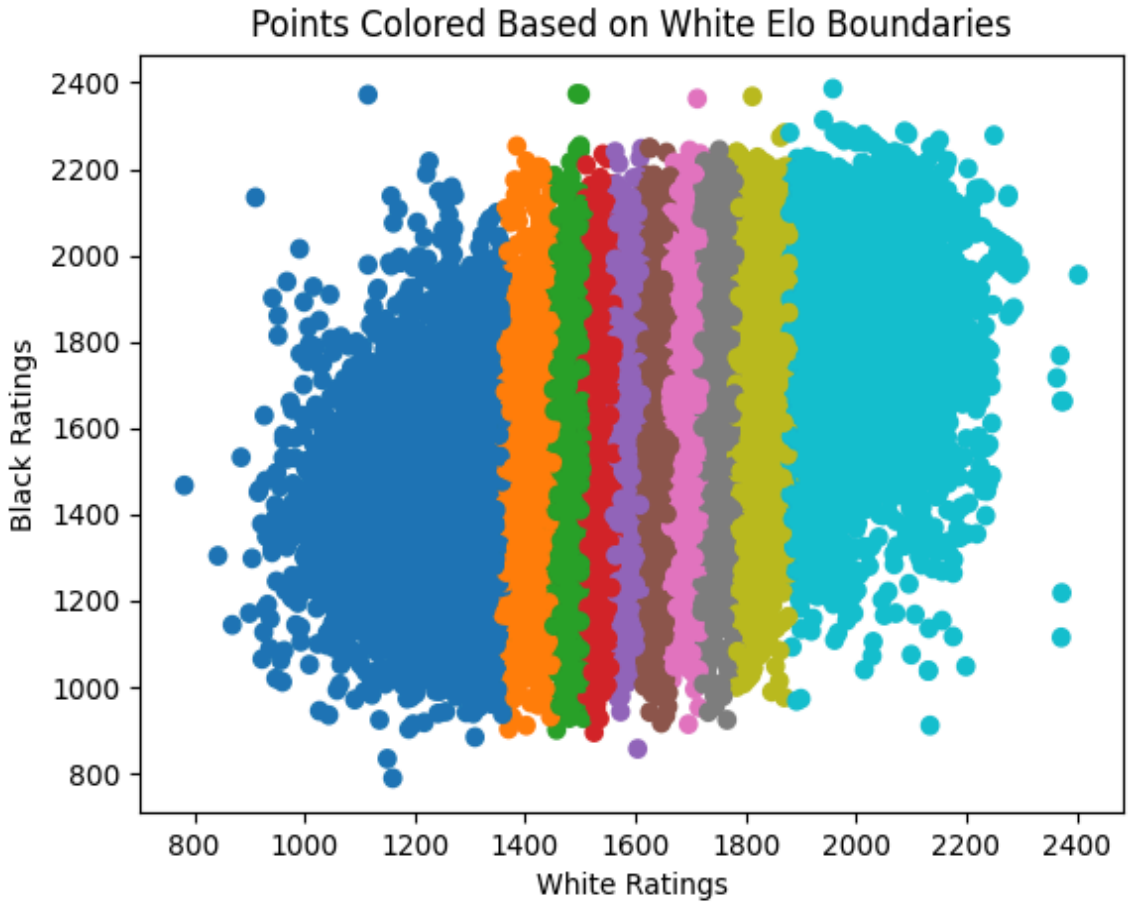
Result	White Rating	Black Rating	Delta Rating	40th FEN Eval	Sum Eval	Total Half Moves
-1	1824	1973	-149	-1	-163	94
-1	1765	1815	-50	-11	-184	46

It is important to note that the first row of str_list corresponds to the first row of int_array, and so on. Because the data is in separate files, I merged them together by row.

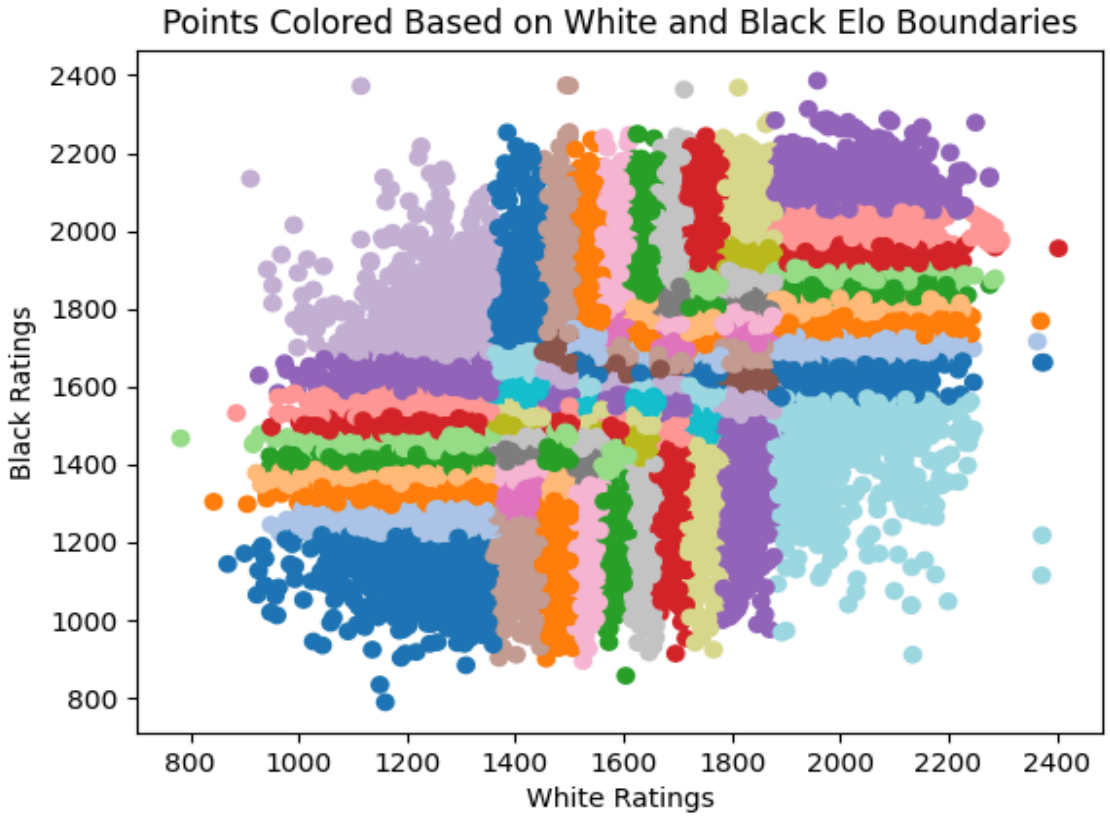
I subsequently wanted to get a view of what the data looks like, in order to gain an understanding of its distribution. Because there are many variables to examine, and given that player rating/Elo is a strong indicator of their skill, it was decided to make a plot based on the black and white ratings for each game.



To begin the process of applying machine learning to the data to predict the result of chess games, a necessary step was to first divide the data into 10 even segments based on white player rating. This is because dividing the data into different categories may help our algorithms detect underlying patterns of the data. The plot below shows these divisions. Note that some areas appear to have more data points, but this is only because of the greater variability of their white ratings.

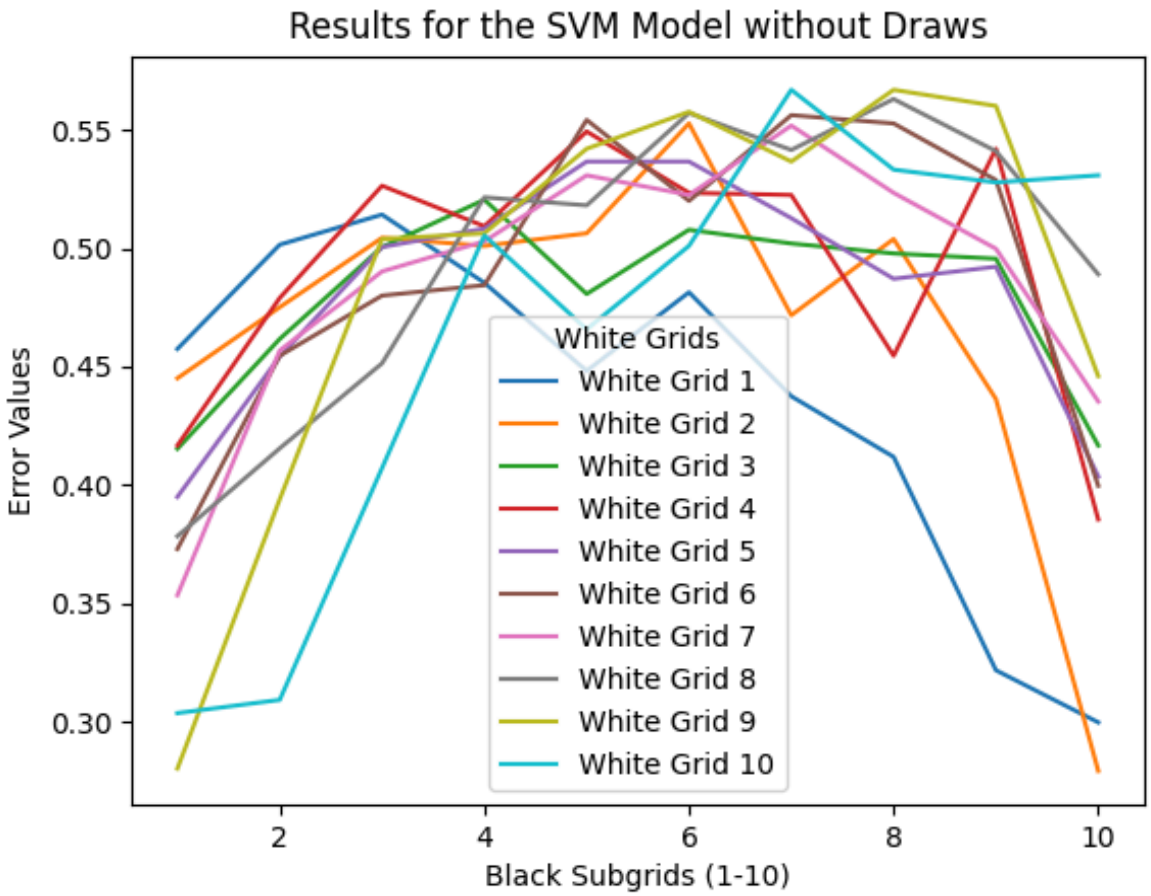


Now we repeat the process by dividing each of the 10 white segments into 10 more segments, this time by black ratings, creating 100 segments in total. The rationale for doing this is the same as before.



Now the data is ready for analysis. We will use the columns: Delta Rating, 40th FEN Eval, Sum Eval, and Total Half Moves as our predictor variables. This is because given the nature of chess and the data, these are the only columns that make sense to predict with. After running the data

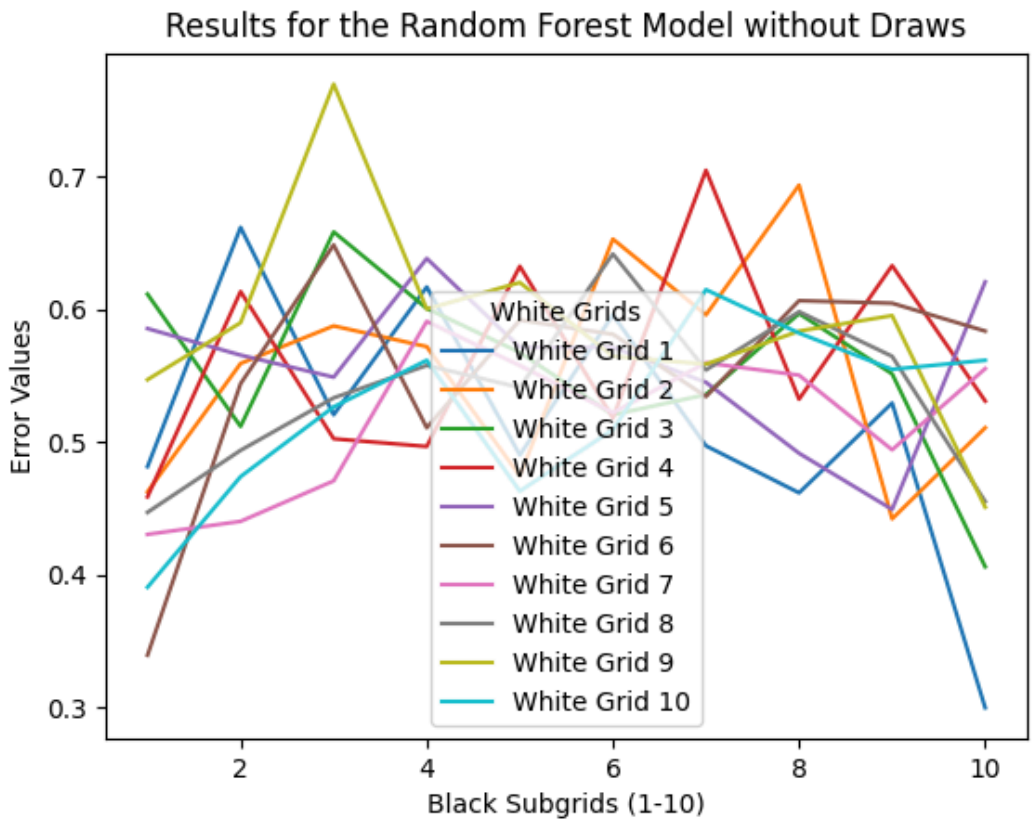
into the Random Forest and SVM algorithms, we obtained the following information:



Sample of SVM Results without Draws:

Row	Black Win Accuracy	White Win Accuracy	Cost Used	Error
15	41.7500	53.1562	10.0	0.5464
26	43.0625	53.1562	10.0	0.5225
36	49.5000	66.3125	100.0	0.5327
38	41.9375	46.2812	150.0	0.4900
41	40.6875	55.2500	50.0	0.4580
60	34.4688	55.5625	10.0	0.3677
61	49.1562	57.2188	50.0	0.4260
72	57.5625	57.1562	100.0	0.4641
81	47.8125	56.5625	1.0	0.4109
83	45.2500	55.4688	1.0	0.5161

Sample of Random Forest Results without Draws:



Row	Black Win Accuracy	White Win Accuracy	Cost Used	Error
14	42.6875	53.6875	136.0	0.4744
18	44.7500	60.8750	150.0	0.4421
26	47.7500	62.0625	139.0	0.5356
38	43.3438	53.5625	149.0	0.6328
60	50.0000	56.3125	100.0	0.4304
62	38.9688	54.1562	145.0	0.4707
73	44.4375	56.2500	140.0	0.5576
77	47.9688	62.5000	141.0	0.5981
92	47.6250	56.0000	131.0	0.5264
94	41.4375	55.1875	107.0	0.4629

It can be seen that the errors for the model results of both algorithms are high, which is due to multiple confounding variables. Lurking variables such as the short time controls, which overwhelmingly make up the majority of the dataset, and players caring less about the game outcome because it is online, both cause game outcomes to be much more random.

Future improvements would be to generate new predictor variables to improve our models, and to use a reliable dataset of in-person games, which would eliminate the lurking variables. Another possible improvement would be to increase the size of the dataset so that blocking by time controls is possible.