CHESS GAMES OUTCOMES ANALYSIS AND PREDICTIONS

Every chess game has three phases: the opening, the middlegame, and the endgame. The opening involves the first moves and relies heavily on established theory. In the middlegame, creativity comes into play, with strategy and tactics being crucial. Finally, the endgame is more technical, and the final result is determined.

Similarly, this project also has three phases. The opening phase involves discussing the project's details, goals, and the questions we aim to answer. The middlegame, like in chess, consists of two parts. Part 1 focuses on analytics and statistics, while Part 2 involves applying machine learning techniques and algorithms. Finally, the endgame covers the key findings and results.

Contents

“The Opening” 1

“The Middlegame” 5

Part_1: Tactics - Analyzing the dataset 5

Part_2: Strategy - Machine Learning Techniques and Algorithms 19

Classification 19

Regression 27

“The Endgame” 32

“The Opening”

The main goal of this project is to create machine learning algorithms that can predict the outcomes of chess games. We are also analyzing our dataset to gain insights. The games were downloaded from lichess.com.

Main Question:

Which machine learning models perform best for predicting chess game outcomes? (We will experiment with different models and compare their performance.)

Supporting Questions:

1. What is the distribution of outcomes (white wins, draws, black wins) in the dataset?
2. How do player Elo rating ranges correlate with game outcomes?
3. What are the most common openings, and how do they relate to game outcomes?
4. What is the distribution of time formats and time increment. How time increment relate to game outcomes and termination?
5. How does the time format affect game results for players with different Elo ratings?
6. How does the Elo difference vary across outcomes?
7. How does the Elo difference relate to game outcomes?
8. Hypothesis Testing
9. Which features have the strongest predictive power for game outcomes?
10. Can game outcomes be predicted more accurately for specific Elo ranges (or Elo differences) or time controls?
11. Can we improve the basic formula for Expected Outcome ?

Key Information about Chess:

- **Outcome:** A chess game can have three outcomes: a win for White, a win for Black, or a draw. A win for White is recorded as "1-0," awarding one point to the White player in tournament formats. A win for Black is recorded as "0-1," awarding one point to the Black player. A draw is recorded as "1/2 – 1/2," awarding half a point to each player.
- **Time Controls:** Chess is typically played with time controls. Chess clocks are unique because they have two timers built into a single unit, one for each player. The two timers never run simultaneously but track each player's time used. This ensures the game progresses at a desired pace, as both players are incentivized to use their time wisely for all their moves. Running out of time results in a loss. Many time formats include an increment, which is an amount of time added to the clock after each move.
- **Openings:** The first moves of a chess game are called the "opening." The opening phase usually relies on established theory. The subsequent phases are the

middlegame and the endgame. Many opening sequences have standard names, such as the Sicilian Defense and the Ruy Lopez.

- **Elo:** The Elo rating system is a method for calculating the relative skill levels of players in zero-sum games, such as chess and sports. (See: https://en.wikipedia.org/wiki/Elo_rating_system)

The difference in ratings between two players predicts the likely outcome of a match. Two players with equal ratings are expected to score an equal number of wins against each other. A player whose rating is 100 points higher than their opponent's is expected to score 64% of the points; with a 200-point difference, the expected score for the stronger player is 76%.

A player's Elo rating is a numerical value that changes based on the outcomes of rated games. After each game, the winning player gains points from the losing player. The difference in ratings between the winner and loser determines the number of points exchanged. If the higher-rated player wins, only a few points are transferred. However, if the lower-rated player wins (an upset), many points change hands. In a draw, the lower-rated player also gains a few points from the higher-rated player. This self-correcting system means that players whose ratings are too low or too high will, over time, perform better or worse than the system predicts, thus gaining or losing rating points until their ratings reflect their true playing strength.

Elo ratings are comparative and valid only within the rating pool in which they are calculated; they are not an absolute measure of a player's strength.

Rating Update Formulas:

The basic Elo rating update formula is: $R_{new} = R_{old} + K * (S - E)$

- R_{new} : The player's new rating.
- R_{old} : The player's current rating..
- K : A constant (K-factor) that determines the sensitivity of rating changes. A higher K-factor means rating changes are more significant, typically used for newer players. A lower K-factor is used for experienced players
- S: The actual score of the game (1 for a win, 0.5 for a draw, 0 for a loss).
- E: The player's expected score.

Formula for Expected Outcome: $E = \frac{1}{1+10^{(R_{opponent} - R_{player})/400}}$

This is a basic formula. There are others more complex which use exponential functions.

Example:

Let's say Player A (rating 1600) plays Player B (rating 1400) and Player A wins. With a K-factor of 50 (for simplicity):

1. Calculate Player A's expected score E_A against Player B.

$$E_A = \frac{1}{1 + 10^{(1400-1600)/400}} \approx 0.76$$

2. Calculate Player B's expected score E_B against Player A.

$$E_B = \frac{1}{1 + 10^{(1600-1400)/400}} \approx 0.24$$

3. Player A's new rating: R_{Anew} = 1600 + 50 * (1 - E_A) = 1612

4. Player B's new rating: R_{Bnew} = 1400 + 50 * (0 – E_B) = 1388

* The calculations and formulas above assume the standard FIDE (International Chess Federation) Elo rating system

Lichess uses Glicko 2 rating system which is an improvement of Elo, but is more complex mathematically.
Note that the formula for Expected Outcome does not provide information for draws. The result E_A = 0.76 means player A will win 76% of the time and lose 24% of the time against player B, but what about the draws? We can use data to calculate an empirical expected score:
E = 1 * (#wins/total number of games) + 0.5 (#draws/total number of games),
This empirical score can then be calculated for specific Elo ranges and compared with standard Expected Outcome formula.

Dataset Overview:

To create the dataset, I downloaded chess games (168070) from a selection of players across the Elo range (including, of course, Magnus Carlsen). The goal was to include players with a substantial number of games in various time formats. The dataset comprises approximately 20-25 specific players, along with their opponents (who are, naturally, a random selection of players). The game files were merged, transformed, and cleaned using Power Query, and then imported into a Jupyter

Notebook as a CSV file.
Key fields:

- **Result, WhiteElo, BlackElo:** These fields are self-explanatory
- **Time_format :**
  Bullet - Less than 2 minutes
  Blitz - 2 to 7 minutes
  Rapid - 8 to 20 minutes
  Classical - 20 minutes or more
- **Increment_binary:** The original field stored the time increment in seconds. This field was transformed into a binary Yes/No format
- **Termination:**
  Normal: checkmate, stalemate, resignation etc
  Time_forfeit: Loss due to time running out
  Abandoned : A small number of games where players left the game (likely to internet connection issues)
- **Opening_name:** The are 500 ECO codes and numerous named variations and subvariations. These were consolidated into 50 key openings using a logical classification system.
- **Custom columns:**
  Elo_Range: the bins were created using WhiteElo
  EloDif : WhiteElo minus BlackElo
  Elo_Dif_Range: "Crushing Black Advantage", "Strong Black Advantage", ..., "Balanced / Even", ..., "Crushing White Advantage" using the "EloDif" field
  Score: 1 for '1-0', 0.5 for draw, 0 for '0-1'

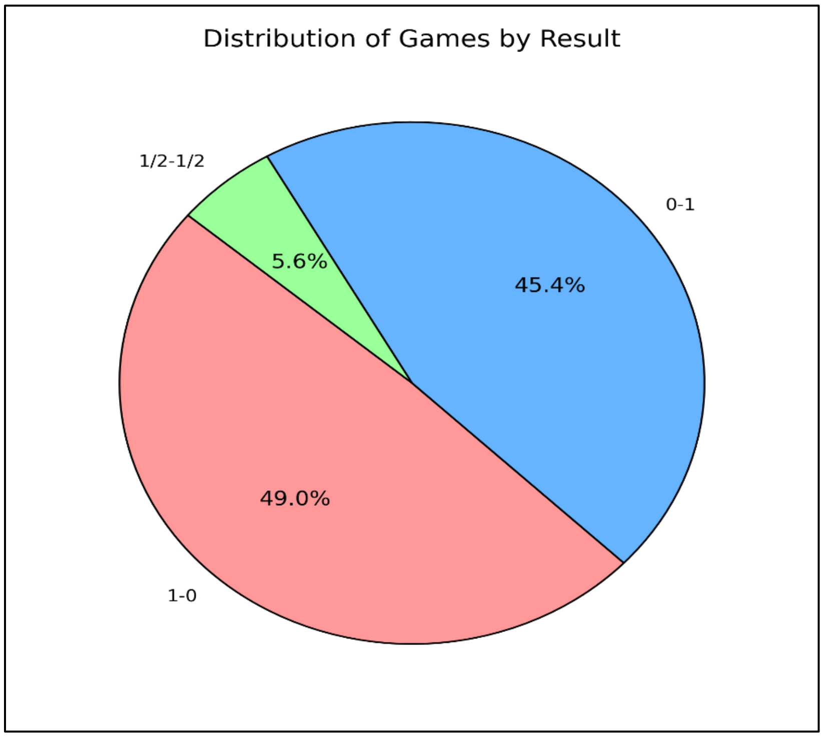
This concludes the "Opening" phase. We will now proceed to the next phase of the project.

"The Middlegame"

Part_1: Tactics - Analyzing the dataset

In Part 1 of the middlegame, we explore the data to answer the supporting questions and gain valuable insights. The analysis was conducted using Python and Jupyter Notebook. The most important graphs and results are presented here, while the complete code and analysis are available in a separate Jupyter Notebooks.

1. **What is the distribution of outcomes (white wins, draws, black wins) in the dataset?**



Overall, White wins 49% of the time, compared to 45.4% for Black wins, with draws occurring 5.6% of the time.

2. **How do player Elo rating ranges correlate with game outcomes?**

```
 Percentage of Game Results Per Elo Range:

Result        1-0    0-1  1/2-1/2
Elo_Range
<1000       47.70  46.05     6.26
1000-1300   50.74  47.34     1.92
1300-1600   48.90  48.43     2.67
1600-1800   48.31  48.73     2.96
```

```
1800-2000  47.94  47.59     4.47
2000-2200  49.04  45.37     5.59
2200-2400  48.24  44.63     7.13
2400-2600  48.82  43.18     8.01
2600+      51.14  40.84     8.01
```

Important note: The Elo ranges used here are based on the white player's rating.

Intersting insights: Across all Elo ranges below 2000, the win ratios for White and Black are very close, with a slight edge to White. This suggests that the first-move advantage is not particularly significant at these levels. Interestingly, in the 1600-1800 range, Black actually scores slightly *higher* than White.

Above 2000 Elo, playing White confers a significant advantage. The win ratio for Black decreases, and the number of draws increases. The higher frequency of draws at higher Elo levels is expected, as stronger players make fewer blunders and generally play more accurately.

However, we observed a very interesting and unexpected phenomenon: players below 1000 Elo also have a surprisingly high draw rate, around 6.26%!

Possible explanation: Regarding the high draw rate in the under-1000 Elo category, a plausible explanation is that these players may lack the skill to deliver checkmate, even in advantageous positions. This could lead to unintentional stalemates, threefold repetitions, or perpetual checks, all of which result in draws.

For the 1600-1800 Elo range, where Black seems to perform slightly better, a possible explanation could be the way Lichess assigns initial ratings. New players on Lichess are assigned a default rating of 1500. This means that many players in the 1600-1800 range might not have reached their true playing strength yet. Their ratings are still in flux. This 'rating inflation' could be affecting the win/loss ratios in this range, and potentially even neighboring ranges.

3. **What are the most common openings, and how do they relate to game outcomes?**

   Top 16

```
Percentage of Games by Opening Name:

Sicilian defense                                   17.96
Queen's Pawn Game                                  10.92
French Defense                                      6.80
```

```
English Opening                                    6.40
Caro-Kann defense                                  4.77
Irregular Openings                                 4.56
Queen's Gambit                                     4.24
Scandinavian Defense (Center-Counter Defense)      3.78
Closed Game, Irregular Responses                   3.26
Zukertort Opening                                  3.02
King's Pawn Opening, Irregular Defenses            2.65
King's Gambit                                      2.49
Petrov's Defense                                   1.99
Alekhine's defense                                 1.92
King's Pawn Game                                   1.83
Ruy Lopez                                          1.79
```

'Sicilian Defense' tops the list at 17.96%. This major opening, with its countless variations, is a popular choice against White's first move, 1.e4.

'Queen's Pawn Game' comes in second at 10.92%. This opening category encompasses many other openings and variations following White's initial 1.d4.

The 'English Opening' ranks fourth with 6.40%, characterizing games that begin with 1.c4.

The 'French Defense' and 'Caro-Kann Defense' occupy the third and fifth spots, respectively. These are both very popular responses to 1.e4.

Some interesting findings: We see a high percentage of 'Irregular Openings,' Closed Games (1.d4) / King's Pawn (1.e4) Irregular Defenses'. This suggests that many players deviate from established theory right from the start. Also, the 'Scandinavian Defense (Center Counter Defense)' makes it into the top 16, which is noteworthy. Perhaps most surprising is that the 'Ruy Lopez' is at the bottom of our list, at just 1.79%. This is a very strong opening choice for White against 1.e4 e5 in over-the-board tournaments, often favored by World Champions. However, our data shows it's not as popular on Lichess.com, where players seem to prefer more "romantic" openings like the 'King's Gambit,' which has a 2.49% frequency.

```
Win/Loss Distribution for Top Openings:

Result                                             1-0    0-1  1/2-1/2
Opening_name
Alekhine's defense                               48.25  43.60     8.14
Caro-Kann defense                                49.61  44.32     6.07
Closed Game, Irregular Responses                 51.13  44.01     4.85
English Opening                                  49.45  44.40     6.15
French Defense                                   48.88  46.08     5.04
Irregular Openings                               49.12  46.26     4.62
King's Gambit                                    48.58  46.45     4.97
King's Pawn Game                                 50.90  44.38     4.73
King's Pawn Opening, Irregular Defenses          50.02  44.60     5.38
```

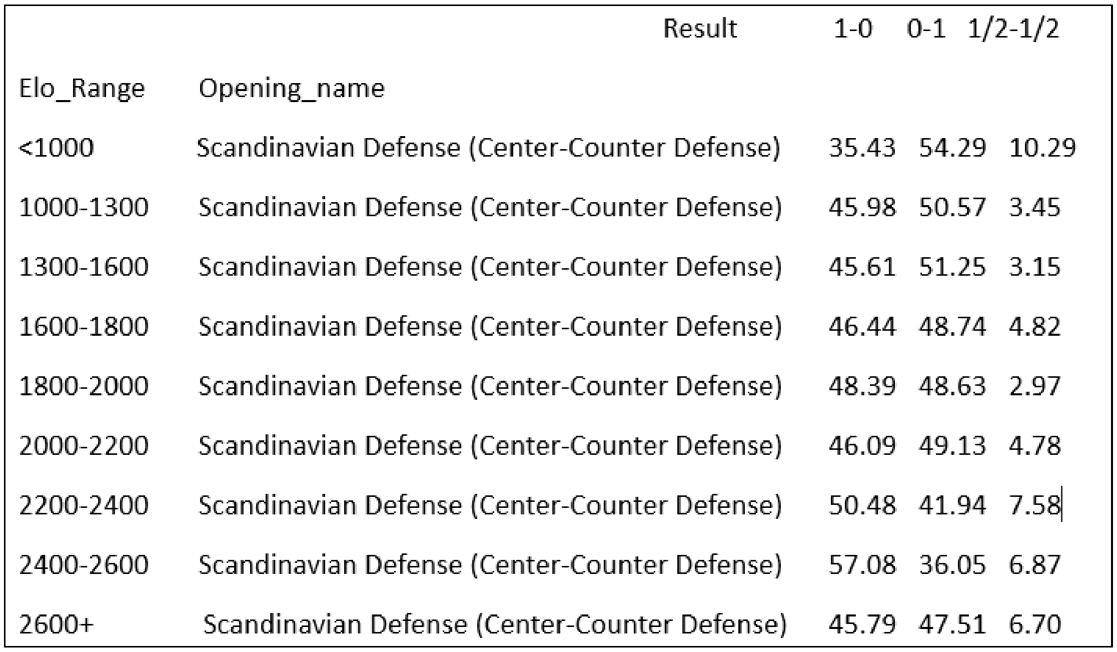
Petrov's Defense	49.03	47.38	3.60
Queen's Gambit	47.91	45.64	6.45
Queen's Pawn Game	49.29	45.65	5.06
Ruy Lopez	50.20	43.15	6.65
Scandinavian Defense (Center-Counter Defense)	46.87	48.21	4.93
Sicilian defense	48.39	45.62	5.99
Zukertort Opening	48.34	44.99	6.66

The best perfoming opening for white is ‘Closed Game, Irregular Responses’. This means white player opens the game with 1.d4 and his opponents sooner or later plays anorthodox leading to white advantage.

Similarly, the next best openings for white are ‘King’s Pawn Game’ and ‘King's Pawn Opening, Irregular Defenses’ meaning white opens 1.e4 and player with black deviates for established theory.

Of course ‘Ruy Lopez’ reaches top performing list for white with 50.2% win ratio and the lowest of the top16 openings win ratio for black with 43.15%.

These percentages were no surprise, but what takes our attention is ‘Scandinavian Defense (Center-Counter Defense)’. This is the only oppening of the top16 that black outperfoms white with 48.21% for black and 46.87% for white. “Scandi”, short name of the opening in question, is a black’s response to 1.e4. Most of the players play the very famous and double edge ‘Sicilian’ as the response to 1.e4 and established theory doesn’t recoment “Scandi” which is suppose to give white player an advantage(if white knows what he/she is doing). The fact that Black is winning more often than White with this opening goes against conventional chess wisdom. We *absolutely* need to understand why this is happening. Lets look at win/loss ration per elo range.



	Result	1-0	0-1	1/2-1/2
Elo_Range	Opening_name			
<1000	Scandinavian Defense (Center-Counter Defense)	35.43	54.29	10.29
1000-1300	Scandinavian Defense (Center-Counter Defense)	45.98	50.57	3.45
1300-1600	Scandinavian Defense (Center-Counter Defense)	45.61	51.25	3.15
1600-1800	Scandinavian Defense (Center-Counter Defense)	46.44	48.74	4.82
1800-2000	Scandinavian Defense (Center-Counter Defense)	48.39	48.63	2.97
2000-2200	Scandinavian Defense (Center-Counter Defense)	46.09	49.13	4.78
2200-2400	Scandinavian Defense (Center-Counter Defense)	50.48	41.94	7.58
2400-2600	Scandinavian Defense (Center-Counter Defense)	57.08	36.05	6.87
2600+	Scandinavian Defense (Center-Counter Defense)	45.79	47.51	6.70

Our Jupyter Notebook analysis reveals a fascinating trend. For Elo ranges from 0 to 2000, Black players using the Scandinavian Defense against 1.e4 have a *higher* win ratio than White. Even in the 2000-2200 range, Black maintains an edge, with a 49.13% win ratio compared to 46.09% for White.

However, the picture changes above 2200 Elo. In the 2200-2400 range, White takes the lead with 50.48% against Black's 41.94%. The disparity becomes even more pronounced in the 2400-2600 range, where White's win ratio skyrockets to 57.08% (the highest we've seen so far), while Black's plummets to a mere 36.05%.

Interestingly, at the highest Elo range (2600+), the Scandinavian Defense seems to revert to its earlier trend. Black players again have a higher win ratio, at 47.51%, compared to White's 45.79%.

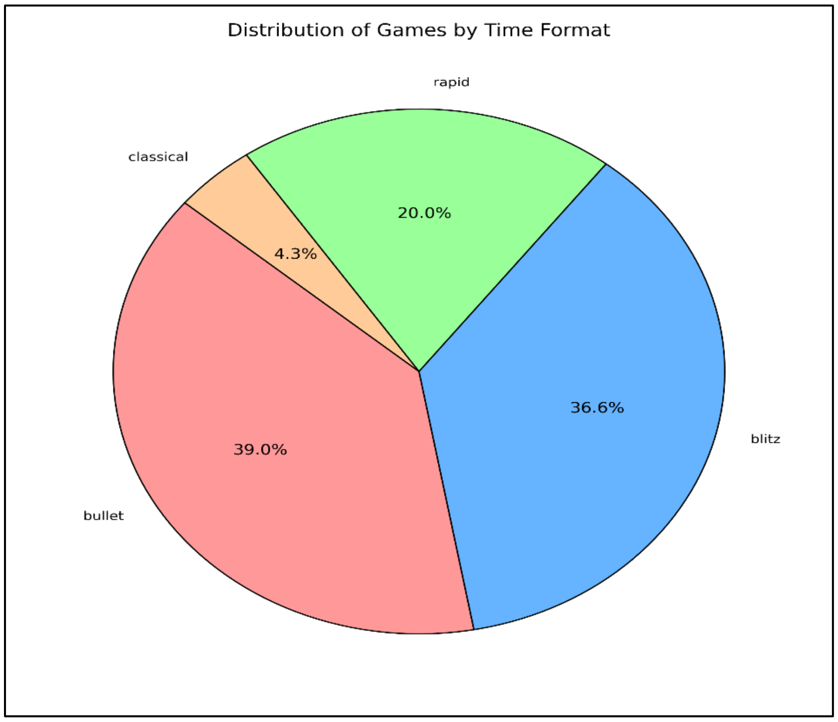
Possible explanation: The general goal of the Scandinavian Defense is to prevent White from controlling the center of the board with pawns, effectively leading to an open game. Black aims to build a strong pawn structure, gaining activity and piece play to create pressure. White, in turn, gains tempi and develops a solid position. Theoretically, White should be able to withstand Black's pressure and leverage their advantages. Perhaps White players at lower levels aren't effectively exploiting the theoretical advantages of the opening. While the pressure in their possition from black's piece play leads to blunders and the loss of the game.

The shift at 2200-2400 and especially the *huge* swing at 2400-2600 is very significant. This suggests that at these higher levels, White players *are* effectively leveraging the

Scandinavian's theoretical weaknesses. They are playing more accurately and punishing Black's choices.

The *reversal* at 2600+ is perhaps the most interesting finding of all. Why does the Scandinavian suddenly become more effective for Black again at the very highest level? This is a puzzle. Perhaps top-level players have developed new and innovative ways to play the Scandinavian, neutralizing White's advantages. Or perhaps they are using it as a surprise weapon, catching White off guard.

4. **What's the breakdown of time formats and time increments? And how do these time increments affect game results and how the game ends (termination)**



It looks like players in your dataset really like the fast-paced games! A lot of them choose "bullet" (39%) and "blitz" (36.6%) time formats. It's interesting that those quicker formats are so popular compared to the slower ones like "rapid" (20%) and "classical" (4.3%) which people often think of as more serious.

```
Percentage of Increment per Time_format:

Increment_binary      No    Yes
Time_format
 blitz             58.55  41.45
 bullet            83.60  16.40
 classical         72.18  27.82
 rapid             81.19  18.81
```

It seems like most players really prefer playing without any time increment, especially in bullet games where a huge 83.6% of games are played that way. Blitz is the only format where a decent number of players (41.45%) actually use a time increment.

```
Percentage of Termination per Increment_binary:

Termination       Abandoned  Normal  Time forfeit
Increment_binary
No                     0.02   71.54         28.44
Yes                    0.02   80.68         19.30
```

That's interesting! A pretty big chunk of players (28.44%) lose because they run out of time in games *without* a time increment. It's still a significant number (19.30%) who lose due to time in games *with* an increment, but it's definitely less.

```
Percentage of Game Results Per Time format:

Result          1-0    0-1  1/2-1/2
Time_format
 blitz        48.41  44.92     6.66
 bullet       49.51  45.92     4.57
 classical    48.71  44.47     6.82
 rapid        49.15  45.29     5.56
```

It looks like the time format doesn't really change the overall win/loss ratios. White still wins about 4% more often than Black, regardless of how fast or slow the game is.

5. **How does the time format affect game results for players with different Elo ratings?**

   We were looking for something interesting, and we definitely found it! It's pretty strange that in blitz games, Black actually wins *more* often than White in the lower Elo ranges (from 1300 up to 2000)! That's unexpected. Also, in bullet games, the win ratios are pretty close to even up to 2000 Elo. This suggests that the advantage of going first as White isn't as big of a deal in those fast time formats and lower Elo ranges.

Once players get above 2000 Elo, things start to look more typical, with White winning more often across all the time formats. However, we did find one exception: in rapid games between players with Elo ratings from 2200 to 2400, Black actually won more often (47.19%) than White (43.49%). It's possible that this is just a random fluke due to the relatively small number of rapid games in that specific Elo range.

6. **How does the Elo difference vary across outcomes?**

```
Elo Difference Mean by Game Outcome:
Result
0-1       -21.678799
1-0        20.445631
1/2-1/2    -2.634087
Name: EloDif, dtype: float64
```

We calculated the average Elo difference for different game results. It turns out that when White wins, the average Elo difference is 20.44 points. When Black wins, the average difference is -21.67 points (meaning Black is, on average, rated higher). And in drawn games, Black's Elo is, on average, 2.63 points higher than White's.

7. **How does the Elo difference relate to game outcomes?**

```
Percentage of Game Results Per Elo Difference Range:

Result                      1-0    0-1  1/2-1/2
Elo_Dif_Range
Crushing Black Advantage  16.11  80.81     3.09
Strong Black Advantage    25.20  68.15     6.65
Moderate Black Advantage  35.11  58.07     6.82
Slight Black Advantage    41.42  52.56     6.02
Minimal Black Edge        45.69  48.97     5.34
Balanced / Even           49.29  45.65     5.06
Minimal White Edge        52.54  42.07     5.39
```

```
Slight White Advantage    56.57  37.61     5.82
Moderate White Advantage  62.37  31.51     6.12
Strong White Advantage    73.06  21.63     5.31
Crushing White Advantage  80.24  15.09     4.67
```

The results aren't surprising – they basically confirm how the Elo system works. Players with higher ratings tend to win more often. We defined a "Crushing Advantage" as a difference of 400+ Elo points. The standard FIDE Elo system predicts a 90% win rate for the higher-rated player in such a matchup. However, experts say that Elo ratings on Lichess.com tend to be higher than FIDE ratings, so we shouldn't try to directly compare them.

Let's see in which time formats and openings the player with a "Crushing" Elo advantage (400+ points) actually loses.

<u>White +400</u>

```
Percentage of Crushing White Advantage losses per Time Format:

 blitz         59.29
 bullet        20.35
 classical     10.62
 rapid          9.73
Name: Time_format, dtype: float64

Percentage of Crushing White Advantage losses per Opening:

Sicilian defense                                 19.47
English Opening                                  11.50
King's Gambit                                     7.96
Queen's Pawn Game                                 7.08
French Defense                                    6.19
Caro-Kann defense                                 6.19
Ruy Lopez                                         3.54
Scandinavian Defense (Center-Counter Defense)     2.65
```

Similarly, continuing with black.

Black +400

```
Percentage of Crushing Black Advantage losses per Time Format:

 blitz         51.67
 bullet        25.00
 classical     12.50
 rapid         10.83
Name: Time_format, dtype: float64

Percentage of Crushing Black Advantage losses per Opening:

Sicilian defense                             27.50
Queen's Pawn Game                            11.67
Caro-Kann defense                             8.33
Queen's Gambit                                5.83
English Opening                               5.83
French Defense                                5.83
Irregular Openings                            5.00
Larsen's Opening                              4.17
Zukertort Opening                             2.50
```

If you ever find yourself playing White against a much stronger opponent (like, 400+ Elo points higher), and it's a blitz game, try asking them to play the Sicilian Defense against you. Tell them you just want to test something out .Trust me, if they're that much stronger, they probably won't mind, and it could give you a chance of beating them or at least an interesting game!

Wait, wait! This is not actually right. It is a common mistake to miss interpet these kind of results. Observing that 59.29 % of losses by White with a crushing advantage occur in Blitz games does not, on its own, demonstrate a genuine pattern. If Blitz games simply make up a larger share of our dataset, we would expect most losses to occur there by default. To distinguish mere base-rate effects from real differences, we must turn to statistical inference: explicitly state a null hypothesis and carry out χ^2 tests to assess whether loss rates truly vary by time control and opening selection.

8. Hypothesis Testing

To examine whether players under a “Crushing Advantage” (≥ 400 Elo points) are more likely to blunder into loss depending on time control or opening, we conducted χ^2 tests on five contingency tables. Below is a concise summary of each test and its outcome.

1. Effect of Time Format under a Crushing White Advantage

We first tested whether, when White enjoys a crushing rating edge, the probability of a loss (i.e. 0–1 result) depends on the time format (Bullet, Blitz, Rapid, Classical).

Time Format	Losses	Non-Losses
Blitz	67	222
Bullet	23	135
Classical	12	47
Rapid	11	232

The 4 × 2 χ^2 test yields $\chi^2(3) = 37.25$, $p < 0.0001$, indicating a highly significant association between time format and loss rate .

2. Blitz vs. Other Formats under a Crushing White Advantage
To isolate the notable spike in Blitz losses, we collapsed formats into "Blitz" and "Other." Under this binary split, players still lose **23.2 %** of Blitz games versus **10.0 %** of all other games. The 2 × 2 χ^2 test confirms a highly significant difference: $\chi^2(1) = 23.06$, $p < 0.0001$.

Format	Losses	Non-Losses
Blitz	67	222
Other	46	414

So we can reject the null hypothesis that the loss probability is the same in Blitz and "Other" formats.

3. Sicilian Defense vs. Other Openings under a Crushing White Advantage
Next, we examined whether the Sicilian Defense yields a different loss rate than all other openings. Here, 17.3 % of Sicilian-Defense games end in a loss (22/127) versus 14.6 % elsewhere. The 2 × 2 χ^2 test gives: $\chi^2(1) = 0.41$, $p = 0.52$,

Opening Group	Losses	Non-Losses
Sicilian Defense	22	105

Opening Group	Losses	Non-Losses
Other Openings	91	531

so we cannot reject the null hypothesis of equal loss-rates.

4. Effect of Time Format under a Crushing Black Advantage

Mirroring the White-advantage tests, we find that when Black holds a crushing edge, loss-rates also vary by speed. In particular, Black still blunders 22.0 % of Blitz games versus 12.5 % across other formats:

Format	Losses	Non-Losses
Blitz	62	220
Other	58	405

Highly significant association. With p = 0.0010 (< 0.05), we can reject the null of equal loss-probabilities: under a crushing Black advantage, the format (Blitz vs Other) significantly affects the chance of Black blundering into a loss.

5. Sicilian Defense vs. Other Openings under a Crushing Black Advantage
Finally, comparing Sicilian Defense to all other openings when Black is crushingly ahead shows loss-rates of 18.6 % versus 15.3 %, respectively.

Opening Group	Losses	Non-Losses
Sicilian Defense	33	144
Other Openings	87	481

No significant association. With p = 0.3501 (> 0.05), we cannot reject the null hypothesis of equal loss-probabilities. In plain terms, Sicilian Defense does not show a statistically different loss-rate compared to other openings when Black is crushingly better.

So, we can see both these results as good news. We can still "trap" better players in blitz, where we've found an edge. Plus, we're not necessarily stuck with "the Sicilian Defense" – we can pick our favourite opening since there's no real difference. Maybe playing the "Scandinavian" as Black is worth exploring, as we have some interesting results there too, but that needs more testing. And I'm keeping a few secrets close to my chest for now!

Part_2: Strategy - Machine Learning Techniques and Algorithms

Classification

In this section, we apply and evaluate classification methods to predict game outcomes. First, we preprocess the data by dropping the redundant **BlackElo** column and recoding **Score** into a three-class target: **White Win**, **Black Win**, and **Draw**. We then limit **Opening_name** to the ten most frequent openings, grouping all others under **Other**. To ensure reproducibility, we split the dataset into fixed training and test sets (using a constant random seed).

As a performance benchmark, we trained two dummy classifiers:

- Most Frequent, which always predicts the majority class, achieving Accuracy = 0.491
- Stratified, which predicts according to the class distribution, achieving Accuracy = 0.455

Our aim is to build models that substantially exceed these baselines—ideally pushing accuracy well above 0.49.

1. **<u>Linear Models: LogisticRegression, SGDClassifier, LinearSVC</u>**

   We loaded the dataset and separated features and target variables. To prepare the data for modeling, we built a transformation pipeline. This pipeline included a numerical_transformer for normalizing numerical features and a categorical_transformer for one-hot encoding categorical features, and it was applied to the feature sets. This data preparation was consistently performed in all model training notebooks. For

hyperparameter tuning, we used Grid Search or Random Search for each linear model. While full details and classification reports are provided in individual Jupyter notebooks for each model, the test accuracies are as follows:
Logistic Regression: test accuracy 0.537
SGDClassifier: test accuracy 0.54
LinearSVC : test accuracy 0.54

2. **Support Vector Machine with Polynomial and RBF Kernel**

Continuing with the Support Vector Machine (SVM) models, we obtained the following test accuracies:
RBF Kernel: test accuracy 0.542
Polynomial Kernel: test accuracy 0.534

3. **DecisionTreeClassifier and PCA**

DecisionTreeClassifier: test set accuracy 0.527

We then performed PCA, reducing the feature count from 19 to 8 (retaining 90% variance). Combining this reduced dataset with the Decision Tree yielded the following results after hyperparameter tuning:

Best parameters (PCA + Tree): {'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 2}
Best CV accuracy (PCA + Tree): 0.536
Test accuracy(PCA + Tree): 0.532

Applying PCA resulted in a slight improvement in the Decision Tree's performance.

4. **Random Forest and Feature Importance**

RandomForestClassifier: test set accuracy 0.548
This is the best accuracy we got so far!
This represents the highest accuracy achieved thus far. To understand the factors contributing most to the model's predictions, we utilized the .feature_importances_ attribute, revealing the following feature importance scores:

```
0.29: WhiteElo
0.61: EloDif
```

```
0.0: Opening_name_Caro-Kann defense
0.0: Opening_name_Closed Game, Irregular Responses
0.0: Opening_name_English Opening
0.0: Opening_name_French Defense
0.0: Opening_name_Irregular Openings
0.01: Opening_name_Other
0.0: Opening_name_Queen's Gambit
0.01: Opening_name_Queen's Pawn Game
0.0: Opening_name_Scandinavian Defense (Center-Counter Defense)
0.0: Opening_name_Sicilian defense
0.0: Opening_name_Zukertort Opening
0.01: Time_format_ blitz
0.01: Time_format_ bullet
0.01: Time_format_ classical
0.01: Time_format_ rapid
0.01: Increment_binary_No
0.01: Increment_binary_Yes
```

As the feature importance scores indicate, the Elo-related columns contain the most predictive information:
0.61: EloDif (Elo Difference): This is by far the most important feature according to the model.
0.29: WhiteElo (White's Elo): White's individual Elo rating also has a significant impact, although less than the Elo difference. This suggests that even when the Elo difference is the same, the absolute strength of White can still influence the outcome.
The rest features have a very small, almost negligible importance score.

5. Advanced Methods: AdaBoostClassifier, BaggingClassifier, GradientBoostingClassifier

We used also some advanced algorithms without much of imporvement on accuracy.
AdaBoostClassifier: set accuracy 0.548
BaggingClassifier: set accuracy 0.517
GradientBoostingClassifier: set accuracy 0.543

6. Voting using Various Models

We constructed a VotingClassifier incorporating our previously trained models and evaluated both hard and soft voting strategies. The results were:
Voting Classifier Accuracy: 0.541
Soft Voting Classifier Accuracy: 0.540

The fact that the Voting Classifier cannot outperform the best individual models is a strong indication that the models in the ensemble are not sufficiently diverse and are likely making similar types of errors.

7. **Exploring different feature ranges to assess their impact on our models' performance.**

   We explored the performances of our models in various elo ranges, elo differences and time formats and here are the results:

   a) Elo Ranges

```
=== Low (<1000) group ===
n_samples: 167
AdaBoost              Accuracy: 0.581
Random_forest         Accuracy: 0.545
Gradient Boosting     Accuracy: 0.557
Decision Tree         Accuracy: 0.497
SVC (poly)            Accuracy: 0.479
SVC (rbf)             Accuracy: 0.551
voting_clf            Accuracy: 0.545

=== Mid (1000–1999) group ===
n_samples: 3583
AdaBoost              Accuracy: 0.530
Random_forest         Accuracy: 0.541
Gradient Boosting     Accuracy: 0.524
Decision Tree         Accuracy: 0.507
SVC (poly)            Accuracy: 0.519
SVC (rbf)             Accuracy: 0.529
voting_clf            Accuracy: 0.523

=== High (>=2000) group ===
n_samples: 6250
AdaBoost              Accuracy: 0.556
Random_forest         Accuracy: 0.552
Gradient Boosting     Accuracy: 0.554
Decision Tree         Accuracy: 0.539
SVC (poly)            Accuracy: 0.544
SVC (rbf)                 Accuracy: 0.549
voting_clf                Accuracy: 0.552
```

Our analysis of different Elo rating groups reveals the best model performance within the under 1000 Elo group. However, with a sample size of only 167, this finding might not be statistically robust. Among the other two groups (1000-1999 and >= 2000 Elo), the group with an Elo rating of 2000 or higher shows approximately 2% better prediction accuracy compared to the 1000-1999 Elo group.
Furthermore, we noted that AdaBoost outperformed Random Forest in both the Low (<1000) and High (>=2000) Elo groups.

b) Elo difference

```
=== Elo_Diff_Low (<100) group ===
n_samples: 7654
 AdaBoost                 Accuracy: 0.525
 Random_forest            Accuracy: 0.526
 Gradient Boosting        Accuracy: 0.521
 Decision Tree            Accuracy: 0.499
 SVC (poly)               Accuracy: 0.510
 SVC (rbf)                Accuracy: 0.519
 voting_clf               Accuracy: 0.518

=== Elo_Diff_Mid (100–200) group ===
n_samples: 1800
 AdaBoost                 Accuracy: 0.590
 Random_forest            Accuracy: 0.590
 Gradient Boosting        Accuracy: 0.590
 Decision Tree            Accuracy: 0.588
 SVC (poly)               Accuracy: 0.584
 SVC (rbf)                Accuracy: 0.590
 voting_clf               Accuracy: 0.590

=== Elo_Diff_High (>=200) group ===
n_samples: 546
 AdaBoost                 Accuracy: 0.711
 Random_forest            Accuracy: 0.712
 Gradient Boosting        Accuracy: 0.711
 Decision Tree            Accuracy: 0.711
 SVC (poly)               Accuracy: 0.705
 SVC (rbf)                Accuracy: 0.711
 voting_clf               Accuracy: 0.711
```

The group with an Elo difference less than the absolute value of 100 (|Elo_diff| < 100) has the lowest accuracy and the largest sample size. The Elo_Diff_Mid (100–200) group shows improved accuracy, reaching 59% with a decent number of samples (1800). The Elo_Diff_High (>=200) group achieves an impressive 71% accuracy, strongly indicating that a large difference in Elo ratings leads to highly predictable outcomes.

c) Time Formats

```
=== bullet group ===
n_samples: 3972
 AdaBoost                Accuracy: 0.538
 Random_forest           Accuracy: 0.550
 Gradient Boosting       Accuracy: 0.534
 Decision Tree           Accuracy: 0.525
 SVC (poly)              Accuracy: 0.537
 SVC (rbf)               Accuracy: 0.544
 voting_clf              Accuracy: 0.540

=== blitz group ===
n_samples: 3587
 AdaBoost                Accuracy: 0.545
 Random_forest           Accuracy: 0.539
 Gradient Boosting       Accuracy: 0.544
 Decision Tree           Accuracy: 0.522
 SVC (poly)              Accuracy: 0.525
 SVC (rbf)               Accuracy: 0.534
 voting_clf              Accuracy: 0.536

=== rapid group ===
n_samples: 2015
 AdaBoost                Accuracy: 0.567
 Random_forest           Accuracy: 0.556
 Gradient Boosting       Accuracy: 0.562
 Decision Tree           Accuracy: 0.545
 SVC (poly)              Accuracy: 0.549
 SVC (rbf)               Accuracy: 0.556
 voting_clf              Accuracy: 0.558

=== classical group ===
n_samples: 426
 AdaBoost                Accuracy: 0.542
 Random_forest           Accuracy: 0.561
 Gradient Boosting       Accuracy: 0.538
```

Decision Tree	Accuracy: 0.502
SVC (poly)	Accuracy: 0.507
SVC (rbf)	Accuracy: 0.531
voting_clf	Accuracy: 0.521

Rapid games are the easiest to predict; bullet and blitz the hardest
Rapid (n=2015): Highest accuracy overall (AdaBoost 0.567, GBM 0.562, voting 0.558). Slower pace gives players more "modelable" behavior.
Bullet (n=3972) & Blitz (n=3587): Both in the 0.53–0.55 band, with Random Forest topping bullet at 0.550 and AdaBoost topping blitz at 0.545. The extreme time pressure in these formats probably introduces more noise—blunders that our features can't fully capture.
Classical (n=426): Despite being the slowest, the sample is smallest, so the scores bounce around (RF 0.561, AdaBoost 0.542, GBM 0.538). The high RF score (.561).

No single "winner" across every format—but some overall standouts
Random Forest has the best bullet accuracy and the best classical accuracy, and stays competitive in other slices.
AdaBoost wins on blitz and rapid, suggesting that it handles the "milder" noise of those formats better than GBM or RF.
Gradient Boosting sits close behind the top two in almost every slice—its stability makes it a solid all-rounder.
Voting ensemble (majority vote of all models) consistently ends up in the middle—better than Decision Tree, but usually a bit behind the top single method.

8. Neural Networks

We constructed two neural networks one simple and one deeper with batchnormalization and dropout. Here are the resutls:
Test accuracy (NN): 0.542
Test accuracy (deep MLP): 0.544

By incorporating Dropout and Batch Normalization, we were able to slightly enhance our neural network's performance, reaching an accuracy of 0.544.

9. Final Results

Here is the final list with the results

Model	Accuracy
Random Forest	0.548
AdaBoostClassifier	0.548
Deeper Network	0.544
GradientBoostingClassifier	0.543
SVM RBF Kernel	0.542
Simple NN	0.542
Voting Classifier	0.541
SGDClassifier	0.54
LinearSVC	0.54
Logistic Regression	0.537
SVM Polynomial Kernel	0.534
(PCA + Tree)	0.532
DecisionTree	0.527
BaggingClassifier	0.517
0. Most Frequent Dummy Classifier	0.491
0. Stratified Dummy Classifier	0.4549

Top of the pack are two tree-based ensembles—**Random Forest** and **AdaBoost**—each at **54.8%**. They narrowly beat out the deeper neural net (54.4%) and Gradient Boosting (54.3%).

Ensembles win: Combining many weak learners (trees for RF/AdaBoost) still edges out single learners or shallow nets.

All of our "strong" learners cluster in a very tight band between **0.540–0.548**:

- RF/AdaBoost: 0.548
- Deep NN: 0.544
- GBM: 0.543
- SVM (RBF) & Simple NN: 0.542
- Voting Ensemble: 0.541
- Linear models (SGD, LinearSVC) sit at 0.540.

Take-away: We have essentially hit a ceiling around 55% accuracy with these features. Different algorithms trade off a few hundredths of a point, but none break through much further. Classical "interpretable" models (Logistic Regression 0.538, plain Decision Tree 0.527) and feature-reduction (PCA+Tree 0.532) sit just above—or barely above—the **Most Frequent Dummy** (0.491) baseline.

Regression

Here, we frame the problem as a regression task, predicting the game score as a numerical value: 1 for white wins, 0.5 for a draw, and 0 for black wins. We again preprocessed the data by dropping the redundant "BlackElo" column and focusing on the top ten "Opening_name" values, categorizing the rest as "Other." We kept the numerical "Score" column as our target and used the same data splitting and preprocessing pipeline established earlier, which will be used for all our regression models.

As a performance benchmark, we trained two dummy regressors:

- Dummy Regressor #1, which predicts the mean

```
=== Dummy (mean) Regressor ===
MSE:  0.235
MAE:  0.472
```

- Dummy Regressor #2, predicts the median of y_train

```
=== Dummy (median) Regressor ===
MSE:  0.235
MAE:  0.471
```

The goal of our regression modeling is to substantially reduce the Mean Squared Error (MSE) below the baseline of 0.235.

1. Linear Regression, Lasso and Ridge

We began with a **Linear Regression** with the following results:

```
 MSE: 0.2279
 MAE: 0.4597
  R²: 0.0320
```

Best Lasso

```
Test MSE: 0.2279
Test MAE: 0.4599
Test R²:  0.0321
```

Best Ridge
Test MSE: 0.2279
Test MAE: 0.4597
Test R^2: 0.0320

The performance is very close to our dummy models

2. **Polynomial**

For our polynomial regression model, we conducted a Random Search which identified a degree of four as the optimal parameter. This resulted in a slight decrease in the Mean Squared Error (MSE). The test set performance metrics were:
Test MSE: 0.2275
Test MAE: 0.4583
Test R^2: 0.0341

3. **TreeRegressor, RandomForestRegressor & ExtraTreesRegressor**

For each of these models, we performed a Random Search to optimize their hyperparameters. The test set performance metrics are as follows:

TreeRegressor:
Test   MSE: 0.2273
Test MAE: 0.4579
Test R^2 : 0.0349

RandomForestRegressor:
Test MSE: 0.2263
Test MAE: 0.4557
Test R^2: 0.0391

ExtraTreesRegressor:
Test MSE: 0.2284
Test MAE: 0.4619
Test R^2: 0.0302

Once again, the RandomForestRegressor shows the best performance, achieving a small but noticeable decrease in MSE to 0.2263.

4. **Advanced Methods: AdaBoostRegressor, GradientBoostingRegressor, StackingRegressor**

Using the same methods with before, using our pipelines and our random searches we trained three more advanced models. Les see how they performed.

AdaBoostRegressor:
Test MSE: 0.2277
Test MAE: 0.4599
Test R^2: 0.0330

GradientBoostingRegressor:
Test MSE: 0.2264
Test MAE: 0.4568
Test R^2: 0.0384

StackingRegressor:
Test MSE: 0.2261
Test MAE: 0.4561
Test R^2: 0.0400

The Stacking Regressor was training using three estimators: a RandomForestRegressor, an ExtraTreesRegressor and a GradientBoostingRegressor and had the lowest MSE so far of 0.2261

5. VotingRegressor

We constructed a VotingRegressor using all the previously trained regression models. However, this ensemble method did not outperform our best individual model.
VotingRegressor:
MSE : 0.2265
MAE : 0.4582
R^2 : 0.0382

6. Neural Networks

Analogous to the classification phase, we constructed two neural network models: a simple network and a deeper network incorporating Batch Normalization and Dropout. The test set performance of these models was as follows:
Simple NN:
Test MSE: 0.2276
Test MAE: 0.4588

Deeper Network:
Test MSE: 0.2284
Test MAE: 0.4590

The fact that the Deeper Network underperformed the Simple NN indicates that its architecture and parameters were probably suboptimal.

7. Custom Models

We developed two custom regression models. The first utilized the standard Elo expected outcome formula $\frac{1}{1+10^{(R_{opponent} - R_{player})/400}}$, achieving the following performance:

EloExpectedScoreDummyRegressor:

Mean Squared Error: 0.2281
Mean Absolute Error: 0.4529
R^2 Score: 0.0311

The second custom model employed the same Elo expected outcome formula but optimized the denominator "b" instead of using the standard 400. Our analysis identified 580.0 as the optimal value for "b," resulting in the following performance:

Best_b_Regressor:

Optimal denominator (b): 580.0
Mean Squared Error: 0.2269
Mean Absolute Error: 0.4577
R^2 Score: 0.0363

With an MSE of 0.2269, this custom model outperformed several of our previously trained regression models.

8. Final Results

Here is the final list with the results

models	MSE
StackingRegressor	0.2261
RandomForestRegressor	0.2263
GradientBoostingRegressor	0.2264
VotingRegressor	0.2265
Best_b_Regressor	0.2269
TreeRegressor	0.2273
Polynomial	0.2275
Simple NN	0.2276
AdaBoostRegressor	0.2277
Linear Regression	0.2279
Best Lasso	0.2279
Best Ridge	0.2279
EloExpectedScoreDummyRegressor	0.2281
ExtraTreesRegressor	0.2284
Deeper Network	0.2284
Dummy (mean) Regressor	0.235

Ensembles on top

- **StackingRegressor** slightly wins with 0.2261 — beating every single model by a little.
- Right behind are the familiar tree-based ensembles: **RandomForest** (0.2263), **GradientBoosting** (0.2264), and the simple **VotingRegressor** (0.2265).

Marginal gains over baseline

- Our best model (stacking) brings MSE down from the DummyMean's 0.235 to 0.2261, a ~ 4% reduction. That's real improvement, but the fact that most models cluster in the 0.226–0.228 band tells us we have almost hit the ceiling of what these features can explain.

Complex ≠ better

- The **Deeper Network** and **ExtraTreesRegressor** actually sit near the bottom (0.2284), performing worse than simple tree ensembles and even the **Elo-expected-score dummy** (0.2281).
- Linear models (**LS, Ridge, Lasso**) all end up around 0.2279, not far behind **polynomial regression** (0.2275) or a tiny **simple neural net** (0.2276).

Tree stumps vs. full depth

- A plain **TreeRegressor** (0.2273) and **AdaBoostRegressor** (0.2277) also underperform the big ensembles
- Pleasant surpise the custom **Best_b_Regressor**(0.2269) reached the 5th place performing better than many standard models.

“The Endgame”

Reaching the conclusion of this project, we have identified the best models for our tasks and addressed all the initial questions. However, the performance of our best classifier, Random Forest, with an accuracy of 0.548 (only approximately 5% better than a dummy model), warrants further discussion.

As revealed during the EDA phase, roughly 75% of the games in our dataset were played between players with an absolute Elo rating difference of less than 50 points. This is likely due to Lichess's matchmaking algorithm, which, in most casual games, prioritizes pairing players of similar strength to maintain engagement. Significant skill disparities can lead to dissatisfaction for both participants.

Formats that allow for matches between players of varying strengths, such as tournaments, utilize different pairing criteria based on current scores or round-robin structures. However, the time commitment required for tournaments means many players participate infrequently.

Consequently, the prevalence of games between similarly rated players leads to more balanced outcomes (near equal wins and losses), making it inherently challenging for our models to achieve significantly higher prediction accuracy!

The reality is that many factors influencing a game's outcome are missing from our data or are simply impossible to capture. A player's current mood or fatigue can play a big role in how they perform. Also, player IDs could unlock valuable feature engineering, like looking at their Elo in different time controls (it's not the same for blitz, rapid, etc.) or their playing style and where they are in their chess journey – are they steadily improving, stable, or declining? Moreover, while we grouped around 500 openings and their variations into just ten, the specifics of those openings likely matter.

Finally, the biggest issue in our project turned out to be predicting draws. Most of our models showed zero precision and recall for this outcome, meaning they couldn't classify any game as a draw. The problem was that with very similar inputs, we saw all three results (wins for White, wins for Black, and draws), just with different frequencies, and draws were the least common. But we have an idea to fix this!

Our solution involves using our best Random Forest model (best_random_forest) and its .predict_proba method. This gives us probabilities for each outcome, including a probability for a draw, like this:

Black Win: 50.5%   Draw: 4.1%   White Win: 45.5%

The Python code is available in the 'Predictions_Final' Jupyter notebook, and the first 5 predictions are shown below:

Game 1:

WhiteElo=2066 | EloDif= -42 | Opening_name=Queen's Gambit | Time_format= bullet | Increment_binary=No

Predictions:

Black Win: 50.5%

Draw: 4.1%

White Win: 45.5%

Game 2:

WhiteElo=1570 | EloDif= 9 | Opening_name=Sicilian defense | Time_format= bullet | Increment_binary=Yes

Predictions:

Black Win: 50.2%

Draw: 2.9%

White Win: 46.9%

Game 3:

WhiteElo=1487 | EloDif= -31 | Opening_name=Scandinavian Defense (Center-Counter Defense) | Time_format= bullet | Increment_binary=Yes

Predictions:

Black Win: 54.8%

Draw: 3.3%

White Win: 41.9%

Game 4:

WhiteElo=2382 | EloDif= -32 | Opening_name=Queen's Pawn Game | Time_format= rapid | Increment_binary=No

Predictions:

Black Win: 46.5%

Draw: 8.0%

White Win: 45.5%

Game 5:

WhiteElo=2266 | EloDif= -101 | Opening_name=Other | Time_format= blitz | Increment_binary=No

Predictions:

```
 Black Win: 53.2%

   Draw: 6.6%

   White Win: 40.1%
------------------------------------------------------------
```

This output looks promising!

The end… for the time being.

Authored by

Dimitris Kaisaris