UNIVERSITÀ DI PISA

Data Analysis and Mining on Lichess Data
(Distributed Data Analysis and Mining Project)

Ruggero Anello

A.Y. 2023/2024

Contents

Chapter 1

Data Understanding and Preparation

1.1 Introduction

This report presents an overview of our investigative endeavors and findings on a dataset of games played on the popular chess website ”Lichess.org” during July 2016. The dataset, consisting of 6.25 million records and originally containing 15 features, required extensive cleaning and feature engineering to enhance its usability and insights. Our initial task focused on cleaning the data to ensure accuracy and consistency. Subsequently, we employed various techniques to generate additional numerical features. After preparing the data, we conducted several data understanding tasks, analyzing data distributions and uncovering intriguing insights within the dataset.

1.2 Data Semantic

In Table 1.1, we provide details about dataset variables: their names, concise descriptions, and types.

Attribute name	**Description**	**Type**
White	Name of the user playing with the white pieces	Categorical
Black	Name of the user playing with the black pieces	Categorical
WhiteElo	ELO score of the user playing with the white pieces	Continuous
BlackElo	ELO score of the user playing with the black pieces	Continuous
WhiteRatingDiff	How much the game affected White ELO Rating.	Continuous
BlackRatingDiff	How much the game affected Black ELO Rating	Continuous
Result	Result of the game, ”1-0” (White Won), ”1/2-1/2” (Draw), ”0-1” (Black Won)	Categorical
UTCDate	Date of the game in Universal Time Coordinated (UTC)	Categorical
UTCTime	Time of the game in Universal Time Coordinated (UTC)	Time
Event	Type of game (Time Control, Tournamenet or not)	Categorical
ECO	Opening in ECO, i.e. Encyclopedia of Chess Openings, encoding	Categorical
Opening	Name of the opening played in the game.	Categorical
TimeControl	Time control for the game, indicating the total time for each player in seconds, plus the increment per move if applicable	Categorical
Termination	Reason for the game’s end, e.g. checkmate, resignation, or timeout	Categorical
AN	Movements in Algebraic Notation, describing the sequence of moves made during the game	Text

Table 1.1: Dataset variables description

1.3 Data Quality

1.3.1 Duplicate Rows Assessment

During the duplicate rows assessment, it was found that only one duplicated row existed in the dataset. This redundant entry was promptly identified and removed to ensure the integrity and accuracy of the data.

1.3.2 Missing Values

In the assessment of missing values, it was observed that there were 4668 missing entries in the columns `WhiteRatingDiff` and `BlackRatingDiff`, and they were removed from the dataset. The decision to remove these entries was based on the difficulty in accurately estimating the true value of the missing data, and on the fact that there were no summary statistics suitable to replace them. Additionally, these variables will be included in the initial attempt of our classification task, and ensuring their completeness is crucial for the effectiveness of the analysis.

1.4 Variable Transformations and Feature Engineering

Since our dataset was limited in numerical features, our initial approach was to generate some additional features to enhance the dataset's utility for the subsequent classification task. One of the features we created is `n_moves`, which represents the total number of moves in each game.

To create the `n_moves` feature, we developed a function with the following steps:

1. **Extract Moves:** We used the column `AN`, which contains a string of all moves for each game (e.g., `"1. e4 d5 2. Qf3 dxe4 ... 32. Be3 Qxg2# 0-1"`).

2. **Remove Endgame Result:** We removed the last three indices from this string, as they represent the endgame result, which is not necessary for this feature since that information is captured in a separate column.

3. **Split Moves:** The remaining string was split into individual move elements, where moves are separated by spaces and move numbers are followed by a period.

4. **Count Moves:** We identified the highest move number in the sequence and multiplied it by two. If, following this number, there were two elements, no further action was taken. If there was only one element after this highest move number, we subtracted one from the total count.

This method ensured that the `n_moves` feature accurately reflects the total number of moves in each game.

We also observed that the `Result` column contained only three unique values: `"1-0"`, `"1/2-1/2"`, and `"0-1"`, which correspond to a win for White, a draw, and a win for Black, respectively. To facilitate subsequent feature engineering tasks involving performance metrics, we decided to convert these categorical values into numerical format. Specifically, we mapped these results as follows: 1 for White Win, 0 for Draw, and -1 for Black Win. This transformation enables us to quantify performance in a more actionable manner. For instance, when evaluating a player's total performance with a specific opening while playing as White, we can sum these numerical values to obtain a summary of his performance. A high positive total indicates strong performance, a high negative total reflects poor performance, and a value near zero suggests a balanced performance, with many draws or nearly equal numbers of wins and losses.

Next, we proceeded to create four additional features. The first feature, `w_opening_n_games`, was created by grouping our data by `White` (Username of the player with the white pieces) and `ECO` (opening played in the game, in ECO encoding), and then performing a count to determine how many games the White user played with a specific opening. This count, renamed `w_opening_n_games`, was then joined with the original dataset on matching `White` and `ECO`. For `w_opening_performance`, the same grouping was applied, but this time the aggregation function used was the sum of `Result`, as explained in the previous paragraph, and that represents the total performance of the White player with a specific opening. Similarly, the same procedure was applied to create `b_opening_n_games` and `b_opening_performance`, but grouping by `Black` (Username of the player with the black pieces) and `ECO`.

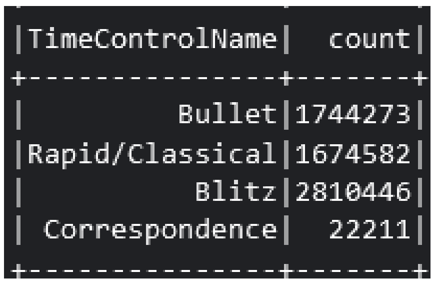
1. `w_opening_n_games`: Group by `White` and `ECO`, count the number of games, and join with the original dataset.

2. `w_opening_performance`: Group by `White` and `ECO`, sum the `Result` values, and join with the original dataset.

3. `b_opening_n_games`: Group by `Black` and `ECO`, count the number of games, and join with the original dataset.

4. `b_opening_performance`: Group by `Black` and `ECO`, sum the `Result` values, and join with the original dataset.

Subsequently, we encountered an issue with our data in the `Event` column. This feature required extensive data cleaning due to inconsistencies in data entries. For example, when grouping by `Event`, both "Blitz" and "Bullet" appeared twice due to trailing spaces or similar issues. Another area of improvement was the distinction between normal and tournament versions of each time control, such as "Blitz" and "Blitz tournament". Since tournaments on Lichess are not a major focus for us and we only care about proficiency and experience in the different time controls, we decided to map all the data into four categories: Bullet, Blitz, Rapid/Classical, and Correspondence. We combined Rapid and Classical into a single category because we noticed that many time controls categorized as Rapid on Lichess were often mislabeled as Classical. Furthermore, the difference between these two is not substantial in online chess, and Classical on Lichess has a very low game count. The column was renamed `TimeControlName`, and the final result is shown in Figure 1.1. As expected, Blitz is the most played time control, due to its balance of quickness and breathing space, followed by Bullet and Rapid/Classical. All this work was essential for developing our final four features: `w_timecontrol_n_games`, `w_timecontrol_performance`, `b_timecontrol_n_games`, and `b_timecontrol_performance`. These features were created in the same way as their "opening counterparts" previously explained, but by grouping based on `TimeControlName` instead of `ECO`. Thus, these features represent the number of games each user (both White and Black) has played with a specific time control and their performance in those games with respect to the sum of `Result`. Lastly, we transformed `UTCDate` from its original string format to a Date format, and that is crucial for enabling future analysis.



```
|TimeControlName|  count|
+---------------+-------+
|         Bullet|1744273|
|Rapid/Classical|1674582|
|          Blitz|2810446|
| Correspondence|  22211|
+---------------+-------+
```

Figure 1.1: `TimeControlName` distribution

1.5 Variable Distributions

In this section, we analyze the distribution of the data to gain insights into the characteristics of various features.

1.5.1 Numerical Features

The primary focus is on numerical features such as players' ELO and number of moves, as well as our custom-created features. Upon examining the distribution of the `WhiteElo` and `BlackElo` features, we observed that both closely follow a normal distribution, and this finding aligns with our initial expectations. However, the mean and median values of `WhiteElo`, as shown in Figure 1.2, are slightly higher than anticipated, and the same can be said for `BlackElo`. We initially expected these central tendencies to be closer to the default ELO rating of 1500 assigned by Lichess upon registration. One potential explanation for this deviation is the lower popularity of Lichess compared to its main competitor "chess.com". Given that the latter is more prominent and appears first in search results, new players are more likely to create accounts there, possibly leading to a slightly higher average ELO among the Lichess player base. The distribution of `n_moves` also approximates a normal distribution, albeit less sharply than the previous ones. This is understandable as games typically do not end in very few moves or extend for an excessive amount of them. The observed mean of 67.88 and median of 64.00 moves are consistent with domain knowledge and expectations, indicating a reasonable average game length. When it comes to the distributions of the custom features with "n_games", which counts the number of games played w.r.t opening or time control, they are positively skewed, resulting in a median significantly lower than the mean. This skewness is reasonable given the large number of unique users, many of whom do not play a high volume of games. Additionally, players often switch between different openings and time controls, further contributing to this distribution pattern. Finally, the custom features with "performance" resemble a normal distribution with a median around zero. This outcome is expected for a large dataset, as wins and losses tend to balance out over time.

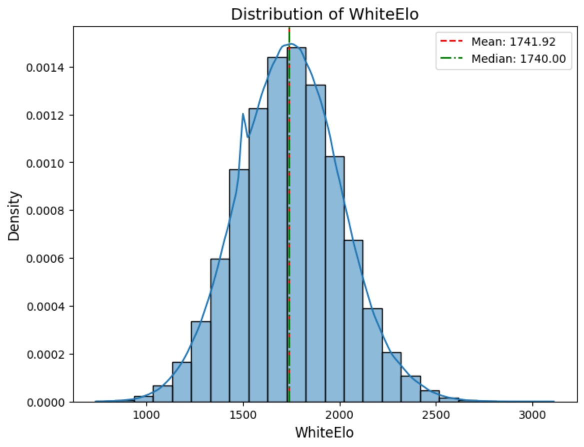


Figure 1.2: `WhiteElo` distribution

1.5.2 Categorical Features

When it comes to Categorical features, the distribution of `TimeControlName` has already been analyzed, and the `ECO` distribution does not reveal any significant insight, as the number of games for each opening encoding is relatively uniform, at least for the most common openings. The distribution of attribute `Termination` presents a more interesting pattern. The majority of games concluded with a "Normal" ending (checkmate or stalemate), which was expected, and those are followed by games lost due to time expiration. Then, there are less than 15000 records of games ended by forfeit, which is surprisingly lower than anticipated, and only 124 games terminated due to rule infractions, primarily involving player disqualification for cheating or other prohibited actions. Lastly, the distribution of the `Result` feature indicates that White wins slightly more games than Black, with approximately 3.1 million victories for White compared to 2.9 million for Black. Drawn games are notably fewer, less than 240000.

1.5.3 Date/Time Features

The analysis of the `UTCDate` feature does not reveal significant insights, as the number of games played each day in July 2016, which is the month that constitutes our dataset, remains relatively constant. No noticeable seasonality patterns either are present in the dataset. The `UTCTime` feature shows more interesting insights, as illustrated in Figure 1.3. The data clearly shows that the most active hours w.r.t number of games are between 16:00 and 21:00 UTC, with a slight dip in activity between 19:00 and 20:00. The least active period is between 01:00 and 07:00 in the morning. These patterns, however, align with typical user behavior and are not unexpected.

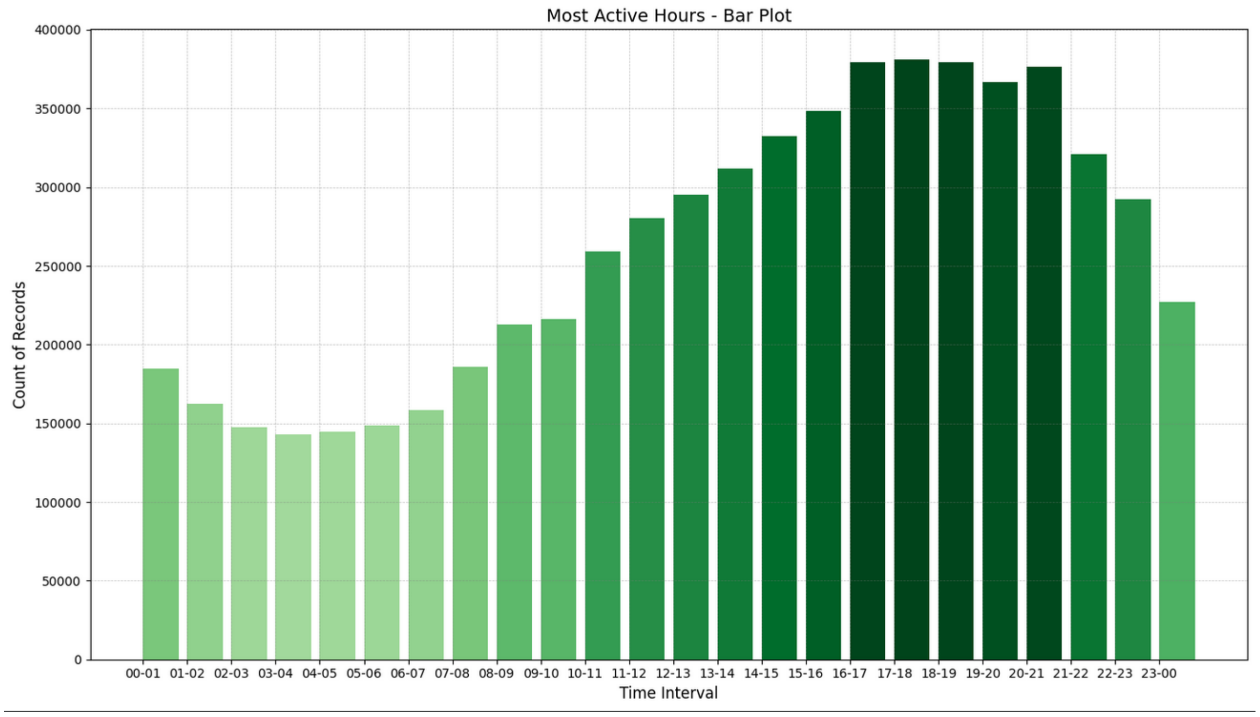


Figure 1.3: `UTCTime` distribution

1.6 Insights on Sicilian Defense: Open, Najdorf Variation

We conducted an in-depth analysis of the Sicilian Defense: Open, Najdorf Variation, including all its variants from ECO codes B90 to B99. This opening is well-known and confrontational for Black, recognized for its complexity and difficulty to master. Our initial findings indicate that the average `BlackElo` for players using this opening is 1899.3, significantly higher than the overall mean and median `BlackElo`. This suggests that the Najdorf Variation is predominantly played by more experienced players, likely due to its challenging nature. Furthermore, the overall performance for Black using this opening, represented by the algebraic sum of the `Result` values, is -971. This is a value lower than `b_opening_performance` median of 0 and mean of -0.08, reinforcing the confrontational idea of the Najdorf Variation, meaning that it makes you win many games even as Black once you master it. To gain deeper insights, we segmented `BlackElo` into eight bins, ranging from '0-1199' to '>=2400'. The highest ELO group (>=2400) achieved the best results with a win rate of 64.92% and a performance per game of -0.35. This high win rate further highlights this confrontational nature, that's why Najdorf Variation is often employed in must-win situations for Black. A negative performance per game indicates success, as we expect negative values for an opening played by Black. The ELO group '1200-1399' had the poorest results, with a win rate of 36.94% and a performance per game of 0.23. These metrics suggest that less experienced players struggle significantly with the complexities of the Najdorf Variation. Finally, in addition to calculating performance per game and win rate for each ELO group, we also computed draw rate and loss rate, as shown in Figure 1.4. The graph provides valuable insights for intermediate players considering the Najdorf Variation. Statistically, the turning point at which playing the Najdorf becomes advantageous, with a win rate slightly above 50% and a performance per game of -0.05, is the '1800-1999' ELO range. For players in the '1600-1799' range, the data shows a loss rate of over 50% and a performance per game of approximately 0.07, indicating that the Najdorf Variation might not be optimal for this group. While individual preferences and enjoyment are crucial, players focused solely on improving their ELO rating in online chess should consider these findings when choosing their openings. Overall, this analysis confirms that the Sicilian Defense: Najdorf Variation is a challenging yet rewarding opening for skilled players.

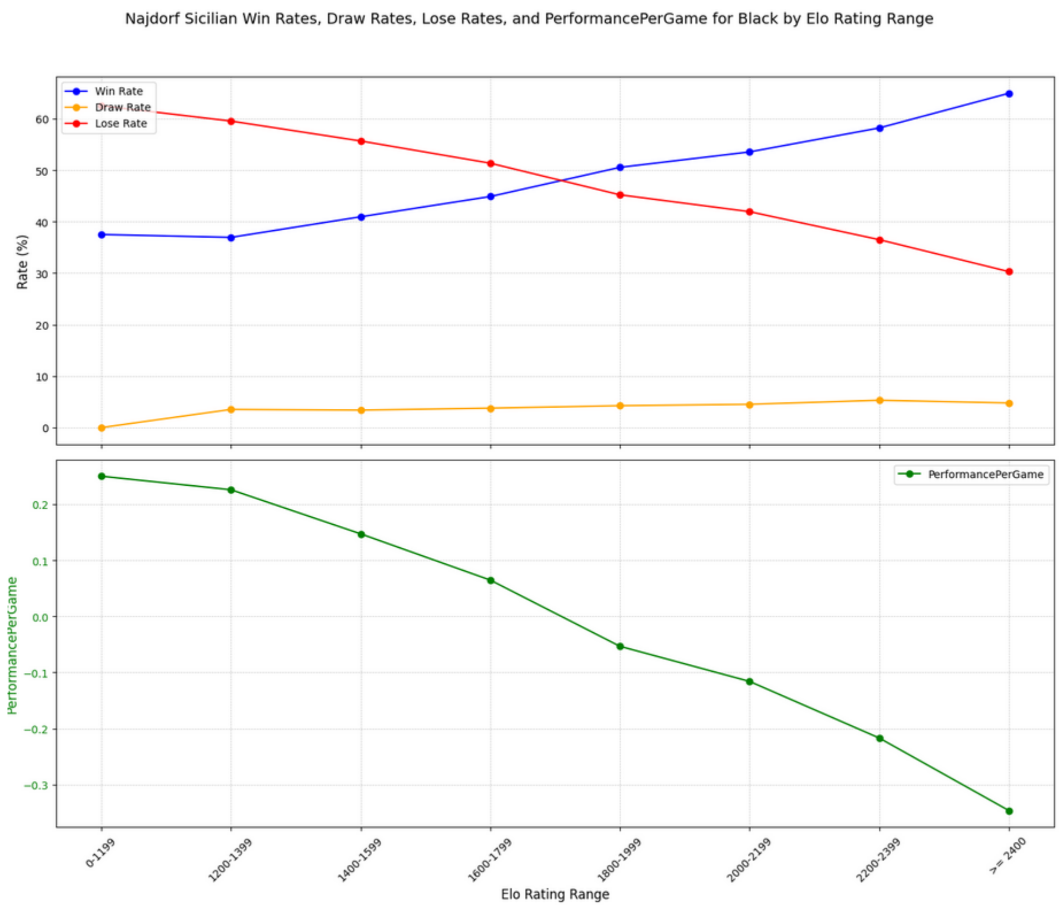


Figure 1.4: Najdorf Win/Lose/Draw Rates and Performance per game by ELO group

1.7 Insights on Resignations and Draws

Another important question we investigated was the distribution of draws and resignations across the different ELO groups. These metrics provide nuanced insights into player behavior and game dynamics, requiring a deep understanding of chess to interpret effectively. First, we analyzed the draw rates (number of games ending in a draw in an ELO bin divided by the total number for that bin). This distribution is negatively skewed, and the relative number of draws consistently increases from the first bin to the last one. This trend is understandable: at lower levels, players make more mistakes, leading to decisive outcomes rather than draws. As skill levels rise, players are better at defending losing positions and securing draws. Additionally, in high-level games with minimal mistakes, draws become more common unless one player has exceptional endgame skills to convert slight advantages into wins. In contrast, the distribution of resignation rates (the number of games ending in resignation in each bin divided by the total number for that bin) resembles a normal distribution. This was unexpected, as we hypothesized that stronger players would resign more often in losing positions, recognizing the improbability of a comeback against equally skilled opponents. However, two factors likely contribute to this distribution. Stronger players, even when losing, might continue to play, creating the so-called ”counterplay” and looking for opponents’ mistakes to turn the game around or force a draw. Additionally, weaker players might be more prone to ”rage-quit” (i.e., for-

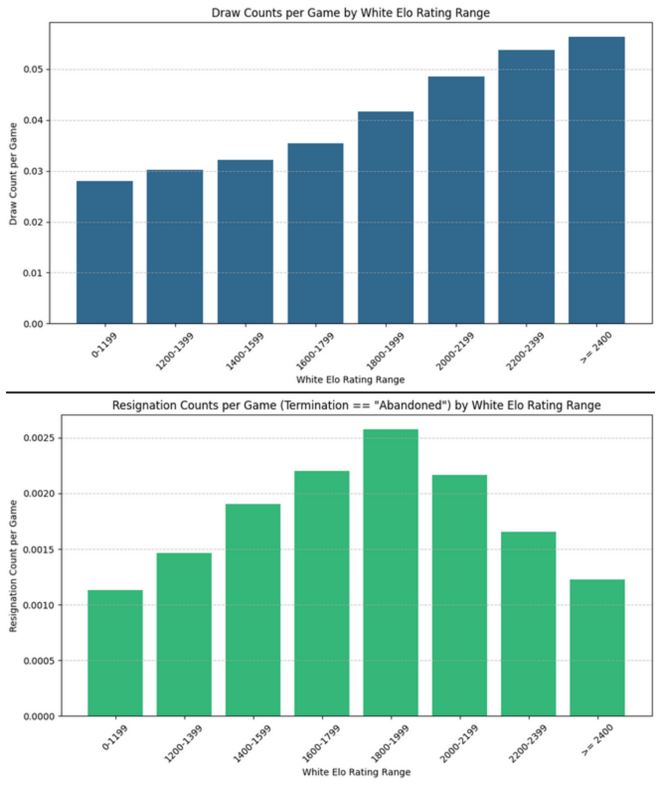


Figure 1.5: Draw rates and Resignation rates by ELO group

feiting from a game because of anger), despite still having draw or win chances. Both distributions are shown in Figure 1.5

1.8 Correlation Matrix and eventual variables removal

In Figure 1.6, we present the correlation matrix for all numerical variables in our dataset. Analyzing the correlation matrix, we observed that the highest correlation coefficient is between `WhiteElo` and `BlackElo`, with a value of 0.71. Despite this strong correlation, `BlackElo` carries essential information that we cannot disregard entirely. For our first classification run, we decided to exclude the `BlackElo` attribute, but we retained `WhiteRatingDiff` and `BlackRatingDiff`, which express how much ELO ratings for White and Black changed after a game, but encapsulate also the information we would gain by mantaining `BlackElo`. For instance, if a player with White, with a 2500 rating, plays an opponent with 1500 ELO, the difference in ELO between the two players is reflected in the aforementioned attributes (e.g., White gains 0 and Black loses 0 if White wins, whereas Black gains 30 and White loses 30 if Black wins). In our second classification run, we aimed for a more "unbiased" prediction of the game result over maximizing performance metrics, which will be later explained. For this run, we instead included `BlackElo` and removed `WhiteRatingDiff` and `BlackRatingDiff`. Additionally, we noted a moderate correlation (coefficients ranging from 0.5 to 0.64) between the performance metrics and number of games of a player given a specific time control and given an opening. Despite these correlations, we do not expect them to negatively impact our task and, more importantly, this information is very valuable for our analysis.

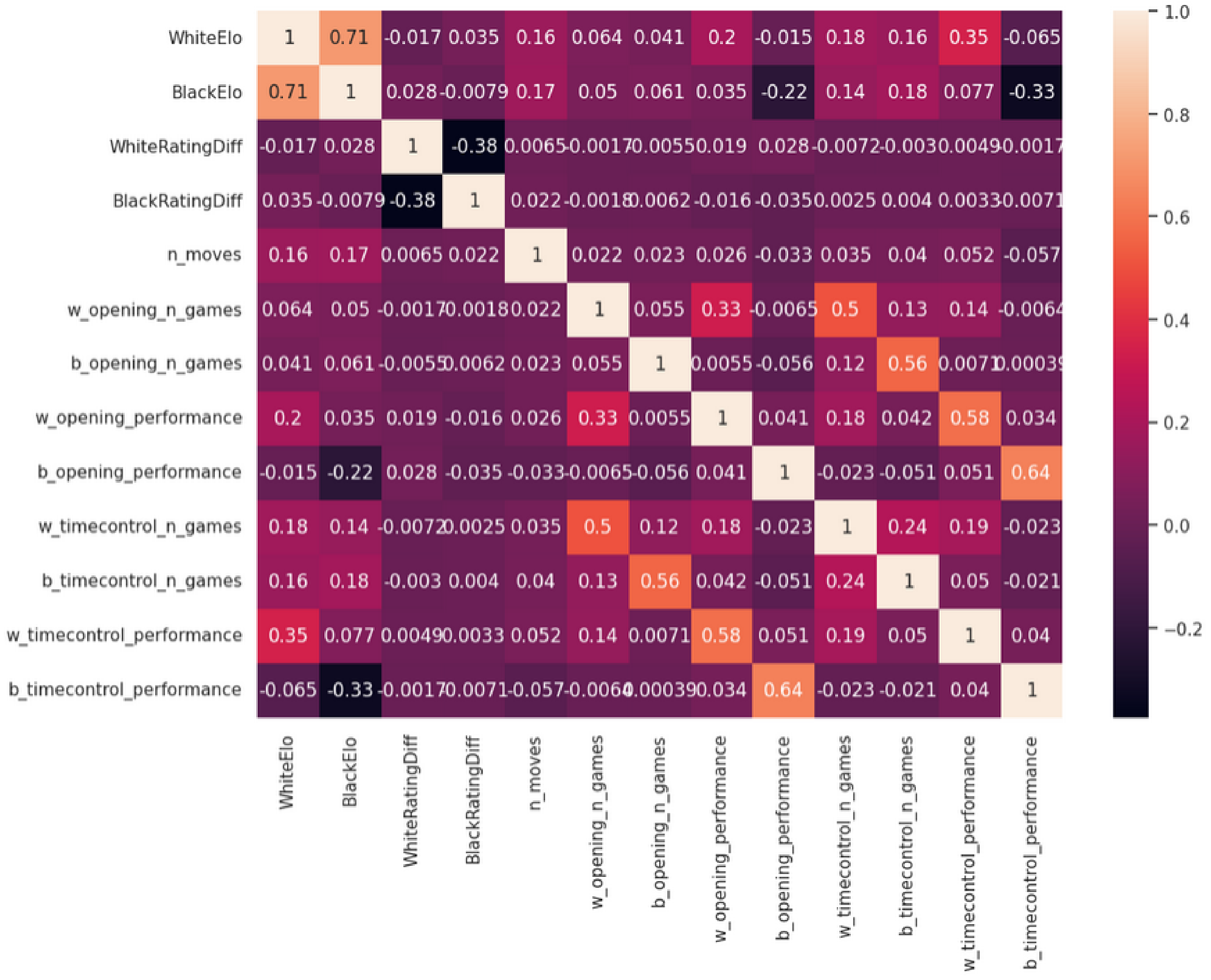


Figure 1.6: Correlation Matrix for our numerical features

Chapter 2

Classification

The objective of this task was to classify the `Result` column in our dataset using two different configurations.

2.1 First Configuration

The first approach aimed at maximizing performance measures such as precision and F1-score. This task serves the purpose of accurately predicting `Result` when a new record is added to our dataset with the `Result` value missing. Although this approach may have limitations in predictive capability, it can be useful for the aforementioned purpose. For this first configuration, we trained the models using all numerical features except for `BlackElo`, which was excluded due to its high correlation with `WhiteElo`. Additionally, two one-hot encoded categorical features, `Termination` and `TimeControlName`, were included. When it comes to preprocessing steps, first of all the target variable was remapped from 0 (Draw), 1 (White won), -1 (Black won) to 0 (Draw), 1 (White won), 2 (Black won) to avoid issues caused by negative values. Then, the dataset was split into training (70%) and testing (30%) sets, and finally data normalization was performed using MinMaxScaler to scale our features between 0 and 1.

2.1.1 Random Forest Classifier

We began our classification task by training a Random Forest model. We opted for a configuration with 20 trees, a choice that seemed to provide a reasonable trade-off between accuracy and resource expenditure. After testing the model on the test set, it performed impressively at first glance, achieving a global F1-score of 0.958 and an overall accuracy of 0.968. However, a deeper analysis of the confusion matrix and classification report revealed some significant issues. As shown in Table 2.1, while the F1-scores for the majority classes (1: White won, and 2: Black won) were outstanding at 0.98, indicating that the model was highly accurate in predicting these outcomes, the performance for class 0 (draw) was far less satisfactory. Specifically, the model demonstrated a high precision of 0.99 for class 0, which suggests that when the model predicted a draw, it was almost always correct. However, the recall for this class was only 0.18, and this contrast between precision and recall resulted in a low F1-score of 0.30 for draws.

Class	Precision	Recall	F1-Score	Support
0	0.99	0.18	0.30	71484
1	0.97	1.00	0.98	932748
2	0.97	1.00	0.98	869538
Accuracy		0.97		
Macro Avg	0.97	0.72	0.75	1873770
Weighted Avg	0.97	0.97	0.96	1873770

Table 2.1: Classification Report Random Forest Classifier - Approach 1

This poor recall rate for draws indicates that the model struggled significantly with class imbalance. While it accurately predicted when a game ended in a draw, it failed to capture most of these instances. The confusion matrix supported this observation, revealing a high number of false negatives and a low number of false positives, with only 88 records misclassified as 0 when the actual value was 1, and 97 records misclassified as 0 when the actual value was 2.

Recognizing the limitations posed by our Random Forest model, we considered increasing the number of trees to potentially improve recall for class 0. However, this approach would have been computationally expensive and could have worsened the model's tendency to maximize precision at the expense of recall.

2.1.2 Logistic Regression

Given these constraints, we decided to explore alternative classification algorithms. Our primary goal was to address the imbalance issue and improve recall for the draw (0) class without compromising the overall performance metrics. Logistic Regression, known for its ability to handle imbalanced datasets effectively, emerged as a suitable candidate for further investigation. The Logistic Regression model was trained using the same preprocessed dataset and we utilized the default parameters. As shown in Table 2.2, the Logistic Regression model demonstrated significant improvements in handling class imbalance, particularly for the draw class. The precision for class 0 (draw) was 0.82, and the recall was 0.55, resulting in an F1-score of 0.66. Although the recall for draws improved substantially compared to the Random Forest model, it still remained slightly lower than desired. However, the precision was reasonably high, indicating that when the model predicted a draw, it was likely correct. For classes 1 and 2, the Logistic Regression model maintained high performance, with F1-scores of 0.99 and 0.98 respectively. This shows that the model continued to perform well for the majority classes while also improving its performance on the minority class, making it better than the previous one.

Class	Precision	Recall	F1-Score	Support
0	0.82	0.55	0.66	71484
1	0.98	0.99	0.99	932748
2	0.97	1.00	0.98	869538
Accuracy			0.98	
Macro Avg	0.92	0.85	0.88	1873770
Weighted Avg	0.97	0.98	0.97	1873770

Table 2.2: Classification Report Logistic Regression - Approach 1

2.2 Second Configuration

As previously explained in Section 1.8, our second approach aimed to predict the `Result` in a less biased manner by excluding the features `WhiteRatingDiff` and `BlackRatingDiff`. These two features often provided an almost direct indication of the outcome, especially for the majority classes. In this scenario, we envision a situation where we not only have a missing value for the label `Result` but also for `WhiteRatingDiff` and `BlackRatingDiff`. Consequently, we cannot rely on this "shortcut" for prediction. The preprocessing steps remained consistent with the previous approach, except for the removal of the aforementioned variables and the inclusion of `BlackElo`, which now provides crucial information.

We used both Random Forest and Logistic Regression for this configuration. As expected, the results were significantly worse compared to the previous approach. The first confusion matrix, shown in Figure 2.1, corresponding to the Logistic Regression model, reveals a substantial number of misclassifications. The model struggled to correctly predict draws (class 0), as evidenced by the high number of false negatives and the very low number of true positives for this class. This indicates that the Logistic Regression model had difficulty identifying draws accurately. Additionally, while the majority classes (1 and 2) showed better performance, there was still a noticeable drop in the number of true positives compared to the initial configuration, resulting in a much lower overall performance. In summary, weighted average f1-score is 0.63, and accuracy is 0.64. The second confusion matrix (Figure 2.2) representing the Random Forest classifier, shows a slightly better performance than Logistic Regression but still highlights significant issues.

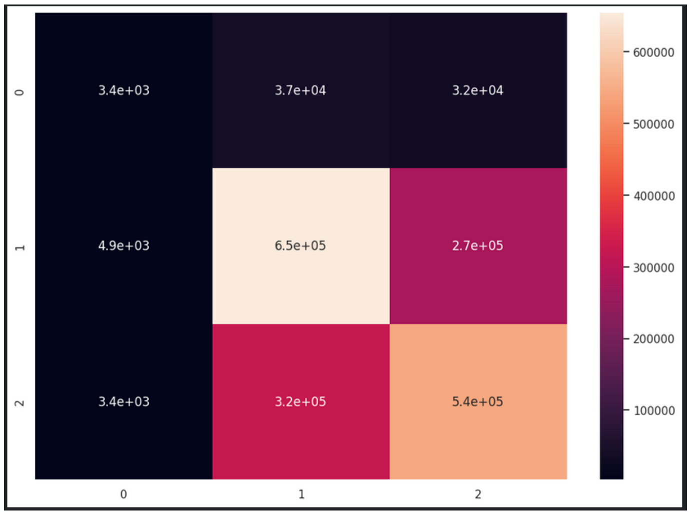


Figure 2.1: Confusion Matrix for LR (y: actual values; x: predicted values)

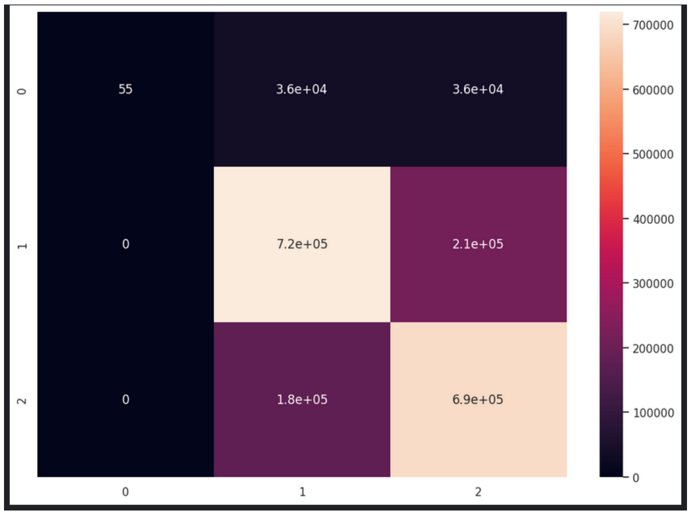


Figure 2.2: Confusion Matrix for RF (y: actual values; x: predicted values)

The model failed to accurately predict draws, with nearly negligible true positives for class 0 and a high number of false negatives. The majority classes (1 and 2) showed a higher number of true positives compared to Logistic Regression, but there was still a significant presence of misclassifications. This indicates that while Random Forest performed better than Logistic Regression in this configuration, weighted average f1-score of 0.739, and accuracy is 0.753, it still fell short of the performance achieved with the initial feature set.

2.3 Conclusions

In conclusion, the last configuration we tested demonstrates that the features `WhiteRatingDiff` and `BlackRatingDiff` were indeed crucial for achieving high predictive performance. The absence of these features led to a substantial increase in misclassifications for both Logistic Regression and Random Forest classifiers. Hence, accurately predicting the `Result` without the aid of `WhiteRatingDiff` and `BlackRatingDiff` would necessitate a larger set of more discriminative features or the application of more sophisticated classification techniques such as Deep Neural Networks to better understand and capture complex patterns within the data. Finally, although the first configuration demonstrated superior overall performance, both Random Forest and Logistic Regression faced challenges in handling the imbalanced class distribution. However, Logistic Regression exhibited a significant improvement in addressing the imbalance, achieving a 0.36 increase in the F1-score for class 0 compared to Random Forest, while maintaining comparable performance for the other classes. Therefore, Logistic Regression emerges as the preferred model for this classification task.