Predicting Chess Game Outcomes: A Statistical Learning Approach

Final Assignment - Statistical Learning
Master in ACIT, 2024-2025
OsloMet – Oslo Metropolitan University

Student: Sigurd Sandlie
Date: 16.11.2025

Table of Contents

---

1. Introduction

1.1 Problem Formulation

The primary research question addressed in this study is framed as conditional on early-game information:

Given the known opening (observed/declared in the first moves), together with pre-/early-game factors such as player ratings and time controls, can we predict the winner of a chess game (white, black, or draw)?

Additionally, we investigate a secondary question:

What factors influence game length (number of turns), and can we accurately predict how long a game will last?

1.2 Motivation and Context

In the age of online chess, platforms like Lichess and Chess.com serve millions of games daily. Understanding the factors that determine a game's outcome and duration from the outset has significant practical applications. This includes:

- **Enhanced Matchmaking:** Creating more balanced and engaging pairings for players.
- **Player Specific Insights:** Helping players understand the implications of their opening choices against other players of varying skill levels.
- **Platform Analytics:** Allowing platforms to forecast server load and manage resources based on predicted game durations.

This report moves beyond simply identifying the "best" player and instead applies statistical learning concepts to model the complex interactions between player skill, strategic choices (openings), and game constraints (time controls) to see how they collectively shape the game.

---

2. Methods and Data

This section outlines the dataset, preprocessing steps, and the statistical learning models applied. All analysis was conducted using Python with libraries such as Scikit-learn, Pandas and XGBoost.

2.1 Data Description and Preprocessing

This analysis utilizes the **Chess Game Dataset (Lichess)** from Kaggle (Mitchell, 2017), containing data from 20,056 chess games played on the online chess server Lichess. The dataset contains 16 original features, including player IDs, ratings, moves, time restrictions, and opening information.

2.1.1 Data Overview

The primary features of interest for this analysis are:

- **Target variables:**

1. **Winner:** A categorical variable with three classes: 'White' (49.86%), 'Black' (45.40%), and 'Draw' (4.74%). The dataset is thus imbalanced, a

critical consideration for model evaluation.

2. **Turns:** A numerical variable representing the total number of moves (ply) in the game. This feature is heavily right-skewed, with most games ending relatively quickly and a long tail of very long games.

- **Predictor Features:** white_rating, black_rating, opening_name, and increment_code (which details the time control).

This is a high-quality dataset, containing no missing values for any of the features present.

2.2 Feature Engineering

To improve model performance and interpretability, we engineered a small set of simple, informative features:

- **Rating difference:** rating_difference $=$ white_rating $-$ black_rating
- **Absolute rating difference:** $|$white_rating $-$ black_rating$|$ (equivalently, $|$rating_difference$|$)
- **Average rating:** avg_rating $= \frac{\text{white\_rating}+\text{black\_rating}}{2}$

These three cover who is stronger (the signed difference), how big the skill gap is (the absolute difference), and the overall level of the game (the average rating).

The increment_code variable (format: "minutes+seconds") was parsed to extract:

- **time_control_minutes**: Starting time in minutes.
- **time_control_seconds**: Increment in seconds.

Time control affects pace and decision pressure. Shorter clocks and smaller increments increase time pressure and error rates; more time generally leads to longer games. These fields let the models learn how timing relates to winner and length.

- **Opening category:** Simplified opening names by extracting the main category.
- **Opening category (encoded):** opening_category_encoded via LabelEncoder for model input.

Additionally, for classification only we engineered interaction terms rating_diff_x_opening, rating_diff_x_time, and avg_rating_x_opening, to capture how rating advantage interacts with opening choice and time budget. We also created opening-popularity measures opening_frequency, opening_rarity, and

opening_frequency_norm, as proxies for how common or rare openings relate to outcome tendencies beyond ratings and time controls.

2.3 Methodological Approach

This analysis employs a comprehensive statistical learning framework covering both supervised and unsupervised learning techniques.

2.3.1 Unsupervised Learning

Principal Component Analysis (PCA):

We use PCA to make the data easier to visualize and to understand which combinations of variables explain most variation. We standardize features first, then inspect the explained-variance curve to see how many components are needed for a faithful summary. PCA is used here for insight, not to reduce inputs for the supervised models.

Supervised Learning Pipeline (shared setup)

We used pipelines to avoid information leakage: all preprocessing and any resampling happen inside each cross-validation fold. Data were split 70/30 into train and test sets (stratified for classification) with random_state=42. We label-encoded opening_category, filled missing values (median for numbers, mode for categories) using training data, excluded post-game variables (turns, victory_status), and kept opening-based clusters for exploration only.

For classification, each model ran inside a pipeline with StandardScaler and SMOTE, and we selected models by balanced accuracy using 5-fold StratifiedK-Fold. For regression, linear models used standardized features and tree models used raw features. We also included a log-target linear model and a Poisson model because game length is right-skewed.

2.3.2 Supervised Learning: Classification

Problem: Predict the winner of a chess game (white, black, or draw) - a multiclass classification problem.

We evaluated three classifiers. Multinomial Logistic Regression is our interpretable linear baseline (features standardized; lbfgs solver; max_iter=1000).

Random Forest captures non-linear patterns and interactions and was tuned with 5-fold cross-validated grid search. XGBoost (when available) is a strong boosting model optimized for multiclass log loss.

This mix balances interpretability and flexibility. PCA suggested substantial class overlap, so a linear baseline is useful to calibrate expectations. Tree-based models can pick up non-linear relations among ratings, openings and time controls. Because draws are rare, we report balanced accuracy and use SMOTE inside cross-validation to help the minority class without leaking information.

We combined base rating features (white_rating, black_rating, rating_difference, abs_rating_difference, avg_rating), time-control variables (time_control_minutes, time_control_seconds), and context (opening_ply, is_rated, opening_category_encoded) with interaction terms (rating_diff_x_opening, rating_diff_x_time, avg_rating_x_opening) and opening-popularity measures (opening_frequency, opening_rarity, opening_frequency_norm).

Models were compared using 5-fold cross-validation (stratified for classification) on the training split, with preprocessing and SMOTE applied inside folds to avoid leakage. We then evaluated almost all candidate models on the held-out 30% test set for transparency, and we highlight the strongest performer in the Results.

Models were compared using cross-validation, and the best-performing model (based on balanced accuracy) was selected for final evaluation on the test set.

2.3.3 Supervised Learning: Regression

Problem: Predict game length (number of turns) - a continuous regression problem.

We fit Linear, Ridge (L2), and Lasso (L1) regressions on standardized features to capture linear signal and handle multicollinearity or sparsity. We also trained a Random Forest Regressor on raw features to probe non-linear structure. Because turns is right-skewed, we added a log-target linear model (predictions inverted back to turns) and a Poisson model on the original target.

The linear family provides interpretable coefficients and handles collinearity (Ridge) or automatic selection (Lasso). The log-target and Poisson variants match the skewed, count-like nature of game length. Random Forest offers a robust non-linear option. We standardize features for linear models; tree models use raw features.

We used the same base rating and time-control features, together with opening_ply, is_rated, and opening_category_encoded, but omitted interaction and opening-popularity features.

We compared regressors via 5-fold cross-validation on the training split (reporting CV RMSE) and also evaluated each model on the held-out test set (RMSE/MAE/R^2). We report all test metrics for transparency and emphasize the best performer in the Results.

Models were compared based on cross-validated RMSE. Best model selected for final evaluation.

2.4 Data Preprocessing

2.4.2 Categorical Encoding

opening_category was derived from opening_name and encoded with LabelEncoder so it could be used alongside numerical features. Numerical missing values were filled with the median and categorical with the mode using training data only. For classification, scaling and SMOTE were applied inside cross-validation pipelines to avoid leakage.

2.4.3 Train-Test Split

We used a 70/30 train–test split with random_state=42. For classification, the split was stratified to preserve class proportions in both sets; regression used the same split without stratification.

2.5 Evaluation Strategy

We evaluated models with 5-fold cross-validation, ensuring that all preprocessing (scaling, encoding, imputation) and, for classification, SMOTE resampling were fit only on training folds. This prevents information from the validation folds leaking into the training process and makes model comparisons fair.

For classification, model selection used balanced accuracy as the scoring metric in GridSearchCV because the draw class is relatively rare. We report accuracy, balanced accuracy, and weighted F1, and inspect confusion matrices to understand error patterns. SMOTE was applied inside the cross-validation loop to improve minority-class learning without contaminating validation folds.

For regression, we selected models by cross-validated RMSE and also report MAE and R^2 to summarize error magnitude and explained variance. Linear models operated on standardized features; tree-based models used raw features, matching their implementation in the analysis.

---

3. Results

3.1 Exploratory Data Analysis

3.1.1 Distribution of Target Variables

Winner distribution shows a mild white advantage and a small share of draws: White 49.86%, Black 45.40%, Draw 4.74%. This imbalance motivates the use of balanced accuracy and SMOTE in classification.

Game length (turns) is right-skewed with mean ≈ 60.47 and median ≈ 55, indicating many short-to-medium games and a long tail of very long games. This supports testing log-target and Poisson variants in regression.

Figure 1 summarizes the class imbalance (white/black/draw) motivating balanced accuracy and SMOTE; Figure 2 shows the right-skewed distribution of game length.

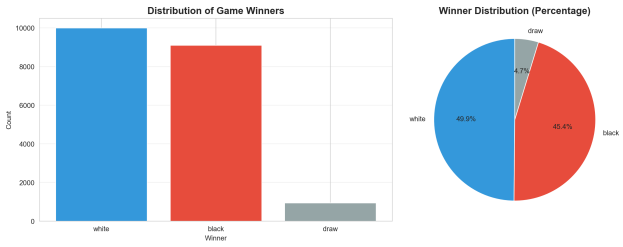

Figure 1: Winner distribution (White/Black/Draw).

Key observations: (i) class imbalance due to few draws; (ii) substantial right skew in turns; (iii) wide spread in ratings and time controls (see next subsections).

3.1.2 Rating Analysis

Ratings are centered around mid-classical Elo ranges with a broad spread. As expected, larger positive rating_difference (White stronger) shifts outcomes toward

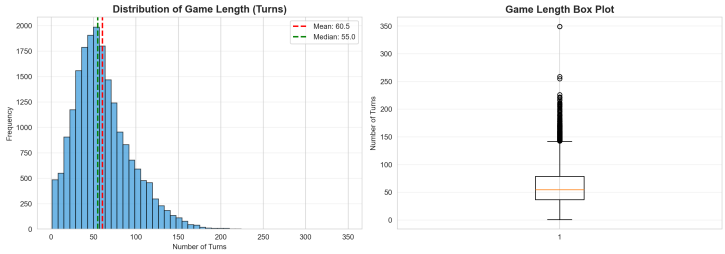

Figure 2: Game length (turns): histogram and boxplot.

White and negative values toward Black; near zero, outcomes are more balanced and draws occur slightly more often. This pattern aligns with the later feature-importance results where rating-based predictors play a central role.

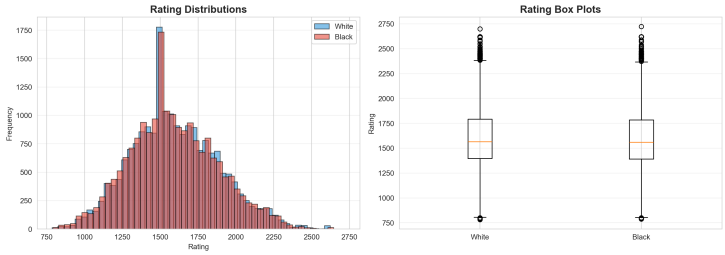

Figure 3: Rating distributions for White and Black Elo.

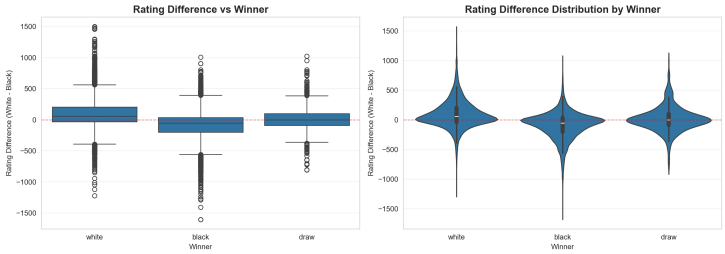

Figure 4: Rating difference vs winner (box plot and violin plot).

3.1.3 Opening Analysis

Openings span a wide set of named categories. Some categories are much more common than others, which is captured by our opening popularity features; these

correlate modestly with outcomes but help tree-based models capture opening-specific tendencies. Opening depth (opening_ply) varies and contributes small additional signal in classification and regression.

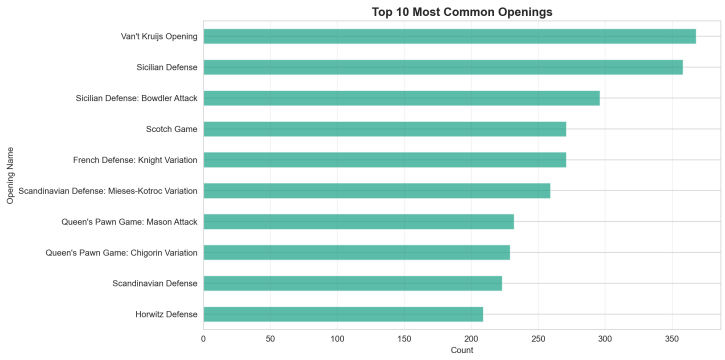

Figure 5: Top-10 most common openings (category frequency).

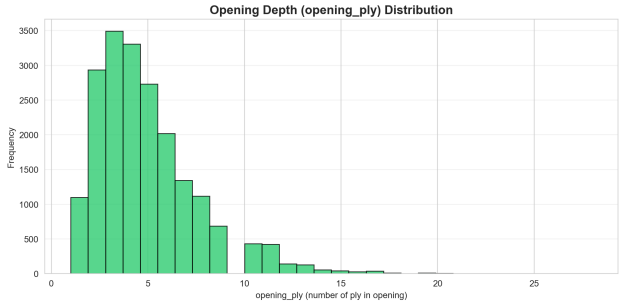

Figure 6: Opening depth (opening_ply) distribution.

3.2 Unsupervised Learning Results (PCA)

3.2.1 Principal Component Analysis

Explained Variance and Loadings:

PC1 explains 33.67% of the variance and PCs 1–2 together explain 52.41%; reaching 80% and 90% requires four and five components, respectively. Loadings indicate that PC1 largely captures overall player strength (high positive weights for avg_rating, white_rating, and black_rating), PC2 reflects time-control settings (time_control_seconds and time_control_minutes), and PC3 separates matches by

rating_difference, distinguishing games with sizeable rating gaps from balanced pairings.

The 2D PCA scatter (PC1 vs PC2; 33.67% and 18.75% variance, respectively) shows partial separation aligned with rating/time-control structure, while class labels (white/black/draw) overlap substantially, consistent with the moderate predictability observed in classification.

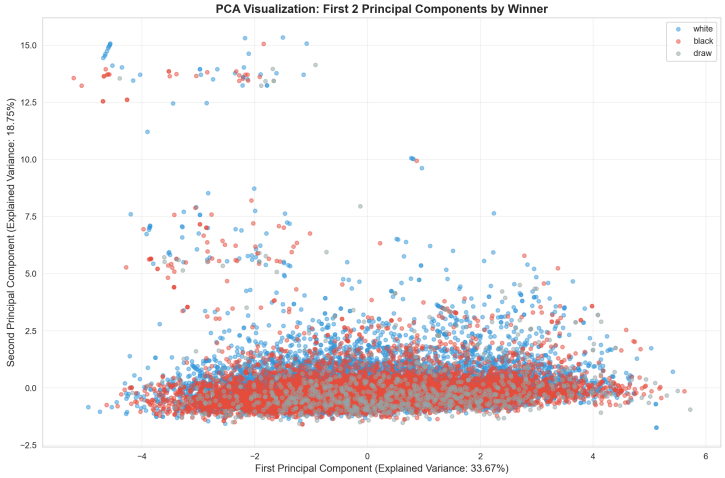

Figure 7: PCA scatter (PC1 vs PC2) colored by winner.

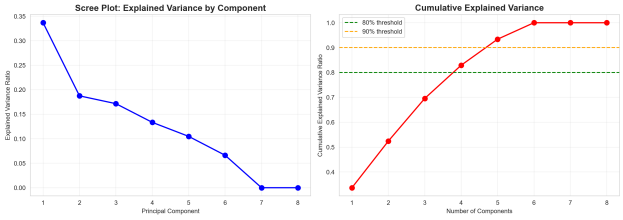

Figure 8: PCA scree/explained-variance plot.

3.3 Classification Results (Winner Prediction)

3.3.1 Model Performance Comparison

Model	Test Acc.	Test Bal. Acc.	Test F1 (W)	CV Bal. Acc.	CV Std
Majority baseline	0.4987	0.3333	0.3319	0.3333	0.0000
Logistic Regression (pipe)	0.4731	0.4408	0.5308	0.4622	0.0125
XGBoost (pipe)	0.5312	0.4451	0.5638	0.4688	0.0146
Random Forest (tuned)	0.5598	0.4549	0.5796	0.4717	NA
Random Forest (pipe)	**0.5997**	**0.4721**	**0.6033**	**0.4630**	**0.0137**

Best Model: Random Forest (Pipeline) with Test Balanced Accuracy of 0.4721 (Test Accuracy 0.5997; Test F1 0.6033; CV BA 0.4630 ± 0.0137).

3.3.2 Detailed Performance Analysis

Logistic Regression underperforms the tree-based models on all metrics (BA 0.441; F1 0.531), which is consistent with the non-linear structure introduced by interactions among ratings, openings and time controls. Errors concentrate on the minority Draw class and on borderline White/Black cases with small rating gaps.

Random Forest achieves the best overall performance (Accuracy 0.600; Balanced Accuracy 0.472; F1 0.603) and shows stable cross-validated scores (CV BA 0.463 ± 0.014). The confusion matrix confirms that Draw remains hardest, with most remaining mistakes split between White and Black when players are similarly rated.

XGBoost sits between Logistic Regression and Random Forest, improving over the linear baseline but not surpassing RF on this dataset and feature set. Given the moderate data size and the engineered interactions/popularity features, RF provides a strong bias-variance trade-off without extensive tuning.

Figure 9 highlights the Random Forest's confusion matrix, illustrating that draws remain the hardest class while white/black errors cluster around small rating gaps.

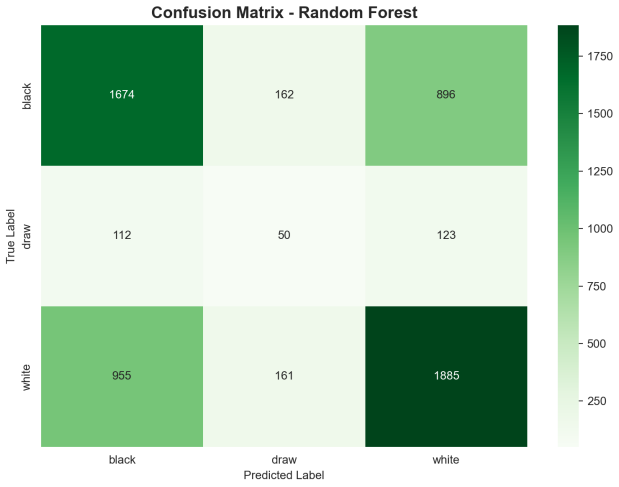

Figure 9: Random Forest normalized confusion matrix (test set).

3.3.3 Feature Importance

Across tree-based models, rating-based variables (rating_difference, avg_rating) and opening context (opening_category_encoded) dominate, with time control (time_control_minutes, time_control_seconds) also contributing. Interaction terms (e.g., rating_diff_x_opening) and opening popularity (opening_frequency, opening_rarity, opening_frequency_norm) add smaller but consistent improvements, indicating that certain openings and time budgets modulate how rating advantage turns into wins. Figure 10 summarizes the Random Forest feature importances.

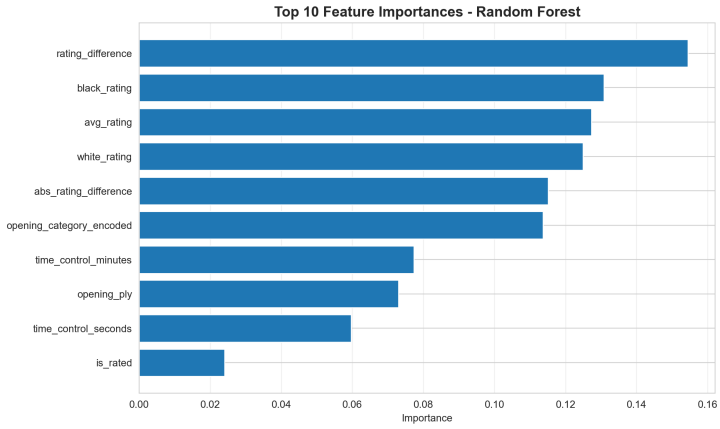

Figure 10: Random Forest classifier feature importances.

3.3.4 Confusion Matrix Analysis

White and Black are predicted with moderate success; Draw remains hardest due to low frequency and overlap with balanced pairings. Misclassifications cluster around near-zero rating_difference, where outcomes depend on in-game dynamics that are not in our pre-game features.

3.4 Regression Results (Game Length Prediction)

3.4.1 Model Performance Comparison

Model	Test RMSE	Test MAE	Test R^2
Linear Regression	32.5350	25.4782	0.0543
Ridge Regression	32.5370	25.4784	0.0542
Lasso Regression	32.5348	25.4763	0.0543
Random Forest	**32.4279**	**25.0964**	**0.0605**
Linear Regression (log target)	34.0544	25.5932	-0.0361
Poisson Regression	33.4568	26.0921	-0.0001

Best Model: Random Forest with RMSE $\approx$ 32.43 turns (MAE $\approx$ 25.10; $R^2 \approx$ 0.0605).

3.4.2 Detailed Performance Analysis

Linear, Ridge, and Lasso regressions perform nearly identically (RMSE $\approx$ 32.53; $R^2 \approx 0.054$), indicating limited linear signal in pre-/early-game features. Random Forest achieves the lowest error (RMSE $\approx$ 32.43; MAE $\approx$ 25.10; $R^2 \approx 0.0605$), a modest improvement consistent with small non-linear effects. The log-target Linear and Poisson variants underperformed on held-out data, reflecting that variance in turns exceeds what simple count or log-normal assumptions capture here.

3.4.3 Feature Importance for Game Length

Time control variables and rating levels provide the most useful signals for length, while opening context adds smaller effects. Even so, the low R^2 across models suggests that in-game dynamics, not present in pre-/early-game data, drive most variability in game length.

Figure 11 shows the Random Forest regressor's feature importances, with time-control variables and rating levels contributing most to length prediction.

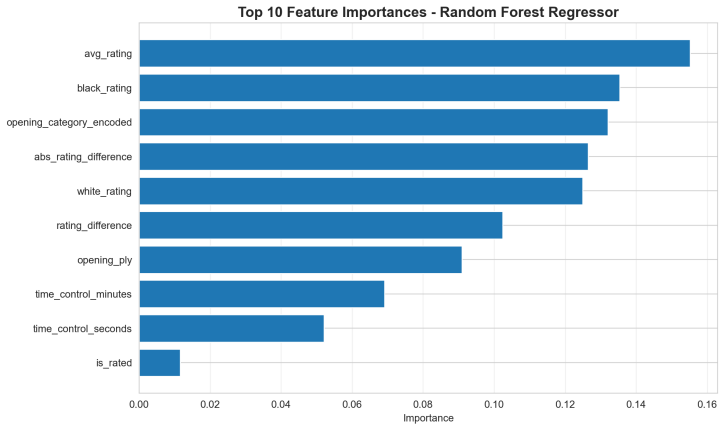

Figure 11: Random Forest regressor feature importances.

3.4.4 Residual Analysis

Residuals are wide relative to the target range, consistent with low R^2. Predicted vs. actual plots show large dispersion around the diagonal, especially for very long games, which the models tend to under-predict, again reflecting missing in-game information.

Figure 12 visualizes residuals and predicted-vs-actual turns on the test set, highlighting substantial dispersion, especially for very long games, consistent with low R^2.

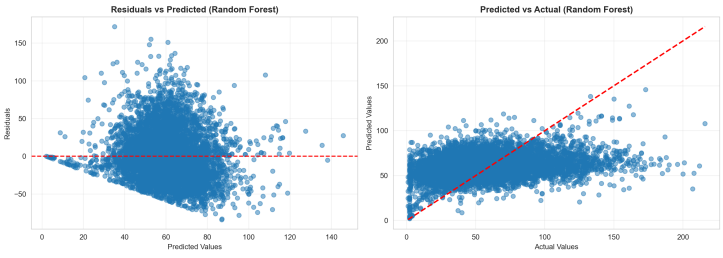

Figure 12: Predicted vs actual and residual diagnostics (test set).

4. Discussion

4.1 Key Findings

4.1.1 Classification Findings

Among classification models, the Random Forest pipeline performed best (Test BA 0.472; Test F1 0.603), clearly outperforming the majority baseline (BA 0.333). Tree-based models (RF/XGB) surpassed Logistic Regression, suggesting that non-linear relationships and interactions among ratings, opening category, and time control are important. The draw class remains the most challenging, so balanced accuracy is the appropriate headline metric. Feature-importance profiles indicate that rating difference/average rating, opening category, and time-control variables dominate.

Rating differences exert a strong effect on outcomes, with opening category and time control also contributing meaningfully. Conditional on knowing the opening, early-game information supports reasonable winner prediction without post-game leakage, but draws remain challenging.

4.1.2 Regression Findings

For game-length prediction, Random Forest achieved the lowest error (RMSE ≈ 32.43), but all models produced low R^2 (≈0.05–0.06), indicating limited explainability from pre-/early-game features. Linear, Ridge, and Lasso performed similarly, and the small advantage for Random Forest suggests only modest non-linear structure. The log-target Linear and Poisson variants underperformed, consistent with overdispersion and right-skew in turns.

Game length appears to depend largely on factors not present in pre-/early-game data (in-game dynamics/quality). The achievable predictive accuracy from these features is modest; time control and opening-related signals contribute, but are insufficient to explain most variance in turns.

4.1.3 Unsupervised Learning Findings (PCA)

PCA indicated that variation is dominated by player strength and time controls: PC1 aligned with average rating, PC2 with increments and base minutes, and PC3 with rating gaps. The PC1–PC2 scatter showed substantial class overlap, which justifies expectations for moderate classification performance and explains why

linear baselines underperform tree models. PCA was used for insight and justification only, it was not used to engineer features for supervised learning.

4.2 Methodological Reflections

4.2.1 Why These Methods Were Appropriate

For classification, Multinomial Logistic Regression provided an interpretable baseline on standardized features, appropriate given the overlap observed in PCA. Random Forest captured non-linear interactions among ratings, opening category, and time controls and delivered the strongest results with limited tuning. XGBoost served as a competitive boosting alternative but did not surpass RF on this dataset and feature set, which is consistent with moderate sample size and feature complexity.

For regression, Linear, Ridge, and Lasso assessed linear signal and addressed multicollinearity and sparsity, while Random Forest probed non-linear structure. The small advantage of RF and uniformly low R^2 confirm that most variability in turns is not explained by pre-/early-game features. Including log-target and Poisson variants was methodologically sound for a skewed, count-like target even though they underperformed on the held-out test set.

PCA was chosen to expose structure and aid communication: it summarized dominant axes (ratings/time), helping set realistic expectations for supervised performance. We explicitly did not inject PCA components into supervised pipelines to maintain strict CV hygiene and avoid leakage.

4.2.2 Alternative Approaches Considered

Support Vector Machines and broader gradient-boosting searches could be explored, but current results suggest diminishing returns over RF given data size and feature design. Deep neural networks or sequence models would better exploit move-level data and richer features, which were beyond scope. For unsupervised learning, DBSCAN/hierarchical clustering and UMAP/t-SNE may reveal finer patterns but add complexity without clear benefit to our supervised objectives. We prioritized interpretability, reproducibility, and leakage safety.

4.3 Limitations

4.3.1 Data Limitations

The dataset is moderately sized (~20k games) and drawn from online play, which may differ from tournament settings. Crucially, in-game information is missing (engine evaluations, move-quality summaries, time remaining per move, player style indicators), which likely explains the low R^2 for game length and the difficulty with draws. Platform-specific behaviors, rating calibration, and the period covered may also introduce biases affecting generalizability.

4.3.2 Methodological Limitations

We avoided leakage by fitting preprocessing and SMOTE inside CV, but relied on a single 70/30 test split rather than nested or repeated CV for model selection. Hyperparameter searches were kept modest for compute reasons, so small gains may be untapped. Linear models assume additive effects on standardized features; tree models can overfit without constraints, though our CV results were stable. Repeated/nested CV, probability calibration checks, and broader tuning would further harden conclusions.

4.4 Practical Implications

4.4.1 For Chess Platforms

Winner predictions around 60% accuracy (BA ~0.47) can inform matchmaking and rating diagnostics when conditioning on known openings and time controls. Even modest length predictions assist aggregate capacity planning (expected session loads by time-control type). Model explanations and feature importances provide transparent signals suitable for player-facing analytics.

4.4.2 For Chess Players

Rating gaps dominate outcomes; small gaps yield the most uncertainty and more draws. Opening category and time controls nudge odds and expected length but do not produce large deterministic effects, so players should align openings with their risk/time preferences without over-expecting outcome shifts. Since pre-/early-game features explain little of length, improving in-game decision quality remains the primary path to longer-term performance.

5. Conclusion

5.1 Summary of Contributions

We built a leakage-safe, cross-validated pipeline to predict chess game outcomes and lengths from pre-/early-game information. For winner prediction, the Random Forest pipeline achieved Test Accuracy 0.5997, Balanced Accuracy 0.4721, and weighted F1 0.6033, showing moderate predictability without using any post-game variables. For game length, the Random Forest regressor reached RMSE $\approx$ 32.43 turns (MAE $\approx$ 25.10; $R^2 \approx 0.0605$), indicating limited explainability from early signals. PCA complemented these results by summarizing variation mainly along rating and time-control axes and by justifying expectations for supervised performance. Overall, the study demonstrates careful CV hygiene, target-leakage prevention, and transparent reporting across models.

5.2 Take-Home Messages

Rating information is the dominant driver of outcomes; opening context and time controls add measurable but smaller gains. Balanced accuracy is the right headline metric because draws are rare and hardest to classify. For game length, most variance stems from in-game dynamics not observed at prediction time; time controls and rating levels help, but only modestly. PCA offers interpretable context for these patterns rather than direct boosts to supervised performance.

5.3 Methodological Insights

Embedding preprocessing and SMOTE inside cross-validation avoided leakage and produced stable estimates (e.g., RF CV BA $\approx 0.463 \pm 0.014$). Tree-based models captured non-linear interactions among ratings, opening category, and time control better than a linear baseline, while additional tuning gave only marginal changes within expected CV noise. For regression, linear variants performed similarly and underscored the limited linear signal; the small RF improvement aligns with weak non-linear structure. Unsupervised methods clarified structure (ratings/time) but were rightly kept out of supervised features to prevent leakage.

5.4 Future Work

Richer features are the clearest path forward: engine-based position evaluations, move-quality summaries, and player style indicators would target the currently missing in-game dynamics. Sequence-aware models for move data could improve both winner and length prediction. From a validation standpoint, repeated or nested CV and probability calibration checks would refine selection and reporting. Scaling to larger, more diverse datasets and stratifying by rating bands or time-control types could also sharpen insights.

5.5 Final Remarks

This work delivers a leakage-aware, reproducible benchmark for predicting chess outcomes and game length using only information available at the start of play. Results show that moderate winner prediction is achievable from ratings, openings, and time controls, while game-length prediction remains intrinsically harder without in-game detail. The framework and findings offer a solid basis for future extensions with richer features and sequence modeling.

Github Repository

https://github.com/Sandliesigurd/ACIT4510-Statistical-Learning.git

References

1. Mitchell, J. (2017). Chess game dataset (Lichess) [Data set]. Kaggle. https://www.kaggle.com/datasets/datasnaek/chess
2. Fan, Z., Kuang, Y., & Lin, X. (2013). Chess game result prediction system [CS 229 Machine Learning project report]. Stanford University.
3. James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). An introduction to statistical learning: with applications in Python. Springer. https://doi.org/10.1007/978-3-031-38747-0
4. Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2
5. The pandas development team. (2020). pandas-dev/pandas: Pandas (Version 1.x) [Software]. Zenodo. https://doi.org/10.5281/zenodo.3509134

6. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
7. Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. Journal of Machine Learning Research, 18(17), 1–5.
8. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), 785–794. https://doi.org/10.1145/2939672.2939785
9. Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55
10. Waskom, M. L. (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021
11. Virtanen, P., Gommers, R., Oliphant, T. E., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods, 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2