CHESS BLUNDER ANALYSIS USING MULTIVARIATE AND LINEAR ALGEBRAIC APPROACHES

VANDIT GOEL

Abstract. This project provides a quantitative analysis of chess blunders using a large Lichess dataset and Stockfish evaluations. The research will study how temporal factors, positional complexity, piece dynamics, and player skill influence blunder probability, and propose modeling approaches (multivariate analysis, matrix factorization, and neural networks) for prediction.

1. Introduction

This project aims to conduct a quantitative investigation into the nature of blunders in the game of chess. Leveraging the large-scale Lichess open database (casual games, ≈ 90 million games/month) and the TWIC Archive (official international games, ≈ 8100 games/week), the research will analyze a significant dataset of games to identify the conditions under which players are most likely to make critical errors.

A "blunder" will be defined as a move that causes a substantial negative shift in the game's evaluation, as determined by the powerful Stockfish chess engine. The core of the investigation will be to correlate the occurrence of these blunders with a variety of contextual factors, including:

1. **Temporal Factors:** The amount of time remaining on a player's clock and the time spent on the move in question.
2. **Positional Complexity:** The "sharpness" or tactical complexity of the board state. This can be quantified by analyzing the win/draw/loss (WDL) probabilities provided by modern engines like Stockfish.
3. **Piece Dynamics:** The type of piece being moved (e.g., are blunders more common with knights than with rooks?).
4. **Player Skill Level:** How blunder frequency and type differ across various player rating brackets.

The ultimate goal is to move beyond simple blunder identification and develop a model that captures the interplay between these variables. The final phase of the project will involve an attempt to create a predictive neural network model that estimates the probability of a blunder occurring in a given position and context.

2. Problem Formulation

Stockfish reports position quality in *centipawns* (cp), where 100 cp equals a single pawn advantage for the side to move. Traditional search-based evaluations treated this value as a near-linear proxy for material; however, modern NNUE-guided assessments correlate the normalized cp score with the probability of winning the game. A one-pawn lead corresponds to roughly a 50% win probability, while 0 cp implies equal winning chances but an almost forced draw between evenly matched engines. These semantics follow the Stockfish FAQ guidance [12].

When the UCI option `UCI_ShowWDL` is enabled, Stockfish additionally outputs *win/draw/loss* (WDL) probabilities $\left(P_t^{\text{win}}, P_t^{\text{draw}}, P_t^{\text{loss}}\right)$, which satisfy $P_t^{\text{win}} + P_t^{\text{draw}} + P_t^{\text{loss}} = 1$ and summarize the expected match

Date: October 10, 2025.
2010 *Mathematics Subject Classification.* Primary .
Key words and phrases. chess, blunder, Stockfish, machine learning, multivariate analysis.

score. These probabilities come from a model fitted on Fishtest self-play data (Stockfish vs. Stockfish at 60+0.6), so equal-strength opponents yield the published curves, while practical WDL values still depend on opponent strength and time control (draw rates drop in bullet, weaker opposition converts negative cp scores into wins more often). The curves continue to evolve as engines improve, trending toward a 100% draw expectation at 0 cp [13].

With these definitions in place, we analyze every move through the lens of cp evaluations and convert those evaluations into an empirical winning chance. Let c_t denote the Stockfish score after move t (positive for the side to move). Following the Lichess implementation [8], we map c_t to a normalized winning chance

$$W(c_t) = \frac{2}{1+\exp(\alpha c_t)} - 1, \qquad \alpha = -0.00368208. \tag{2.1}$$

The same transformation can be reported as a percentage that is easier to interpret by human players:

$$\mathrm{Win}\%(c_t) = 50 + 50 \cdot W(c_t) = 50 + 50\left(\frac{2}{1+\exp(\alpha c_t)} - 1\right). \tag{2.2}$$

2.1. **Winning-Chance Delta.** Let Δc_t denote the change induced by the move ($\Delta c_t = c_t - c_{t-1}$). We evaluate the impact of a move via the absolute difference between consecutive winning chances:

$$\Delta W_t = |W(c_t) - W(c_{t-1})| = \left|\frac{2}{1+\exp(\alpha c_t)} - \frac{2}{1+\exp\big(\alpha(c_t - \Delta c_t)\big)}\right|. \tag{2.3}$$

The absolute value treats equally sized swings in either direction while keeping the sign information available through Δc_t if further stratification is needed [6, 7].

2.2. **Position Sharpness.** Two positions can be equal from the engine's point of view while having very different practical difficulty. In one equal position no side has meaningful winning chances and both players are expected to hold a draw with accurate play; in another, both sides may have realistic paths to victory and small inaccuracies can immediately tip the result. To quantify how easy a position is to "mess up" for either player, we define a *sharpness* score based on the WDL probabilities.

Given WDL probabilities $\big(P_t^{\mathrm{win}}, P_t^{\mathrm{draw}}, P_t^{\mathrm{loss}}\big)$ for the side to move, the sharpness of move t is defined as

$$S_t = P_t^{\mathrm{win}} + P_t^{\mathrm{loss}}. \tag{2.4}$$

Sharp positions are those in which either player can still win, so S_t increases as the model assigns more probability mass to decisive outcomes and decreases when the draw probability dominates. By construction $0 \le S_t \le 1$, with $S_t \approx 0$ in dead-equal endgames and $S_t \approx 1$ when one side is overwhelmingly winning or losing.

Several alternative sharpness functionals were explored before settling on the simple sum in (2.4). These alternatives attempted to downweight highly decisive positions (where one of P_t^{win} or P_t^{loss} is close to 1) and to emphasize genuinely balanced but tactically sharp states (where both are moderately large). Concretely, we considered the following families, writing $w_t = P_t^{\mathrm{win}}$ and $\ell_t = P_t^{\mathrm{loss}}$ for brevity and introducing a small numerical constant $\varepsilon > 0$ to avoid division by zero and undefined logarithms:

$$S_t^{(1)} = \left(\frac{2}{\log(w_t+\varepsilon) + \log(\ell_t+\varepsilon)}\right)^2, \tag{2.5}$$

$$S_t^{(2)} = \frac{4 w_t \ell_t}{w_t + \ell_t + \varepsilon}, \tag{2.6}$$

$$S_t^{(3)} = (w_t + \ell_t) \cdot H\big(\tilde{w}_t, \tilde{\ell}_t\big), \tag{2.7}$$

where

$$\tilde{w}_t = \frac{w_t}{w_t + \ell_t + \varepsilon}, \qquad \tilde{\ell}_t = \frac{\ell_t}{w_t + \ell_t + \varepsilon},$$

and $H(p,q) = -\big(p\log p + q \log q\big)$ denotes the binary entropy.

In principle these refinements distinguish between positions where both sides have comparable winning chances (near 0.5) and those where only one side can reasonably hope to win, assigning the highest scores to states with $w_t \approx \ell_t \approx 0.5$ and penalizing positions where one of them is close to 0 or 1.

In practice, however, the sample we analyze contains almost no positions where w_t and ℓ_t are simultaneously close to 0.5, so the more elaborate constructions offered little empirical benefit and added unnecessary complexity. Moreover, they were numerically fragile in the regimes that actually occur in our data: when either w_t or ℓ_t is very close to 0 or 1, the underlying expressions suffer from division-by-zero and undefined logarithms, and the required clipping with ε can drive the resulting scores artificially toward 0 even in positions that are intuitively sharp. For this reason we adopt the simple sharpness score (2.4) — effectively "winning chance + losing chance" — and reserve richer functionals for future work on datasets that contain a broader mix of extremely balanced yet tactically volatile positions.

2.3. **Error Taxonomy.** Moves are categorized solely by the magnitude of ΔW_t. Define thresholds

$$\tau_B = 0.30, \qquad \tau_M = 0.20, \qquad \tau_I = 0.10,$$

ordered so that $\tau_B > \tau_M > \tau_I$. The judgement assigned to move t is

$$\text{Judgement}_t = \begin{cases} \text{Blunder}, & \Delta W_t \geq \tau_B, \\ \text{Mistake}, & \tau_M \leq \Delta W_t < \tau_B, \\ \text{Inaccuracy}, & \tau_I \leq \Delta W_t < \tau_M, \\ \text{None}, & \Delta W_t < \tau_I. \end{cases} \tag{2.8}$$

No additional modeling assumptions are required at this stage; the labels arise purely from the calibrated winning-chance deltas. Because the new slope α was tuned to align with human outcomes, we retain the existing accuracy formula for downstream evaluation metrics.

3. Methodology

3.1. **Data Extraction from Lichess PGN Datasets.** The Lichess PGN archives were converted into an analysis-ready dataset via a two-stage extraction pipeline that prioritizes sequential I/O efficiency and parallelism. In the first stage the corpus is scanned in a single pass to record the byte offsets of individual games; these offsets permit deterministic, non-overlapping assignment of disjoint batches to worker processes so that each portion of the file is read exactly once. In the second stage multiple workers read games by offset, parse headers and move lists into structured records, and extract game-level metadata (for example, time control, termination mode, and Elo ratings). For each ply we collect engine-derived evaluations (centipawn values, win/draw/loss probabilities, and mate distances) through a controlled evaluation interface.

Per-move records are written to a columnar store (Parquet) with a fixed schema so that each row corresponds to a single move event and columns encode numerical and categorical descriptors. This storage choice supports linear-time streaming aggregation and efficient matrix-style batch loading for downstream statistical and machine-learning pipelines. The pipeline design enforces reproducibility (via deterministic offset partitioning), minimizes redundant I/O (single-pass discovery plus parallel readers), and maintains a clear, auditable mapping from raw PGN to analytic features suitable for vectorized analysis.

3.2. **Feature Engineering Using Evaluation Data and Time Metrics.** Let $x_t \in \mathbb{R}^d$ denote the feature vector extracted for move t. Each per-move record produced by the extractor contains a set of raw fields together with a small number of derived quantities; these form the building blocks for downstream matrices and models. Below we list the record fields, their interpretation, and the calculations used to produce derived features.

- **Identifiers and provenance:** the fields `game_id`, `offset`, and `site` uniquely identify the game and the byte position within the original archive. These fields enable reproducible joins and deterministic partitioning of the corpus.

- **Player ratings:** the player's Elo values are recorded as integer fields for White and Black. The reported rating changes are stored separately. A simple derived scalar is the rating differential
$$\Delta\text{Elo} = \texttt{white\_elo} - \texttt{black\_elo}.$$
- **Opening and game descriptors:** `eco` is the ECO opening code; `time_control`, `game_time`, and `increment` encode the time-control category and parameters. The categorical time-control is used to group and standardize temporal features.
- **Game outcome and termination:** `result` is encoded as +1 (White win), 0 (draw), and -1 (Black win). `termination` records the termination mode (resignation, mate, timeout, etc.) as a categorical string.
- **Move context:** `turn` (which side moves), `move` (SAN move string), and `fullmove_number` give the ply context required for sequential analyses.
- **Engine-based evaluations:**
  - `cp_score`: a centipawn-style score (White-perspective). Mate scores are converted to large numeric sentinels so the field remains numeric for vectorized pipelines.
  - `winning_chance`, `drawing_chance`, `losing_chance`: probability estimates derived from the engine's WDL output (in $[0, 1]$).
  - `mate_in`: mate distance when applicable; otherwise set to $+\infty$.
  - `sharpness`: a scalar derived from WDL probabilities that quantifies how easy the position is to "mess up" for either player, defined as $S_t = P_t^{\text{win}} + P_t^{\text{loss}}$ as in Section 2.
- **Board and tactical context:** `piece_moved` is the moved piece type (categorical), `board_fen` is the FEN of the position after the ply, and `is_check` is a Boolean indicating whether the side to move is in check.
- **Temporal information:** `clock` records the player's remaining time (seconds) at the moment of the move when available; missing clocks are encoded as null. In addition to raw clocks we compute a normalized *time_ratio* feature: for each move we look at the previous clock value (including increment) and measure the fraction of available thinking time consumed by the current move,
$$\text{time\_ratio}_t = \frac{(\text{clock}_{t-1} + \text{increment}) - \text{clock}_t}{\text{clock}_{t-1} + \text{increment}},$$
with null ratios set to zero. This ratio absorbs both the base time and increment structure, yielding a comparable scale across formats.
- **Local evaluation change:** the primary short-term derived measure is the per-ply evaluation change
$$\Delta\text{eval}_t = \texttt{cp\_score}_t - \texttt{cp\_score}_{t-1},$$
computed as the difference between the engine score after the current ply and the engine score before the same ply. Positive values indicate an advantage shift for White; negative values indicate deterioration. When the previous score is unavailable (e.g., initial position), the change is computed with respect to the starting-position score.

From the raw and derived fields we construct analysis-ready features. Typical choices used in this work include:

- Intercept: a constant 1 to absorb baseline effects in linear models.
- Evaluation features: `cp_score` and Δeval_t, optionally winsorized to limit the influence of mate sentinels.
- Probabilistic features: `winning_chance`, `drawing_chance`, and `losing_chance` (optionally transformed via log-odds for linear models), together with the sharpness score $S_t = P_t^{\text{win}} + P_t^{\text{loss}}$.
- Mate indicators: binary indicator $\mathbf{1}[\texttt{mate\_in} < \infty]$ and the mate distance when present.
- Temporal features: standardized remaining clock (within-game z-score), the normalized time-ratio described above, and an indicator or ordinal encoding of time-control category.
- Categorical encodings: one-hot or embedding encodings of `piece_moved`, `eco`, and `termination`.
- Rating-based controls: the rating differential ΔElo and raw player ratings.

These raw and derived features are persisted in columnar form so that the final design matrix $X \in \mathbb{R}^{n \times d}$ (with rows ordered by move-event) can be loaded without further parsing.

3.2.1. *Feature Preprocessing and Transformations.* Before applying dimensionality reduction and modeling techniques, several preprocessing transformations are applied to the raw feature set:

1. **Categorical encoding:** Boolean features (`is_check`) are cast to integer type (0/1). Categorical features `time_control_type` and `piece_type` are one-hot encoded, producing binary indicator columns for each category value. The `UNKNOWN` category is dropped as it provides no discriminative information.
2. **Target variable construction:** The response variable `is_error` is created as a binary indicator:
$$\texttt{is\_error}_t = \mathbf{1}[\texttt{judgement}_t \in \{\text{Blunder}, \text{Mistake}, \text{Inaccuracy}\}],$$
with null judgements (moves with no evaluation delta) mapped to 0 (no error).
3. **Evaluation score scaling:** The centipawn score `cp_score` exhibits extreme values in mate positions (set to large sentinels such as ± 10000). To prevent these outliers from dominating distance-based analyses, we apply a bounded transformation:
$$\texttt{cp\_score}_{\text{scaled}} = \begin{cases} \tanh\left(\frac{\texttt{cp\_score}}{2019}\right) \cdot 0.99 & \text{if } \texttt{mate\_in} = 0, \\ \text{sign}(\texttt{cp\_score}) & \text{if } \texttt{mate\_in} \neq 0. \end{cases}$$
The hyperbolic tangent squashes non-mate evaluations into the interval $(-0.99, 0.99)$, preserving order while bounding magnitude. Positions with forced mate are mapped to ± 1 to mark them as decisively winning or losing without introducing unbounded variance. The divisor 2019 was chosen empirically to yield smooth compression across typical centipawn ranges.
4. **Missing value imputation:** Infinite values in `mate_in` (indicating no forced mate) are replaced with null, then filled with 0. Similarly, any remaining null values in `time_ratio` are set to 0, representing moves with no recorded time consumption.
5. **Standardization:** Each numeric feature column f (excluding the binary target `is_error`) is z-score standardized:
$$f_{\text{std}} = \frac{f - \mu_f}{\sigma_f},$$
where μ_f and σ_f are the mean and standard deviation computed over all moves. This transformation centers each feature at zero and scales to unit variance, ensuring that features with different native ranges (e.g., `player_elo` in $[1000, 3000]$ versus `time_ratio` in $[0, 1]$) contribute comparably to subsequent linear algebraic operations such as PCA.

This preprocessing pipeline produces a standardized, numeric-only feature matrix suitable for least-squares regression, principal component analysis, and classification models, while maintaining clear provenance from the original per-move records to the numeric arrays used in the mathematical analyses.

3.3. **Board Position Encoding: FEN to Bitboard Representation.** While the engine evaluation metrics (centipawn scores, WDL probabilities) provide global assessments of position quality, many of the structural and tactical patterns underlying blunders require direct access to the spatial configuration of pieces on the board. The standard representation for chess positions is the Forsyth-Edwards Notation (FEN), a compact string encoding the placement of all pieces, active side, castling rights, en passant targets, halfmove clock, and fullmove number. For example, the starting position is encoded as:

```
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
```

where lowercase letters denote black pieces, uppercase letters denote white pieces, digits indicate consecutive empty squares, and slashes separate ranks.

However, FEN strings are not directly amenable to linear algebraic operations or gradient-based learning algorithms. To enable matrix factorization techniques (PCA, SVD) and eventual neural network modeling, we convert each board position into a *bitboard tensor*—a fixed-size binary representation that exposes the spatial structure of the position in a format compatible with standard numerical analysis tools.

3.3.1. *Bitboard Tensor Construction.* Each chess position is encoded as a $12 \times 8 \times 8$ binary tensor, where the first dimension indexes piece type and color:

$$\mathbf{B} \in \{0,1\}^{12\times 8\times 8}.$$

The 12 channels correspond to the six piece types (pawn, knight, bishop, rook, queen, king) for each color (white, black), yielding planes:

$$\{\text{WP, WN, WB, WR, WQ, WK, BP, BN, BB, BR, BQ, BK}\}.$$

For each plane p and square (i, j) with $i, j \in \{0, \ldots, 7\}$ (representing rank and file), the entry $\mathbf{B}_{p,i,j}$ is set to 1 if the corresponding piece occupies that square, and 0 otherwise. This representation is standard in modern chess engines and neural network architectures (e.g., AlphaZero [11]).

3.3.2. *Flattening for Linear Algebra.* For compatibility with dimensionality reduction techniques such as PCA and SVD, the $12 \times 8 \times 8$ tensor is flattened into a single 768-dimensional binary vector:

$$\mathbf{b} = \text{vec}(\mathbf{B}) \in \{0,1\}^{768}.$$

This flattening operation concatenates all 12 planes in a fixed order, producing a canonical vectorized representation of the position. Each move in the dataset is thus associated with a feature vector $\mathbf{x}_t \in \mathbb{R}^d$ that combines:

- the 768-dimensional bitboard encoding $\mathbf{b}_t$,
- scalar evaluation and temporal features (centipawn score, clock, time ratio, sharpness),
- categorical encodings (piece moved, opening, time control).

The full design matrix $X \in \mathbb{R}^{n\times d}$ therefore incorporates both positional structure (via bitboards) and contextual metadata (via derived scalars), enabling joint analysis of spatial patterns and external factors.

Converting FEN strings to bitboard tensors transforms a symbolic, human-readable representation into a numerical format suited for rigorous mathematical analysis. This encoding bridges classical statistical methods (multivariate regression, PCA) and modern machine learning techniques (neural networks, transformers), forming the foundation for all subsequent modeling and interpretation tasks in this research.

Note on dimensionality constraints: The 768-dimensional bitboard encoding $\mathbf{b}_t$ was excluded from the initial PCA experiments reported in Section 4 due to memory limitations. The full dataset (approximately 10^8 move records) combined with high-dimensional spatial features exceeds available RAM for batch matrix operations. Future work will address this constraint through: (i) streaming PCA implementations that process mini-batches sequentially without materializing the full covariance matrix [9], (ii) sparse representations that exploit the typical occupancy of ≈ 32 squares per position, or (iii) convolutional autoencoders pretrained on position embeddings to reduce the spatial feature dimension before joint PCA. The analyses in this paper focus on the lower-dimensional evaluation, temporal, and categorical features, which remain highly informative for blunder prediction while fitting comfortably in memory.

3.4. **Dimensionality Reduction via Incremental PCA.** Principal Component Analysis (PCA) is a fundamental tool for identifying directions of maximum variance in high-dimensional data and for reducing feature dimensionality while retaining explanatory power. Standard batch PCA requires constructing the full covariance matrix $C = \frac{1}{n}X^T X$ where $X \in \mathbb{R}^{n\times d}$ is the centered data matrix, then computing its eigendecomposition. For our datasets, this computation exceeds available memory.

To address this constraint, we employ *Incremental PCA* (IPCA) [9], an online algorithm that processes the data in sequential mini-batches and updates the principal component estimates incrementally. The IPCA algorithm maintains a running estimate of the covariance matrix and its eigendecomposition without requiring the entire dataset to reside in memory simultaneously. Formally, IPCA is equivalent to batch PCA in the sense that given the same dataset, the final transformation coefficients are identical; the difference lies solely in the computational strategy.

3.4.1. *Implementation Details.* In our experiments we selected $k = 10$ principal components and processed batches of 10,000 move records at a time. The features included in the analysis comprised:

- Temporal features: `game_time`, `increment`, `time_ratio`.
- Evaluation features: scaled `cp_score`, `winning_chance`, `drawing_chance`, `losing_chance`, `eval_delta`.
- Positional context: `turn`, `is_check`, `mate_in`.
- Player skill: `player_elo`.
- One-hot encoded categories: time control types (Blitz, Bullet, Rapid, Standard, Unlimited) and piece types (Pawn, Knight, Bishop, Rook, Queen, King).

After preprocessing, standardization, and one-hot expansion, the feature dimension was $d = 23$.

The target variable `is_error` was excluded from the PCA fitting but retained alongside the projected features for subsequent supervised modeling. This separation ensures that the principal components reflect the intrinsic structure of the input features without contamination from the response variable.

The use of IPCA enables scalable analysis of datasets that far exceed available memory, making it feasible to extract low-dimensional representations from massive corpora while maintaining mathematical equivalence to classical batch PCA. The resulting principal components serve as input features for downstream regression and classification models aimed at blunder prediction.

4. Results

This section presents empirical findings from the processed Lichess sample. Unless stated otherwise, an "error" denotes any move classified as Inaccuracy, Mistake, or Blunder via the winning chance delta thresholds (Section 2).

4.1. **Error Rates Across Elo and Time Controls.** Error prevalence generally declines with increasing Elo across time controls (Figures 1b–1d), but the pattern of decline depends strongly on rating strata and time control. The steepest slope of decline occurs among the very highest-rated players (Elo > 2600), who show marked reductions in error rates relative to slightly lower tiers. Longer time formats (Rapid and Standard) exhibit a larger decline in errors across Elo than Bullet; Bullet still shows a noticeable improvement for the top (> 2600) players but the gradient is less pronounced than in longer formats.

Key observations:

- **Bullet (Figure 1a)**: Error proportions are elevated at lower Elo and improve with rating, suggesting pattern recognition and practical skills partially compensate for extreme time pressure.
- **Blitz (Figure 1b)**: Shows a sharper decline across Elo than Bullet for higher rating bands, reflecting that modestly longer decision time reduces time-induced noise.
- **Rapid (Figure 1c)**: Transitional regime; error rates fall more substantially with Elo than in Bullet/Blitz with higher-rated players.
- **Standard (Figure 1d)**: Distinct profile: higher-rated players (Elo ¿2400) make significantly fewer errors, while lower-rated players continue to exhibit relatively high error rates, producing a more bimodal-like relationship between Elo and error prevalence.

4.2. **Error Rates as a Function of Sharpness.** Figure 3 summarizes how the empirical proportion of moves labeled as Inaccuracy, Mistake, or Blunder varies with the sharpness score $S_t = P_t^{\text{win}} + P_t^{\text{loss}}$ defined in Section 2. Recall that S_t is close to 0 in positions with essentially no winning chances for either side and approaches 1 when the engine believes a decisive result is likely for one player.

Across most of the range, higher sharpness is associated with a higher fraction of moves being flagged as errors. All three curves (Inaccuracy, Mistake, Blunder) exhibit an overall upward trend from low S_t toward the mid-to-high range, consistent with the interpretation of sharp positions as "easy to mess up" for either

side. Positions where both sides still have realistic winning chances—reflected by substantial mass in P_t^{win} and/or P_t^{loss}—lead to more frequent deviations from engine-best play.

The effect is strongest for inaccuracies: their proportion rises steeply as S_t increases from very low values and peaks around intermediate-to-high sharpness, suggesting that players often make small but non-fatal errors in tactically volatile positions. Mistakes and blunders show a similar but less steep monotone increase, indicating that severe errors also become more common as positions become sharper, though the absolute rates remain lower than for inaccuracies.

At the extreme ends of the sharpness scale the curves drop, but with an important nuance at the low end. All three series exhibit a dip in error proportion around $S_t \approx 0.1$, where positions are low-sharpness and players rarely commit large mistakes. In contrast, the blunder curve spikes very close to $S_t = 0$, suggesting that some moves made in objectively "dead" or trivially equal positions receive almost no attention and are played carelessly, leading to disproportionately many catastrophic errors. The drop near $S_t \approx 1$ is less structural and should be interpreted cautiously. Very sharp, almost-decided positions are rare in the sample, so the corresponding empirical proportions are based on few observations and are sensitive to binning choices; in addition, once a position is nearly winning or losing, many moves may be equally decisive in the engine's view, which can dampen measured error rates despite high nominal sharpness.

4.3. **Move-level time usage (time_ratio).** The per-move time usage is summarized by the ratio

$$\text{time\_ratio}_t = \frac{(\text{clock}_{t-1} + \text{increment}) - \text{clock}_t}{\text{clock}_{t-1} + \text{increment}},$$

which measures the fraction of the available clock at move t that was consumed making that move. Values near 0 indicate very quick moves (including pre-moves or routine, low-effort moves), while values near 1 indicate that almost the entire available time allotment was used.

Empirically (Figure 2), the proportion of moves classified as errors increases with 'time_ratio' across all fast time controls. Key patterns:

- **Monotonic increase:** Error proportion rises from low 'time_ratio' (quick moves) up through intermediate values, with a clear upward trend for Bullet and Blitz and a milder slope for Rapid. This suggests that moves made after spending a larger share of available time are, on average, more likely to be errors than very quick, routine moves.
- **Stronger effect in faster controls:** The steepest increase appears in Bullet and Blitz, consistent with time-pressure amplifying decision noise. Rapid shows a similar shape but lower overall error proportions.
- **Extreme-value volatility:** At very high 'time_ratio' (close to 1.0) the series becomes noisy, with occasional spikes and abrupt drops. Those features are at least partly sampling noise: very few moves consume essentially the entire allotment in some bins, producing high variance in the empirical proportion.
- **Interpretation hypotheses:** Low 'time_ratio' often corresponds to low-complexity positions or automatic pattern play (hence low errors). Larger 'time_ratio' can represent either (i) genuinely harder positions requiring long calculation (increasing error risk despite more time), or (ii) extreme time pressure where players are forced to use most of their remaining clock repeatedly and still make mistakes.

For modeling, 'time_ratio' should be included as a non-linear predictor (e.g., spline or low-order polynomial) and interacted with time-control category and Elo. Additionally, control for move phase and position-sharpness metrics to disambiguate whether high 'time_ratio' reflects complexity or time-scramble. Finally, use binning or variance-aware smoothing and report counts per bin to avoid over-interpreting noisy extremes.

4.4. **Piece-Type Error Distribution.** Figure 4 reports error composition by moved piece. Knights and Queens show relatively higher severe error proportions compared to Pawns. Hypotheses:

- **Knights**: High branching tactical motifs; miscalculation of knight forks and intermediate squares leads to larger evaluation drops.
- **Queens**: Centralization and overextension errors produce large tactical liabilities when tempo is misjudged.
- **Pawns**: Majority of pawn moves produce incremental positional shifts, hence lower severe error proportion; inaccuracies dominate rather than blunders.
- **Rooks/Bishops**: Intermediate; their long-range nature creates both strategic and tactical opportunities but fewer catastrophic single-move losses than queen blunders.

4.5. **Principal Component Analysis of Feature Space.** To understand the intrinsic dimensionality of the feature space and identify dominant patterns of variation, we applied Incremental PCA (Section 3.4) to the standardized feature matrix excluding the target variable `is_error`. The analysis used 10 principal components and processed $12,540,813$ move records in batches of 10,000 observations.

4.5.1. *Explained Variance.* Table 1 reports the explained variance ratio for each of the 10 principal components. The first component captures 10.73% of total variance, the second 8.29%, and subsequent components capture progressively smaller fractions. The cumulative explained variance reaches approximately 62.3% after 10 components.

Component	Variance Ratio	Cumulative Variance
PC1	0.1073	0.1073
PC2	0.0829	0.1902
PC3	0.0755	0.2657
PC4	0.0572	0.3229
PC5	0.0566	0.3795
PC6	0.0550	0.4345
PC7	0.0519	0.4864
PC8	0.0502	0.5365
PC9	0.0435	0.5801
PC10	0.0430	0.6231

TABLE 1. Explained variance ratios for the first 10 principal components.

4.5.2. *Interpretation of Low Explained Variance.* The relatively low individual and cumulative explained variance ratios indicate that the feature space does not exhibit strong low-rank structure. Several factors contribute to this finding:

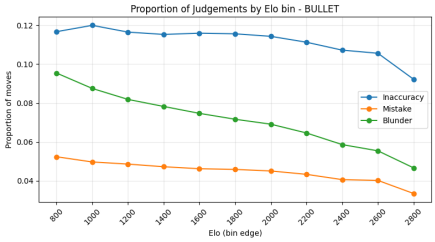
- **Feature heterogeneity:** The feature set combines diverse modalities—temporal metrics (`game_time`, `time_ratio`), evaluation scores (`cp_score`, WDL probabilities), categorical encodings (time control, piece type), and player skill (`player_elo`). These features capture fundamentally different aspects of the game state and player behavior, and there is no reason to expect them to align along a small number of shared axes.
- **High intrinsic dimensionality:** Chess positions and move contexts are inherently high-dimensional phenomena. The strategic, tactical, temporal, and psychological factors that determine blunder likelihood operate on distinct, weakly correlated dimensions. Standard PCA seeks linear combinations that maximize variance, but the absence of dominant global directions suggests that blunder patterns are better described by localized, non-linear manifold structure or feature interactions rather than simple linear projections.
- **Exclusion of positional features:** The 768-dimensional bitboard encoding was excluded from this analysis due to memory constraints (Section 3.3). Board structure likely contains rich, compressible spatial patterns (e.g., pawn chains, king safety configurations) that could exhibit stronger low-rank structure. The current analysis is limited to evaluation-derived and metadata features, which are more abstract and less redundant than raw piece placements.

- **Noise and stochasticity:** Human move choices contain irreducible noise from cognitive limitations, time pressure, and psychological factors. This noise inflates the effective dimensionality and prevents variance from concentrating in a few components.

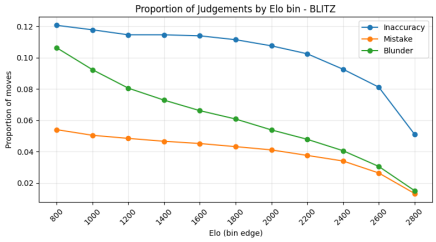
Despite the modest explained variance, the PCA projection remains useful for several purposes:

1. **Decorrelation:** The principal components are orthogonal by construction, eliminating multicollinearity for downstream linear models.
2. **Baseline for comparison:** The low explained variance establishes that simple linear dimensionality reduction is insufficient for this task, motivating more sophisticated approaches such as autoencoders, kernel PCA, or direct supervised feature learning via neural networks.

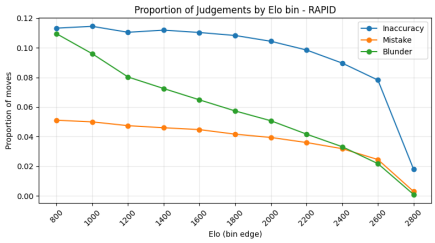
4.5.3. *Scatter Plot of First Two Components.* Figure 5 shows a scatter plot of the first two principal components, with points colored by the binary `is_error` label. The two classes (error vs non-error) overlap substantially in the PC1–PC2 plane, confirming that linear projections do not achieve clean separation. This overlap is consistent with the low explained variance and suggests that blunder prediction will require non-linear classifiers or richer feature representations (e.g., convolutional embeddings of board positions) to capture the complex decision boundaries separating errors from correct moves.



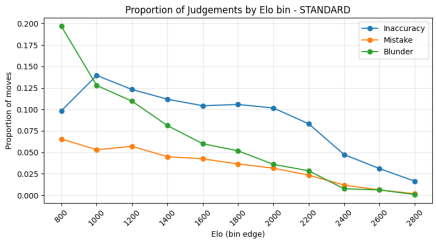
(A) Bullet



(B) Blitz

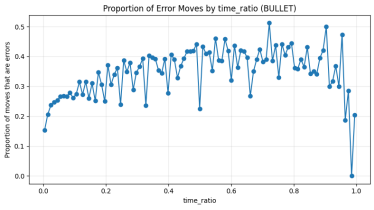


(C) Rapid

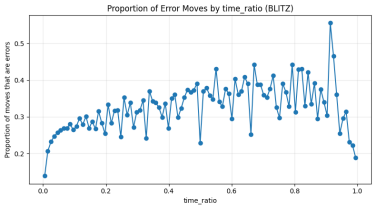


(D) Standard

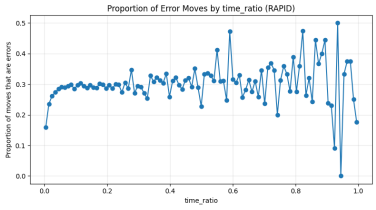
FIGURE 1. Elo-binned error proportions by time control: Bullet, Blitz, Rapid, and Standard.



(A) Bullet



(B) Blitz



(C) Rapid

FIGURE 2. Proportion of moves classified as errors by 'time_ratio' for Bullet, Blitz, and Rapid time controls.

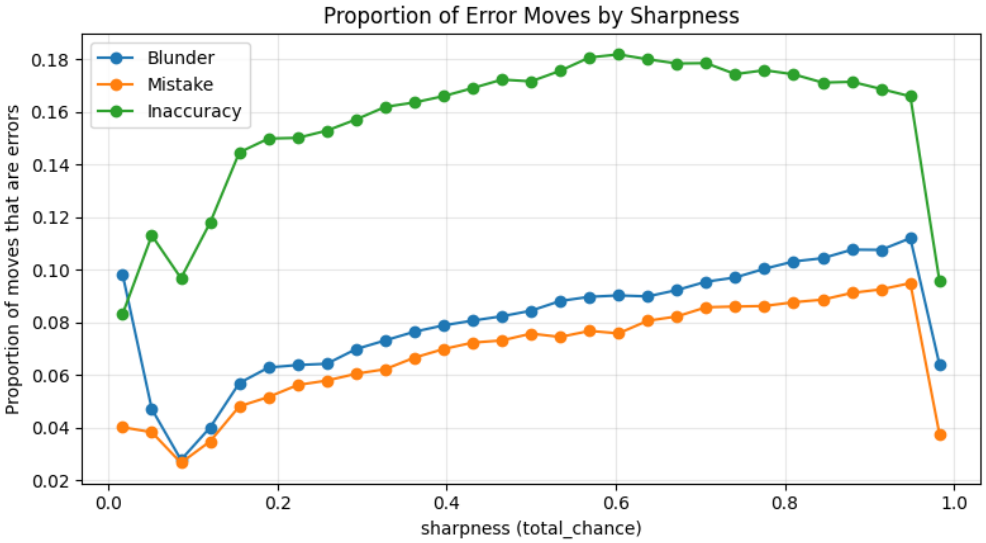


FIGURE 3. Proportion of moves classified as Inaccuracy, Mistake, or Blunder by sharpness score $S_t = P_t^{\text{win}} + P_t^{\text{loss}}$.

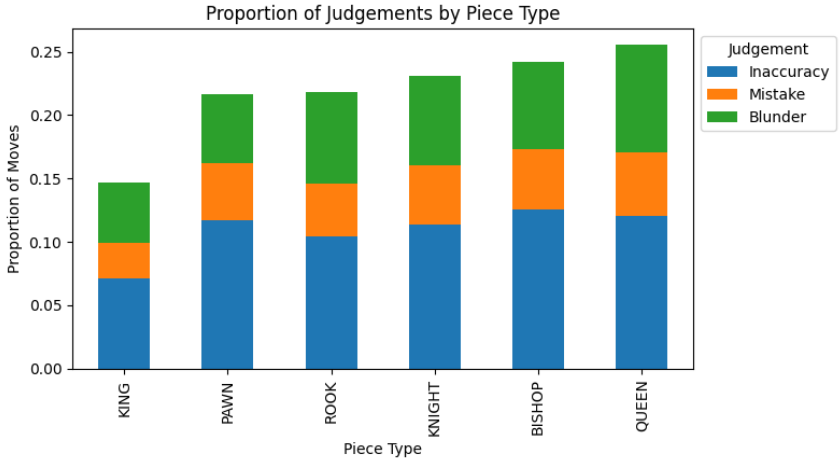


FIGURE 4. Error composition by moved piece type.

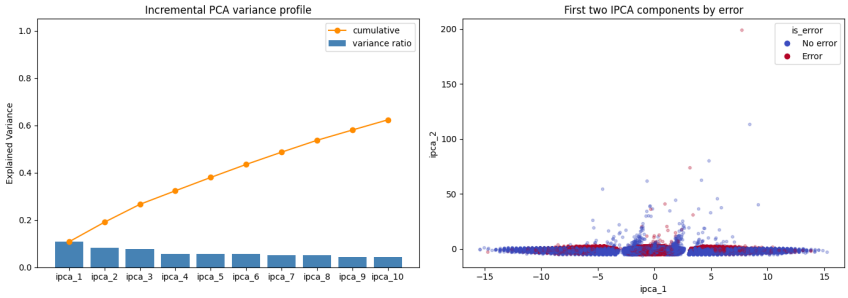


FIGURE 5. Principal Component Analysis.

5. Conclusions and Future Work

This research has presented a comprehensive quantitative analysis of chess blunders using large-scale game data from Lichess, Stockfish engine evaluations, and multivariate statistical methods. The principal findings establish clear empirical relationships between blunder rates and player skill, time pressure, positional complexity, and piece dynamics, while also revealing the limitations of linear dimensionality reduction for this high-dimensional, heterogeneous feature space.

5.1. Key Findings.

1. **Time pressure and blunder rates:** Error proportions increase monotonically with the fraction of available time consumed per move (`time_ratio`), with the strongest effects observed in faster time controls (Bullet and Blitz). This pattern suggests that high `time_ratio` correlates with either genuinely difficult positions requiring extensive calculation or extreme time scrambles where cognitive resources are exhausted.
2. **Positional sharpness amplifies errors:** Positions with high sharpness scores $S_t = P_t^{\text{win}} + P_t^{\text{loss}}$ exhibit elevated error rates across all severity categories (Inaccuracy, Mistake, Blunder). The relationship is approximately monotonic, confirming the hypothesis that tactically volatile positions are inherently "easier to mess up" for human players.
3. **Skill-dependent error profiles:** Error rates decline with increasing player Elo, but the slope of decline varies by time control. The sharpest improvement occurs among the highest-rated players (Elo > 2600), particularly in longer time formats (Rapid and Standard), suggesting that elite players leverage additional thinking time more effectively than lower-rated players.
4. **Piece-specific blunder patterns:** Knights and Queens exhibit higher proportions of severe errors (Mistakes and Blunders) compared to Pawns, Rooks, and Bishops. This finding is consistent with the tactical complexity of knight moves (non-linear movement patterns, fork motifs) and the high-value, tempo-sensitive nature of queen maneuvers.
5. **Low explained variance in PCA:** The first 10 principal components of the standardized feature matrix capture only 62.3% of total variance, with the leading component explaining just 10.73%. This low concentration of variance indicates that the feature space does not exhibit strong low-rank structure. The heterogeneity of features (temporal, evaluation-based, categorical, skill-based) and the exclusion of high-dimensional positional encodings contribute to this finding. The result suggests that simple linear projections are insufficient for blunder prediction and that non-linear models or richer representations will be necessary.

5.2. Limitations and Open Questions.

Several limitations of the current analysis point toward important directions for future investigation:

- **Exclusion of positional structure:** The 768-dimensional FEN tensor encoding was excluded from PCA due to memory constraints. Board structure encodes critical spatial patterns (e.g., king safety, pawn chains, piece coordination) that likely exhibit compressible low-rank structure and contribute directly to blunder risk. Integrating positional features into the dimensionality reduction pipeline is a high priority.
- **Static feature representation:** The current feature set treats each move in isolation, ignoring the sequential and temporal dependencies inherent in chess games. Blunder likelihood depends not only on the current position but also on the trajectory of recent moves, the opponent's style, and cumulative time pressure over the course of the game.
- **Linear modeling assumptions:** PCA and standard multivariate regression assume linear relationships among features. However, the interactions between time pressure, sharpness, and skill are likely non-linear and context-dependent. The low explained variance in PCA and the substantial class overlap in low-dimensional projections confirm that linear models are inadequate for this task.
- **Thresholding and discretization:** The binary `is_error` target aggregates Inaccuracies, Mistakes, and Blunders into a single category. This coarse-graining discards information about error severity and may obscure distinct causal mechanisms underlying minor versus catastrophic mistakes.

5.3. **Next Steps: Sequential Modeling and Blunder Prediction.** The analyses presented in this paper serve as a foundation for the next phase of research: developing predictive models for blunder occurrence and severity. The following approaches are planned:

1. **Recurrent neural networks (RNNs) and transformers:** To capture sequential dependencies and temporal dynamics, we will train recurrent architectures (LSTMs, GRUs) or transformer-based models on move sequences. These models will take as input the history of moves, evaluations, and time usage within a game and predict the probability of an error on the next move. Attention mechanisms in transformers can identify which prior moves or positions are most predictive of subsequent blunders, enabling interpretable analysis of error cascades and momentum shifts.
2. **Convolutional autoencoders for positional embeddings:** To incorporate the 768-dimensional FEN tensor while managing memory constraints, we will pretrain a convolutional autoencoder on board positions. The encoder will compress each position into a lower-dimensional latent representation (e.g., 32 or 64 dimensions) that preserves spatial structure and piece relationships. These embeddings can then be concatenated with evaluation and temporal features and used as input to sequential models, enabling joint modeling of positional and contextual factors.
3. **Multi-task learning:** Instead of predicting a binary error indicator, we will train models with multiple output heads: one for error occurrence (classification), one for error severity (ordinal regression: None, Inaccuracy, Mistake, Blunder). Multi-task learning can improve generalization by forcing the model to learn shared representations that capture the underlying structure of chess positions and move quality.
4. **Player-specific and Elo-conditioned models:** Blunder patterns vary significantly across skill levels (Section 4). We will incorporate player Elo as a conditioning variable in neural network architectures (e.g., via an embedding layer or feature-wise linear modulation) to learn skill-dependent error profiles. This approach will enable the model to adapt its predictions to the cognitive capabilities and typical mistake patterns of different rating bands.
5. **Incremental PCA with positional features:** As a stepping stone toward full neural modeling, we will extend the Incremental PCA framework [9] to include the FEN tensor by processing mini-batches sequentially and maintaining a low-rank approximation of the covariance matrix. Alternatively, we will explore sparse PCA or randomized SVD methods that exploit the sparsity of board occupancy (typically ≈ 32 out of 64 squares) to reduce computational cost.
6. **Interpretability and feature importance:** To understand which features and interactions drive blunder predictions, we will apply interpretability techniques such as SHAP (SHapley Additive exPlanations) values, integrated gradients, and attention weight visualization. These tools will reveal whether the model relies primarily on evaluation deltas, time pressure, piece configurations, or higher-order interactions, providing actionable insights into the cognitive and strategic factors underlying human errors.

5.4. **Broader Implications.** Beyond technical advances in predictive modeling, this research has implications for chess training, cognitive psychology, and human-computer interaction:

- **Adaptive training systems:** Predictive models that estimate blunder risk in real time can be integrated into chess training platforms to provide personalized feedback. For example, a system could flag positions where the player is statistically likely to err (due to high sharpness or time pressure) and suggest additional practice or slower play.
- **Cognitive load assessment:** The relationship between time pressure, sharpness, and error rates offers a window into human decision-making under resource constraints. The findings may generalize to other domains where time-limited choices under uncertainty lead to systematic errors (e.g., medical diagnosis, financial trading, cybersecurity incident response).
- **Fair play and cheat detection:** Understanding the typical distribution of blunder rates and patterns for a given skill level can inform statistical tests for anomalous play. Deviations from expected error profiles (e.g., suspiciously low blunder rates in high-complexity positions) may indicate computer assistance or other forms of cheating.

5.5. **Closing Remarks.** This project has demonstrated that large-scale empirical analysis of chess games, powered by modern engine evaluations and incremental linear algebraic methods, can yield quantitative insights into the structure and predictability of human errors. The low explained variance in PCA underscores the complexity and high dimensionality of the problem, motivating the transition to sequential neural models that can capture temporal dependencies, non-linear interactions, and rich positional structure.

The next phase of research will build on these foundations to develop end-to-end predictive systems for blunder detection and severity estimation. By integrating sequential architectures, convolutional positional embeddings, and skill-conditioned modeling, we aim to construct interpretable, actionable models that advance both our understanding of human chess performance and the broader science of decision-making under pressure.

Appendix A. Source Code

The complete implementation, including PGN parsing, game evaluation, feature engineering, and analysis notebooks, is available in our GitHub repository.[1]

Appendix B. Dataset Statistics

Metric	**Value**
Total games	187575
Total moves analyzed	12540813
Moves with judgements	2748316
Blunders	815201
Mistakes	546066
Inaccuracies	1387049
Time control distribution:	
Bullet	38 %
Blitz	48 %
Rapid	13 %
Standard	0.4 %
Elo range	400–3288
Mean player Elo	1707

Table 2. Summary statistics of the analyzed dataset.

[1] https://github.com/vandyG/blunder-analysis/tree/master/notebooks

References

[1] Vandit Goel. Predicting level of game play (chess) using transformers. Unpublished manuscript. Available as local file: Supplement1.pdf, 2025.

[2] Matej Guid and Ivan Bratko. Computer analysis of world chess champions. *ICGA Journal*, 29:65–73, 06 2006.

[3] jk_182. Evaluating sharpness using lc0's wdl, Apr 2024. https://lichess.org/@/jk_182/blog/evaluating-sharpness-using-lc0s-wdl/EXZ3pRoy.

[4] jk_182. Using lc0's wdl to analyze games, March 2024. https://lichess.org/@/jk_182/blog/using-lc0s-wdl-to-analyze-games/spKmwjw5.

[5] jk_182. How does the clock impact the rate of mistakes?, Oct 2025. https://lichess.org/@/jk_182/blog/how-does-the-clock-impact-the-rate-of-mistakes/JSazQplM.

[6] Lichess. Lichess judgement, March 2024. https://lichess.org/page/accuracy#first-compute-win.

[7] Lichess. Lichess judgement, March 2024. https://github.com/lichess-org/lila/blob/9dec42fe22264b6c57a747e441ef33e50f6ed395/modules/tree/src/main/Advice.scala#L58.

[8] Lichess. Lichess pr 11148, March 2024. https://github.com/lichess-org/lila/pull/11148.

[9] Vittorio Lippi and Giacomo Ceccarelli. Incremental principal component analysis: Exact implementation and continuity corrections. *ICINCO 2019 - Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics*, 1:473–480, 2019.

[10] Reid McIlroy Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery &; Data Mining*, KDD '20, pages 1677–1687. ACM, 09 2020.

[11] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[12] Stockfish. Stockfish faq, March 2024. https://official-stockfish.github.io/docs/stockfish-wiki/Stockfish-FAQ.html#centipawns.

[13] Stockfish. Stockfish wdl model, March 2024. https://github.com/official-stockfish/WDL_model?tab=readme-ov-file#background.

University of Texas at Arlington, 701 S. Nedderman Dr., Arlington, TX 76019

Email address: vxg5699@mavs.uta.edu