How good is a chess player?

Final Project Report

Abstract— **This purpose of this project is to analyze the game of chess, based on the data-set provided by "lichess.org". We use the data mining, visualization techniques, data processing, and machine learning to identify interesting insights from the data and present objective reports for the various outlined challenges. We were able to achieve good results with a mined dataset of over 120,000 entries and 8GB of disk space.**

I. OVERVIEW OF THE REPORT

The report begins with a detailed introduction, where we discuss about the game of chess in general, the rules associated with it and a very subjective overview of what makes a good chess player based on the insights provided by the famous "Chess Hustlers"(and the players who won against them) from Washington square park. The introduction is followed by the overview of the data-set and the data mining involved in extraction of the data. The fourth section deals with the feature engineering involved with each separate challenge that we had decided to tackle along with the rational involved in the pursuit of the feature with a hint of foreshadowing as to how to feature performed. Then we delve deep into the methods involved for each specific challenge, we discuss first about the baseline model, then the more convoluted models and finally we enumerate the wins and the losses involved with each of the challenges. We end the report with a conclusion and a bibliography.

II. INTRODUCTION

According to Wikipedia, Chess is a two-player strategy board game played on a checkered board with 64 squares arranged in an 88 grid. Play involves no hidden information. Each player begins with 16 pieces: one king, one queen, two rooks, two knights, two bishops, and eight pawns. Each piece type moves differently, with the most powerful being the queen and the least powerful the pawn. The objective is to checkmate the opponent's king by placing it under an inescapable threat of capture. To this end, a player's pieces are used to attack and capture the opponent's pieces, while supporting each other. During the game, play typically involves exchanging pieces for the opponent's similar pieces, and finding and engineering opportunities to trade advantageously or to get a better position. In addition to checkmate, a player wins the game if the opponent resigns, or (in a timed game) runs out of time. There are also several ways that a game can end in a draw. According to "Chess Hustlers" a good chess player is patient, takes calculated moves and consistent with his approaches. We incorporate their advice in our methods.

With the general rules out of the way, we look at the first challenge which requires us to predict the ELO of the players based on the game transcript. To tackle this challenge well we first need to understand how the ELOs are calculated and what factors affect the flow in the game of chess. The Elo rating system is a method for calculating the relative skill levels of players in zero-sum games such as chess. It is named after its creator Arpad Elo, a Hungarian-American physics professor. A player's Elo rating is represented by a number which may change depending on the outcome of rated games played. After every game, the winning player takes points from the losing one. The difference between the ratings of the winner and loser determines the total number of points gained or lost after a game. The total number of points gained or lost will depend a lot of the difference between the ELO before the game begun. An Elo rating is a comparative rating only, and is valid only within the rating pool where it's established, that would mean that the ELO dataset that we had taken from Lichess will only be able to predict the ELOs for the player in Lichess and not any other chess servers. As a direct consequence of the ELO based ranking the model may not be directly portable to other chess server. That said, the approach should ideally be similar.

The second challenge requires game type prediction from transcript. The game types that we are considering are Classic Chess, Rapid Chess, and Blitz Chess. These types differ from each other in terms of the time limit, each variant requiring faster move timings from the previous one. The game of chess requires a considerable amount of time to make each move since all the parallel moves have to be considered before dwelling deep into a move and checking if that move will maximize the chance of securing a win. Now when limiting the time for each moves the probability of the quality of move decreasing is very high. We use the per move information and metric while analyzing the data to predict the variant of the game.

The third challenge subtranscript-level analysis, requires us to predict/analyze the quality of game just by looking at the final board positions. There are a number of problems that we are tackling over here. First we define the quantitative measure which defines the quality of the game then we try to extract data from the board FEN and finally use machine learning models to verify our hypothesis.

The fourth challenge is to predict the AI v/s human games. We started to look for clues marking the games played by the AI. Since we were not able to find a feature which directly tells us if a player is AI or not we had to conjure up a heuristic to analyze the data.

The fifth challenge requires to predict upsets in a game where a weak player wins against a strong player. When a stronger player lose to weaker ones, ELO rating of weaker players increases drastically, which that of stronger player drops sharply . We tried to predict such games through machine learning models and we were able to predict such game very accurately.

III. Overview of the Dataset

The dataset provided by the website lichess.org is in terms of the PGN files. Portable Game Notation (PGN) is a plain text computer-processible format for recording chess games (both the moves and related data), supported by many chess programs. The various fields in the dataset available are as follows :-

1) Event: The type of the game being played.
2) Site: The UUID for each game in form of a URL which can be accessed.
3) White: The UserID of the white player.
4) Black: The UserID of the black player.
5) Result: The result of the game from the white players' perspective.
6) UTCDate: The date when the game was played.
7) UTCTime: The time when the game was played.
8) WhiteElo: The ELO rating of the white player.
9) BlackElo: The ELO rating of the black player.
10) WhiteRatingDiff: The change in the rating of the white player.
11) BlackRatingDiff: The change in the rating of the black player.
12) ECO: The Encyclopedia of Chess Openings (ECO) is a classification system for the opening moves in chess.
13) Opening: The name of the opening move.
14) TimeControl: The time control value.
15) Termination: The way in which the game was terminated possible values include "Time forfeit", "Normal" etc.
16) The list of the moves played during the game is presented in a new line as part of the PGN file.

Various games in one PGN file are new line separated after the move list. This information is particularly important since we want to parse the file to extract insights from it.

IV. Data Processing and Feature Engineering

To understand our data processing pipeline we first need to understand the way the PGN files are created. Each file has multiple games and each game contains data about the game, players, and the move list. To extract insights from the data we need to first evaluate the move list for each game since most of the information of a game is stored in the move list itself. Once we have the information about the move list we store the complete information in a No-SQL database mongoDB for easy access. The complete data processing is done on a cloud neutral architecture currently running on google cloud platform.

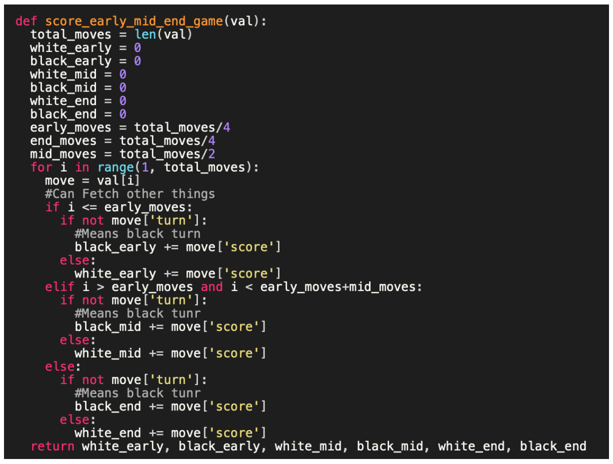
Algorithm 1: Processing game data

```
Result: Series containing information about the
        move distributions per player
df = read_data_from_mongo();
while more data is available do
    moveList = data["move"];
    while next move is available do
        if white_move then
            white += score_early_mid_end_game();
        else
            black += score_early_mid_end_game();
        end
    end
end
```

A. The Data Pipeline

Our data pipeline uses the PGN files from Lichess.org as input and stores the processed data in MongoDB server for easier access. As shown in the figure each PGN file is read by a python program which then processes each file and stores the data. We process our data in multiple ways to ensure that we extract the most information from each game given the time and the processing power constraints. The approaches are defined below.

1) Using the Stockfish engine: In this approach we iterate over each game, storing first the metadata i.e the data related to the time/data, player information, termination information etc into the database. We then iterate over each move in the move list creating a FEN(Forsyth–Edwards Notation) for each move in the move list and passing the FEN value to the stockfish engine to get a score for each move. We then store this list of moves with their score into the database for easier access.



```python
def score_early_mid_end_game(val):
  total_moves = len(val)
  white_early = 0
  black_early = 0
  white_mid = 0
  black_mid = 0
  white_end = 0
  black_end = 0
  early_moves = total_moves/4
  end_moves = total_moves/4
  mid_moves = total_moves/2
  for i in range(1, total_moves):
    move = val[i]
    #Can Fetch other things
    if i <= early_moves:
      if not move['turn']:
        #Means black turn
        black_early += move['score']
      else:
        white_early += move['score']
    elif i > early_moves and i < early_moves+mid_moves:
      if not move['turn']:
        #Means black tunr
        black_mid += move['score']
      else:
        white_mid += move['score']
    else:
      if not move['turn']:
        #Means black tunr
        black_end += move['score']
      else:
        white_end += move['score']
  return white_early, black_early, white_mid, black_mid, white_end, black_end
```

Fig. 1. Scoring the early, mid and end game

2) Ranking the piece placement based on the win / loss of the player performing the move in the game: This is a simple but remarkably effective method of processing the current state of the game with respect to the win/loss ratio of the game. In each game increment the value of each state with 1 if that state is achieved after the move of the player who eventually wins the game. Similarly we decrease the value of the state if the player who's move resulted in that state in the game eventually loses the game. In this way for each state we have a count

$$value = wins - losses \tag{1}$$

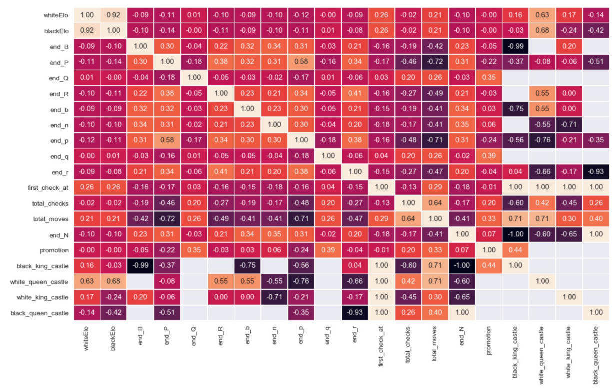


Fig. 2. Correlation Matrix

B. Feature Engineering

We engineer a number of features from the move list and the metadata of the game to assist us in our voyage. The list of the features that we created are as follows :-

1) Score: We score each and every move based on the probability of reaching a check-mate after that move is played. We use the Stock-fish engine to determine this value for each move per game.
2) State Score: We score each possible unique board state on bases of wins and losses as described in the previous section. We use the FEN as the unique identifier for each board state. For the sake of simplicity we consider similar pieces together, for example both the black knights will be considered the same.
3) Bitwise encoded board state: We encoded the fen into 0's and 1's to provide a bitwise encoding for each state for faster possible processing and holding more information in the string.
4) Early Game Score: According to [19] we can characterize the middle game as the portion of the game between early and the end game and having 40-50 moves which is approximately 50% of the total moves in a game. We work under the assumption that the first 25% of the moves are the early game moves. We calculate the average of the early game score for this field.
5) Mid Game Score: Similar to the above field we calculate the average score of the moves in the mid game section.
6) End Game Score: Again similar to the above two we calculate the average score of the moves in the end game section of the game of chess.
7) Aggregate Game Score Per Player: The average score of the moves made by a player in each game.
8) Average Difference in Score Per move per player: The mean difference between each consecutive move per player. We felt this is an interesting feature since it would give us an idea about the consistency of the player in the game.
9) Queen Movement: Tracking the position of the most powerful piece on the board might help us in extracting interesting insights
10) Promotion: Promotion of a pawn to one of the more powerful pieces is tracked, knowledge of a promotion can help us understand the quality of the player who earned the promotion and the quality of(or lack thereof) of the player against which the game is being played.
11) Total Checks: Another interesting insight from the board play is the total number of checks during the game play. In our experience the higher the number of checks the more mediocre the player is, since the higher quality player might want to sneak a check in only when he has the opportunity to end the game.

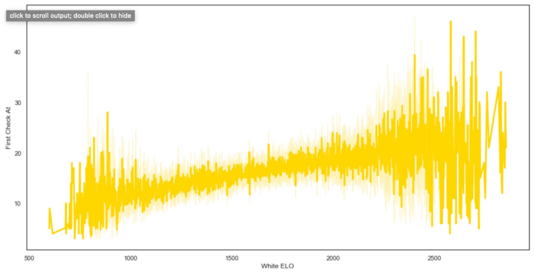


Fig. 3. The move when the first check happens vs ELO

12) First Check At: The move number where we see the first check. The above figure shows that the higher the value for the first check is the lower the number of checks are.
13) White/Black King Castle: A Boolean variable which tells the if the white king has castled during a game. Castling being a mid level move is a sign of player being >= Amateur level.
14) White/Black Queen Castle: A Boolean variable which tells if the white/black queen has castled during the game.
15) Total number of moves: Total number of moves being played in the game.
16) End Pieces: The information about the total number of pieces left on the board after checkmate. Will help us gauge the level of the game.

Each of these features were generated with the challenges in mind. Some were extremely useful and some others not so much.

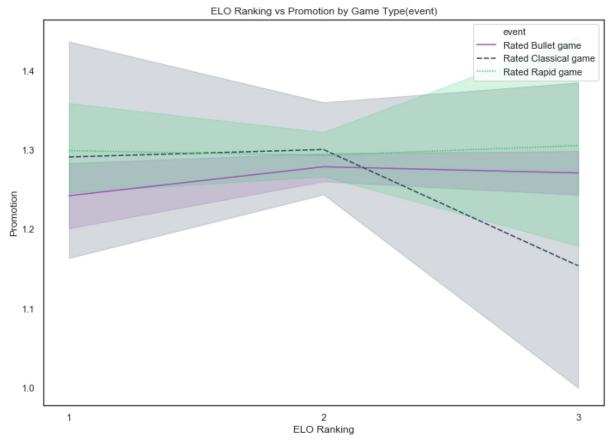


Fig. 4. Average number of promotions by ranking

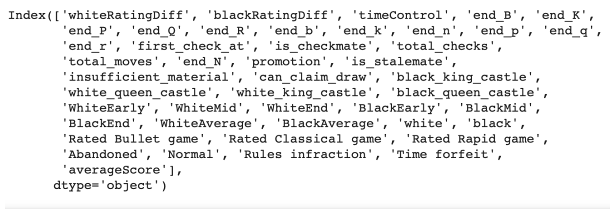
V. PROCEDURES

In this section we take a look at the experimental setup for each challenge. We discuss our thought-process and rational for choosing each model.

A. *Rating prediction from game transcript*

In this challenge we had to predict the ELO rating for a player based on the gameplay. The challenge was to parse the transcript in such a way that we could extract information from the transcript without changing the dimensionality of the dataset(asymptotically speaking).

1) Predictor Variables: The primary variable which helped us in this endeavour was the move list that the data provided. We used the transcript as the input to a python program which used stockfish to assign rating to each move. Once we had the ratings in place we extracted more data from the transcripts.



```
Index(['whiteRatingDiff', 'blackRatingDiff', 'timeControl', 'end_B', 'end_K',
       'end_P', 'end_Q', 'end_R', 'end_b', 'end_k', 'end_n', 'end_p', 'end_q',
       'end_r', 'first_check_at', 'is_checkmate', 'total_checks',
       'total_moves', 'end_N', 'promotion', 'is_stalemate',
       'insufficient_material', 'can_claim_draw', 'black_king_castle',
       'white_queen_castle', 'white_king_castle', 'black_queen_castle',
       'WhiteEarly', 'WhiteMid', 'WhiteEnd', 'BlackEarly', 'BlackMid',
       'BlackEnd', 'WhiteAverage', 'BlackAverage', 'white', 'black',
       'Rated Bullet game', 'Rated Classical game', 'Rated Rapid game',
       'Abandoned', 'Normal', 'Rules infraction', 'Time forfeit',
       'averageScore'],
      dtype='object')
```

Fig. 5. Predictor Variables

2) Target Variable: The target variable is supposed to be the ELO of the players. Our first approach was to segregate the players based on the rating in three groups :- Novice, Amateurs, Master. We first took the average ELO of the two players per game named it "Average ELO" and then we took the mean and standard deviation of the whole column. Any entries below 1 standard deviation from the mean would be novice and any above 1 standard deviation above the mean would be masters. The distribution of the average ELO was very close to normal distribution which helped us uniformly segregate the data. This helped us reach our initial classification model estimates.

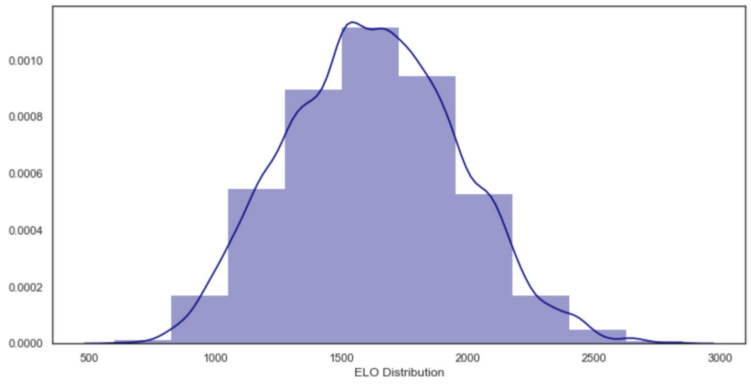
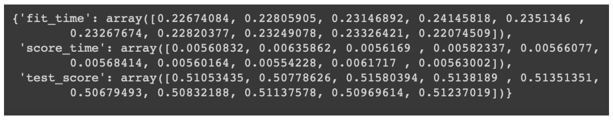


Fig. 6. Distribution of the average ELO rating

Now since each game has two players our first approach for the baseline model was to take the average ELO of the players and then move forward with the modelling. Once we segregated the white features from the black features we used both of them as separate entries effectively doubling our data. This helped us perform the regression modelling.

3) Modelling: We started our Modelling for this challenge as a simple classification problem. As mentioned before we divide the data into 3 groups and then perform modelling.

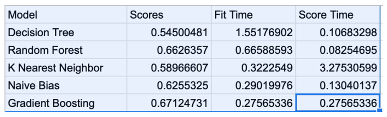
1) Baseline Model(Decision Tree): Our first model used just the average move score per game and the total number of moves then passed it through a simple decision tree. We ran a cross validation with 10 folds and here are the results:-



```
{'fit_time': array([0.22674084, 0.22805905, 0.23146892, 0.24145818, 0.2351346 ,
        0.23267674, 0.22820377, 0.23249078, 0.23326421, 0.22074509]),
 'score_time': array([0.00560832, 0.00635862, 0.0056169 , 0.00582337, 0.00566077,
        0.00568414, 0.00560164, 0.00554228, 0.0061717 , 0.00563002]),
 'test_score': array([0.51053435, 0.50778626, 0.51580394, 0.5138189 , 0.51351351,
        0.50679493, 0.50832188, 0.51137578, 0.50969614, 0.51237019])}
```

Fig. 7. Results of the baseline classification model

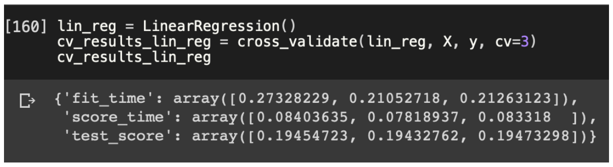
2) Other models: We tried a bunch of other more complex models to validate our outcomes and the results are listed below.



Model	Scores	Fit Time	Score Time
Decision Tree	0.54500481	1.55176902	0.10683298
Random Forest	0.6626357	0.66588593	0.08254695
K Nearest Neighbor	0.58966607	0.3222549	3.27530599
Naive Bias	0.6255325	0.29019976	0.13040137
Gradient Boosting	0.67124731	0.27565336	0.27565336

Fig. 8. Results for classification modelling

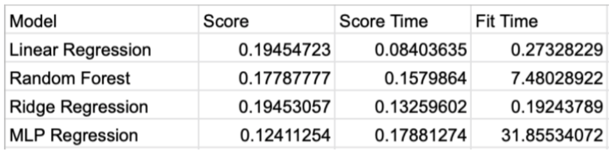
3) Regression Modelling: Now that we had an acceptable accuracy in the classification modelling we moved to the regression modelling. Our baseline model was a simple linear regression model.
Once we had the results of the baseline model we proceeded to apply more complex models to the data. Unfortunately the results either deteriorated or remained the same even with Random forest and ridge regression models. The following are the scores that we were able to achieve.



```
[160] lin_reg = LinearRegression()
      cv_results_lin_reg = cross_validate(lin_reg, X, y, cv=3)
      cv_results_lin_reg

      {'fit_time': array([0.27328229, 0.21052718, 0.21263123]),
       'score_time': array([0.08403635, 0.07818937, 0.083318  ]),
       'test_score': array([0.19454723, 0.19432762, 0.19473298])}
```

Fig. 9. Results for baseline regression modelling



Model	Score	Score Time	Fit Time
Linear Regression	0.19454723	0.08403635	0.27328229
Random Forest	0.17787777	0.1579864	7.48028922
Ridge Regression	0.19453057	0.13259602	0.19243789
MLP Regression	0.12411254	0.17881274	31.85534072

Fig. 10. Results for regression modelling

B. Game type prediction from transcript

Our aim with this problem is to predict the transcript using just the game type. This proves to be a little challenging since some game types are more popular than others. Most online players want to play fast paced variants of the game rather than slow and more elaborate classical chess.

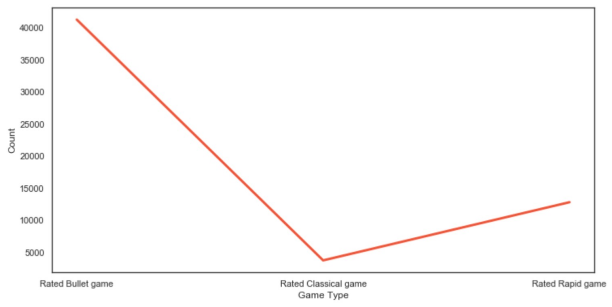


Fig. 11. Distribution of the games in dataset

Although this doesn't stop us from achieving a high accuracy we still had to engineer a set of features to achieve better accuracy. One very interesting insight is that one would expect the number of checks to decrease as the speed of the game increases but a very interesting insight is that the number of checks for the rapid and classical chess variants are very similar but same can't be said for the bullet variant. The average value drastically decreases.

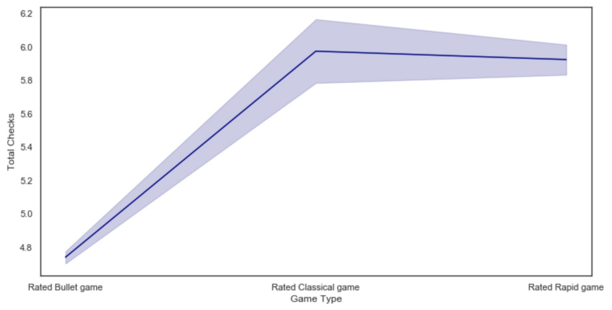


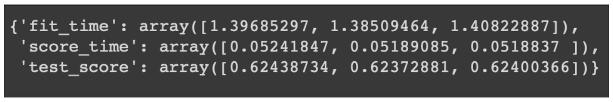
Fig. 12. Average amount of "checks" per game type

1) Predictor Variable: The predictor variables remain essentially the same from the previous challenge but the target variable is what has changed.

2) Target Variable: We label encoded the event type, our target variable, as follows :- Rapid: 1, Bullet: 2, Classic: 3. This helped us in directly fitting the target variable into the standard libraries and getting the results.

3) Modelling: With the help of the engineered variables and the event as the target variable we were able to achieve good results for this challenge as well. The approach and the results for the modelling are highlighted below.

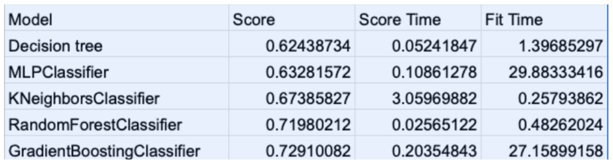
1) Baseline Models: The baseline model was a simple decision tree with our predictor variables. We ran a k-fold cross validation on it to get the results. Shown in the figure below



```
{'fit_time': array([1.39685297, 1.38509464, 1.40822887]),
 'score_time': array([0.05241847, 0.05189085, 0.0518837 ]),
 'test_score': array([0.62438734, 0.62372881, 0.62400366])}
```

Fig. 13. Results for baseline model

2) Other Models: After achieving acceptable test scores for the baseline model we turned to more convoluted models using same set of the variables.



Model	Score	Score Time	Fit Time
Decision tree	0.62438734	0.05241847	1.39685297
MLPClassifier	0.63281572	0.10861278	29.88333416
KNeighborsClassifier	0.67385827	3.05969882	0.25793862
RandomForestClassifier	0.71980212	0.02565122	0.48262024
GradientBoostingClassifier	0.72910082	0.20354843	27.15899158

Fig. 14. Results for baseline model

C. Subtranscript-level analysis

If we are given the final board position, then we can easily get the pieces of the two players, from which we can get the quality of play. Quality of play can be defined as how aggressively the two players have played to arrive at the current board state. Theoretically, a master player playing against a novice is expected to defeat the novice player badly and would directly target the king, so the total pieces left on the board will be more for both players. However, player with similar ELO ratings, playing against each other are expected to have less pieces left on the board at the end. A final board position can be depicted as shown in figure.

We extracted the FEN (Forsyth–Edwards Notation) for the final board. From there we can get the pieces of individuals. Below is an analysis for one of the final board FEN.

```
            Knight  Rook  Pawn  Bishop  King  Queen
Black          2      2     7       2     1      1
White          0      2     7       2     1      1
Black is winner
```

Fig. 15. Final chess board

From the final pieces on the board we tried to predict the ELO of the players. We applied Linear Regression as a baseline for prediction but didn't get enough exciting results.

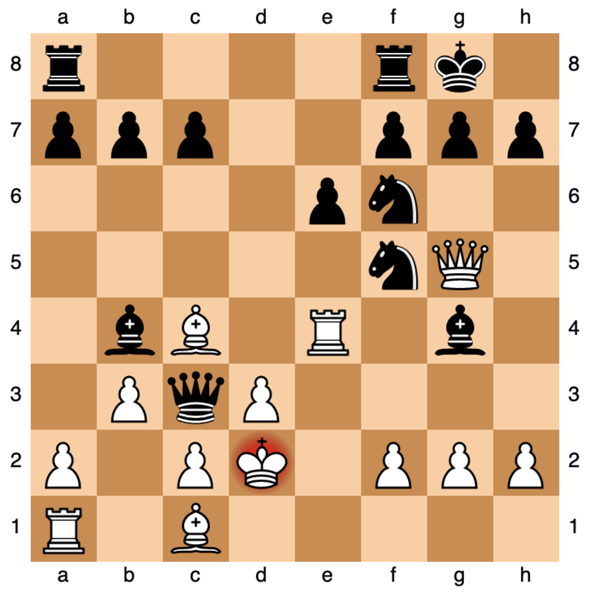


Fig. 16. Final chess board

D. Computer V/S AI

This challenge was particularly difficult because the data-set had no labels indicating that an AI was playing a game or not. But we were able to develop a heuristic to overcome that challenge by rationalizing the behavior of an AI v/s a human.

1) Heuristic: This section first requires us to find a set of predictor variables which will help us to determine if a player is an AI. We rationalized that an AI would ideally be more consistent in its move choices as compared to a user. For example: A 6 level AI will ideally only play 6 level AI moves from the available move list while a low rated player might play some very good move or some very bad moves during a game. We rationalized that the difference in moves for a player will be very high and for the bot should be very close to 0.

This turns out a very interesting heuristic. Because we learn that a lot of human players are consistent as well with their moves not just the AI and most of the highly rated players will be more consistent with their moves as well.

Unfortunately we couldn't verify our heuristic against real world data and hence we don't know how correct this is.

E. Characterizing upsets in chess

This was an interesting challenge as upsets in a game has a major impact on the players ELO rating. When a stronger player lose to weaker ones, ELO rating of weaker players increases drastically, which that of stronger player drops sharply . When we plotted our data for average ELO ranking as explained above, we found that it follows a bell curve

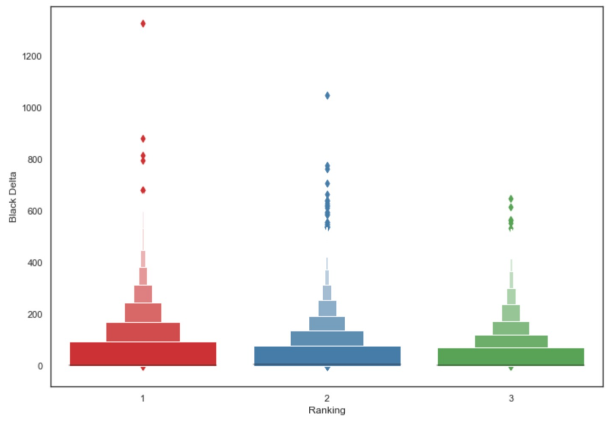


Fig. 17. Difference in move quality v/s the player ranking

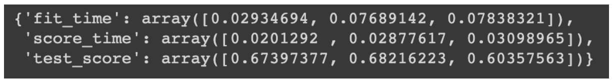
distribution. So for the task of characterizing upsets, we took players with ELO one standard deviation below mean, who won against players with ELO one standard deviation above mean. This gave us a good subset of data for this classification task.

1) Predictor Variable: In this task also, we have kept the input variables as in previous tasks.

2) Target Variable: The target variable is this challenge is to predict the games in which a weaker players wins.

3) Modeling:

1) Baseline Model: The baseline model was Linear Regression with our predictor variables. We ran a k-fold cross validation on it to get the results. Shown in the figure below



```
{'fit_time': array([0.02934694, 0.07689142, 0.07838321]),
 'score_time': array([0.0201292 , 0.02877617, 0.03098965]),
 'test_score': array([0.67397377, 0.68216223, 0.60357563])}
```

Fig. 18. Variation of ELO rating in upsets

2) Other Models: To improve the accuracy of prediction we used other classification models, namely KNeighbor classifier, and Random forest. The results are shown below:

Model	Score	Score Time	Fit Time
KNeighborsClassifier	0.99948214	0.01073408	0.10082126
RandomForestRegressor	0.71665411	0.1007762	0.01339674

Fig. 19. Results or other models

As we can see from the results that we are able to characterize the upsets in a game very accurately . Below is a figure that shows how ELO ranking varies in these upsets. Theory would say that there would be sharp increase and decrease in the ELO ratings, and our model validates the theory.

Also we observe a pattern in such upsets. We observed that the games in which a weaker player wins over a strong one, were either Rated Biltz game or Rate Bullet game as referenced above.

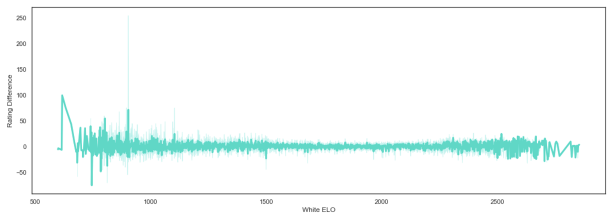


Fig. 20. Variation of ELO rating in Upsets

VI. Wins and Draws

In this section we discuss the wins and challenges in case of each of the challenge that we had worked on.

A. *Rating predictions from game transcripts*

1) Wins: We were able to predict the ELO in terms of novice, amateur and master with a reasonably high accuracy of around 70% which is a win. With this model in place we can tell if a player is new to the game or a seasoned veteran.

2) Draws: The regression analysis on the other hand yielded a score of only 0.19454723 with the best score coming from the linear regression. Improving the score is a challenge that we have yet to overcome.

B. *Game type prediction from transcript*

1) Wins: We were able to achieve a high score of around 0.7 for our classification problem which is a reasonably high value. Our data mining from the move list helped us achieve this high a score.

2) Draws: We were able to achieve good accuracy with the analysis the only thing that we are lacking is an even better accuracy which is a never ending pursuit.

C. *Subtranscript-level analysis*

1) Wins: We were able to analyze the pieces of individuals and tell how aggresively a player has played.

2) Draws: We were not able to get a good prediction for difference in ELO rating from the final board position.

D. *Computer vs. human?*

1) Wins: We were able to conjure up a heuristic and interesting insights without the data being present in the dataset.

2) Draws: We were not able to back this heuristic with data nor we were able to verify the applicability. In future we plan to incorporate the AI data when it becomes available.

E. *Characterizing upsets in chess*

1) Wins: We were predict whether this game would be a upset or not.

2) Draws: As the dataset for games in which lower ELO rating player played against a stronger one are very less, so the nearly accurate results might be due to overfitting of data. More data of the same, would have helped us to validate our model more strongly.

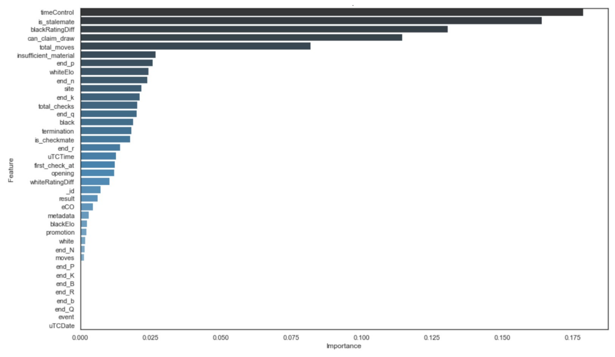


Fig. 21. Feature importance

VII. Conclusion

In this report we were able to tackle the challenges reasonably well. We worked on each challenge individually, working on them through multiple angles, stating are observations and concluding with a list of wins and draws. We created a mongodb pipeline to parse more than 120,000 rows and assign stockfish analysis to the move set in each of them. The data, the cloud infrastructure and the analysis show a lot of promise and are portable enough that they have highly reusable code structure which can be used in future projects in this domain. In the end we were able to tackle not one/two but all 5 challenges with respectable results in each of them, which we consider an achievement.

References

[1] https://chess.stackexchange.com/questions/4132/how-to-automatically-evaluate-a-players-performance-in-a-game
[2] https://github.com/petercunha/stock
[3] https://en.wikipedia.org/wiki/List_of_chess_variants
[4] https://en.wikipedia.org/wiki/Chess_strategy
[5] https://en.wikipedia.org/wiki/Branching_factor
[6] https://en.wikipedia.org/wiki/Elo_rating_system
[7] https://www.lichess.net/blog/best-chess-opening-moves/
[8] https://en.wikipedia.org/wiki/List_of_chess_gambits
[9] https://en.wikipedia.org/wiki/Chess_strategy
[10] https://en.wikipedia.org/wiki/Endgame_tablebase
[11] https://www.reddit.com/r/chess/comments/c4nzje/how_lichess_ratings_compare_analysis_of_35/
[12] https://en.wikipedia.org/wiki/Crazyhouse
[13] https://en.wikipedia.org/wiki/King_of_the_Hill_(chess)
[14] https://en.wikipedia.org/wiki/Losing_Chess
[15] https://en.wikipedia.org/wiki/Atomic_chess
[16] https://www.kaggle.com/c/findingelo/discussion/12377#latest-66277
[17] https://database.lichess.org/