SUPERVISED PROJECT

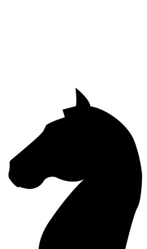
Chess Classification

Guido Giacomo Mussini

988273

Abstract

The aim of this paper is to use supervised learning algorithms to train accurate models able to correctly classify the result of a chess match given several features. A long part of the work regarded the design of new variables, for the most from the SAN notation of the games. Another important section regarded the visualisation of some interesting relation between the features. The last part of the paper concerned the classification using Decision Tree, Random Forest, XGBoost and Logistic Regression. weighing several factors, the best model turned out to be the one provided by the Logistic Regression.



Contents

1 Introduction and Findings

Chess was invented in India, between the 6^{th} and the 8^{th} century. More than 1000 years later, on February 10, 1996 a computer won for the first time against a chess world champion, Gerry Kasparov. Nowadays, chess engines are on a completely different level with respect to human players.
This paper has the much more limited aim of build a model able to correctly classify the results of a chess games Dataset.
The Dataset is composed by a list of around 20,000 chess matches played in different fashions, by players with a wide range of ability.
The theory behind this game is immense, but having a Dataset mostly composed by amateurs, limits the power of many variables, in particular the ones regarding the openings.

The most unexpected finding is probably the low importance of the variable *material* in approximately all the algorithms.
This variable is the nearest one to the concept of 'evaluation of the position' on which engines base their analysis.
On the other hand, the whole group of variables regarding the course of the game has been identified as the most important for the classification, in general even more than the variables which regard the Ratings.
The variables *takes* and *DiffELO* are the most important ones. The first one concerns what happens on the board, the second is important since regards the initial conditions.
Although the importance of *DiffELO* was predictable, such an importance of *takes* was unexpected, even more considering the much lower importance of other similar variables as *checks* or *material.*
At the end, it is interesting to observe the Figure 8: in particular the symmetric distribution of the takes and the distribution of the checkmates which is higher on the lateral sides of the chessboard.

2 Dataset

The dataset is composed by a list of chess matches played on Lichess, which is one of the most popular on-line chess sites. As anticipated, the dataset contains data about different types of matches and players at different levels of ability.

2.1 Variables

The original dataset was composed of 20058 observations and 16 variables:

- **id**: sequence of alphanumeric characters set by the chess server, used to identify uniquely the matches.
- **rated**: logical variable that is TRUE if the game is rated and FALSE otherwise.
  A Match is rated if implies gain or loss of ratings points.
- **created at / last move at**: these two variables contain information about the start and the end of the matches.
- **turns**: number of turns in a match. Note that a turn is completed only when both players have made their move. The only exceptions to this rule occur when:
  - White ends the game with a checkmate
  - Black resigns after a white move
  - Black runs out of time
- **victory status**: character variable that contains information about the way in which the game ended:
  - **mate** if the game ends with a checkmate
  - **outoftime** if the game ends with one of the two players which run out of time
  - **resign** if one of the players resigns.
  - **draw** if the match ends with a draw
- **winner**: character variable that inform about the match winner. It can assume 3 values: 'White', 'Black' or 'Draw'
- **increment code**: each row contains 2 numbers:
  - the first one represents how much time each player has to complete a game, in minutes.

  - the second one represents how much time each player receives at the end of each move, in seconds

  e.g: a 10+2 game is a game in which both players have 10 minutes, and at each move they receive back 2 seconds.

- **white and black id** id that players have chosen on Lichess.
- **white and black rating** it is a number that represents the strength of a player. Lichess adopt a rating system that is similar to the official one, named ELO.
  The ELO system is a complex formula that ultimately assigns points to a player based on the game result and opponent's ELO:
  - Higher the opponent's ELO, higher the points gained in case of victory, lower the points lost in case of defeat.
  - Lower the opponent's ELO, higher the points lost in case of defeat, lower the points gained in case of victory.

  Lichess rating system usually overestimates players strength. Generally speaking, There is a shift of around 300 points between the Lichess rating and the official ELO.

- **moves** each row contains a list of the moves played in the game expressed in SAN notation.
  Below an example of SAN notation of the famous game 6 from the 1972 world chess championship played by Fisher and Spassky:
  1.c4 e6 2.Nf3 d5 3.d4 Nf6 4.Nc3 Be7 5.Bg5 0-0 6.e3 h6 7.Bh4 b6 8.cxd5 Nxd5 9.Bxe7 Qxe7 10.Nxd5 exd5 11.Rc1 Be6 12.Qa4 c5 13.Qa3 Rc8 14.Bb5 a6 15.dxc5 bxc5 16.0-0 Ra7 17.Be2 Nd7 18.Nd4 Qf8 19.Nxe6 fxe6 20.e4 d4 21.f4 Qe7 22.e5 Rb8 23.Bc4 Kh8 24.Qh3 Nf8 25.b3 a5 26.f5 exf5 27.Rxf5 Nh7 28.Rcf1 Qd8 29.Qg3 Re7 30.h4 Rbb7 31.e6 Rbc7 32.Qe5 Qe8 33.a4 Qd8 34.R1f2 Qe8 35.R2f3 Qd8 36.Bd3 Qe8 37.Qe4 Nf6 38.Rxf6 gxf6 39.Rxf6 Kg8 40.Bc4 Kh8 41.Qf4
  - 1., 2., ..., 41. indicate the turns
  - White always moves first, which means that white always plays the odd moves, black the even ones
  - the first letter indicates the piece, the other letters indicate the square in which the piece is moved. e.g Bh4 means that the bishop is moved to the h4 square. Here the pieces notation:
    - B bishop
    - K knight
    - R rook
    - Q queen

    * K king

  if the piece is not specified, it is a pawn.

  - 'x' indicates that the specified piece has taken the piece in the specified square: Nxd5 means that the Knight has taken the piece in d5.
  - 0-0 (0-0-0) indicates the short (long) castle.
  - if the match ends with a checkmate, the last move will have a hash at the end, e.g Qf4#. In this example there is not the hash since Spassky resigned.

- **opening eco**: contains the ECO encoding of the opening played in the match.
  The ECO code has been created in 1974 to classify the different opening lines. It contains more or less 500 different lines, labeled in 5 main groups.

- **opening name**: contains the name of the opening played.

- **opening ply**: contains the number of moves played in the opening. This count stops when the games has reached a completely new position or a non-classified line.

2.2 Variable Transformation

2.2.1 Response variable

The aim of this classification project is to create a model able to correctly classify the result of a game given some features. Consequently, the response variable has been identified in the variable **Winner** which can assume 3 possible values:

Result	White	Draw	Black
%	0.498	0.047	0.454

Table 1: matches results

One of the possible values of the variable *victory status* is 'Draw', which obviously perfectly classifies the draws. Since draws were less then 5% of the observations, classifying them would have be challenging and would probably have made necessary the use of specific techniques as undersampling or oversampling to balance the dataset. Moreover, keeping the draws would have made necessary even to remove the variable *victory status* that could provide some interesting information.

After the changes, the response variable is defined as:

Result	White	Black
%	0.524	0.476

Table 2: matches results

the chances of winning for whites are higher since they have a slight advantage given the fact that they always move first.

2.2.2 Modified Variables

- **rated** has been converted into a [0,1] factor variable in which:
  - 1, the game is rated
  - 0, the game is not rated
- **ECO** has been transformed into a factor variable with 5 levels: [A, B, C, D, E] which represent the 5 main labels of the ECO opening code:
  - A: Flank games
  - B: Semi-open games other than the french defence
  - C: Open games and the french defence
  - D: Closed and semi-closed games
  - E: Indian defences

2.2.3 New Variables

- **matchtime** and **increment**: these two variables have been created from increment code.
  - **matchtime**: how much time each player has to complete a game, in minutes
    The game-time is extremely important in a chess game. It define how much a player can think and can be crucial to define the final result, e.g:
    * a stronger player who is winning could lose for time.
    * In a bullet game (games with time $\leq$ 2 minutes), a weaker player can *flag* his opponent, which is start moving his pieces randomly as quick as possible in order to make his opponent to run out of time.
  - **increment**: how much time each player receives at the end of each move, in seconds
    playing a 2+0 match or playing a 2+1 match is completely different. Even a small increment can give a relevant help in thinking,

allowing deeper analysis of the game situation. It is mostly useful in the end game.
It could be simpler play a 2+2 game rather than a 3+0.
This variable can mitigate the effects of matchtime.

- **FirstMove** and **Firstresponse**: features generated from the variable *moves*:

  - **FirstMove** is a factor variable with 2 levels: [Usual; Unusual] depending on the first move played in the game. If the first move is one of the top 3 used moves in the Lichess master database, it assumes value 'Usual', 'Unusual' otherwise.
  - **FirstResponse** follows the same rules of FirstMove: if the response to one of the most used moves is one of the top 3 used moves in the lichess master database, it assumes the value 'Usual', 'Unusual' otherwise.

First Move	e4	d4	Nf3
Common Responses	c5 e5 e6	Nf6 d5 e6	d5 c5 e6

Table 3: Usual Moves and Responses

  This variable can be useful to define whether the game will be standard or not, and whether the players have a basic knowledge about the principal lines of the most common openings.

- **diffELO** created as the weighted difference between *white rating* and *black rating.* Below the procedure:

  1. define the vector deltaELO = *white rating* − *black rating*
  2. define the vector meanRating as the mean by row between *white rating* and *black rating*: $meanRating_i = \frac{whiterating_i + blackrating_i}{2}$
  3. divide meanRating by its own total mean. In this way the elements of the resulting vector will be > 1 if the i-element is greater than the total mean, and < 1 otherwise.
  4. scaling meanRating between [1, 2]:
     $meanRating = \frac{meanRating - \min(meanRating)}{\max(meanRating) - \min(meanRating)} + 1$
     This vector works as a multiplier for the difference in ratings in the higher matches. Higher the mean in ratings of the two players, higher the weight for deltaELO. Consequently, the high rated matches will have the difference in ELO more stretched
  5. finally, find diffELO = $meanRating \cdot deltaELO$

The reason which underlies this procedure is that a difference in ability between a 2600 and a 2700 player is much more relevant than the difference between a 2600 and a 2500 player.
e.g: the ratio between the number of 2700 and 2600 ELO players is 15%, the ratio between the number of 2600 and 2500 ELO players is 32%.

- **Game Moves**: group of 10 variables which contain information about how the pieces were moved during the match:
  - **king.moves**: difference between number of white king moves and black king moves, except for the castling
  - **queen.moves**: difference between number of white queen moves and black queen moves
  - **rook.moves**: difference between number of white rooks moves and black rooks moves, except for the castling
  - **bishop.moves**: difference between number of white bishops moves and black bishops moves
  - **knight.moves**: difference between number of white knights moves and black knights moves
  - **takes**: difference between number of pieces taken by white and number of pieces taken by black
  - **w.castle**: factor with 2 levels:
    1. castle: if the white has castled
    2. no castle: if the white has not castled
  - **b.castle**: factor with 2 levels:
    1. castle: if the white has castled
    2. no castle: if the white has not castled
  - **checks**: difference between the number of checks given by white and number of checks given by black
  - **material**: difference in the value of the material on the board between white and black at the end of the match.
    The piece evaluation is standard and it is used to determinate the strength of the pieces.

Piece	**Value**	**Piece**	**Value**
Queen	9	Knight	3
Rook	5	Pawn	1
Bishop	3	King	-

Table 4: Pieces value

Assigning to the King the value of 1, the starting total value is 40. From this number have been removed all the values of the pieces taken, and added the values of the pawns promoted.
it has been assumed that pawns are all promoted to Queen, as it usually is.
material = (40 − white lost material + white gained material) − (40 − black lost material + blcak gained material)

2.2.4 NA and Unique rows

There weren't missing values in the dataset, and no one has been produced by the new variables.
Some rows presented the same id, which means that a few games were considered more than one time in the dataset. The duplicated rows have been removed.

2.2.5 Deleted Variables

Irrelevant variables have been deleted from the dataset:

- id
- black id
- white id
- started at
- last move at
- moves
- Increment Code

2.2.6 Absurd Data

From the dataset have been removed all the observations about games with less than 5 turns. These matches were probably started by error or resigned for chess reasons.
Moreover, the variable *matchtime* had 46 observation '0'. Since obviously it is impossible to play a match with 0 minutes of times, these observations have been removed.

2.2.7 Summary

After these changes, the dataset is now composed of 17895 observations and 23 variables.

Numeric	Range	Factor	levels
turns	[5 : 210]	rated	2
white rating	[784 : 2621]	victory status	2
black rating	[791 : 2621]	**Winner**	2
opening ply	[1 : 28]	ECO	5
matchtime(m)	[1 : 180]	firstmove	2
increment(s)	[0 : 180]	firstresponse	2
diffELO	[-2590.725 : 2468.320]	w.castle	2
king.moves	[-21 : 19]	b.castle	2
queen.moves	[-15 : 16]	-	-
rook.moves	[-17 : 25]	-	-
bishop.moves	[-15 : 14]	-	-
knight.moves	[-19 : 16]	-	-
takes	[-11 : 13]	-	-
checks	[-39 : 32]	-	-
material	[-58 : 43]	-	-

Table 5: Dataset composition

2.3 Data exploration

In this section will be analyzed how variables are related to verify the presence of some relevant pattern in the data, and to discover whether the behavior of the features is consistent with theoretical expectations.

2.3.1 Winner and Rated

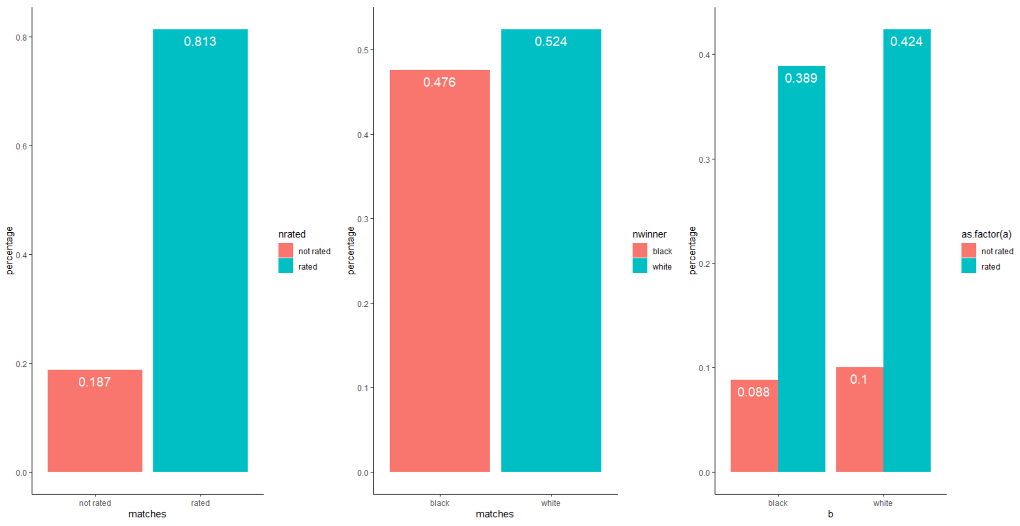


Figure 1: relation between the response variable and rated

The number of rated matches is higher than the number of the friendly ones: 80% vs 20%, which means that the majority of the games have been played with a certain level of focus.

from the second graph instead, it can be seen that the response variable is well balanced: the white victory rate is 52% and the black one is 48%, as previously discussed.

The last graph shows the proportion of winning in rated and not rated matches. It seems that there are no significant differences in the response variable given by a rated or not rated match, only a slight raise in the white victories if the game is rated, but the increment is so small that may not be significant.
This means that probably the variable *rating* will not be particularly useful in the models.

2.3.2 Turns

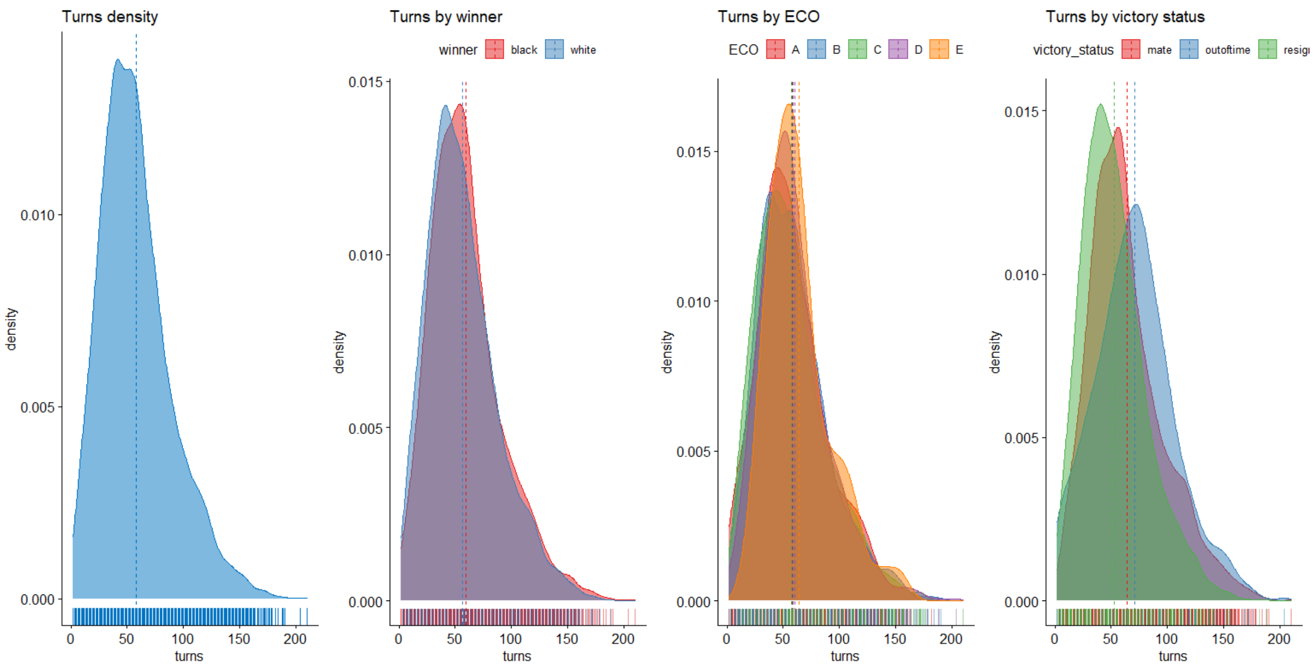


Figure 2: Relation between turns and: response variable, ECO, Victory status

- In the first plot it can be observed that the distribution is clearly not normal. The average is between 58 and 59 turns, which is reasonable with a typical chess match.
- The second plot shows the number of turns grouped by the response variable. There are not significant differences, it can only be observed that the black distribution is a bit shifted to the right, which means that blacks win in average matches with an higher number of turns than whites. However the mean between black and white is approximately the same.
- The third plot is more interesting: Even if all the ECO have more or less the same number of turns in average, it seems that the ECO D (Closed and semi-closed games) and E (Indian defences) are more concentrated around the mean.
  This could indicate that this type of opening leads to a more structured and 'canalized' games.
- In the last plot the differences are even more noticeable: it was predictable that the games which ended with a resign would have a lower number of turns.

As it was predictable that in average the matches that end by time have a higher number of turns with respect to the games which end by mate.
Resigns are more concentrated around the mean with respect to *out-oftime* which are more sparse, with a longer right-tail.

The shape of resigns may signify that the winner of a game *in average* is often clear after around 50 rounds. The fact that this shape is similar to the mate one, implies that most of the players aren't able to foresee the future lines, and so that they are not pro-players. This suggestion is confirmed by looking at the average player's rating, which is around 1600 points, far from being professional:
1600 rating points on Lichess, in fact, it corresponds approximately to 1300 ELO points.

2.3.3 Ratings

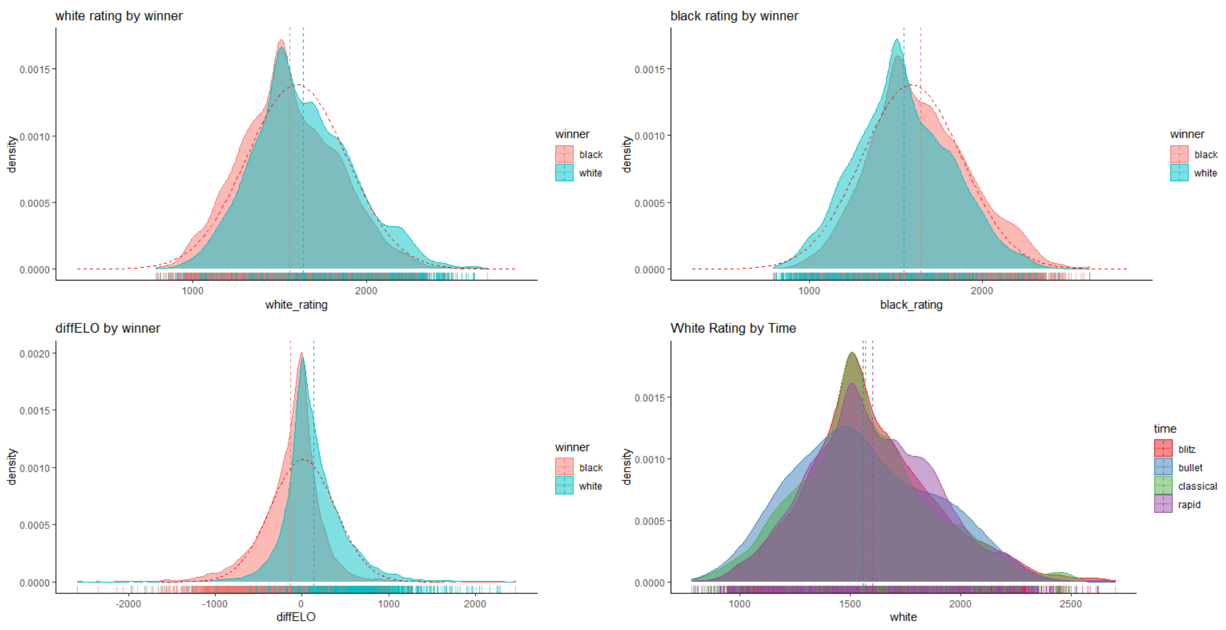


Figure 3: White Rating, Black Rating, DiffELO vs the response variable. White Rating vs Match Time

in the first two graphs can be observed the distribution of *white and black ratings* by the response variable. The dashed line represents a normal distribution curve, black and white rating approximately follow the shape except for the fact that they are a lot more centered around the mean.

As expected, the victory chances are higher for the players who have the higher ratings. the plot of *diffELO* shows how around the center of the distribution -where the ratings of the two opponents are similar- the victory chances overlap. On the other hand, on the tails the victory chances increase more than proportionally in favor of the player with the higher rating, consistently with how this variable is built.

Lastly, it can be observed that the winning average is significantly different, especially for *black rating* and *diffELO*, which means that these two variables will be important in the classification models.

The experience suggests that rating should be strongly correlated with *matchtime*.
Games are divided into 4 main categories:

- **Bullet** time $\in [1 : 2]$ minutes. time is really limited, amateur players makes a lot of mistakes and the match result may depend on the chance
- **Blitz** time $\in [3 : 5]$ minutes. A player has sufficient time to think.
- **Rapid** time $\in [10 : 15]$ minutes. This is generally the most common mode chosen by amateurs.
- **Classic** $\in [30, +\text{inf})$ minutes. this modality is not often used in online chess, is more common in normal chess where however it has other rules.

In the last plot that shows how the ratings are distributed based on the *match time*, it can be noticed how the bullet rating is more sparse than the others. This is because of the lower number of observations in the dataset (table 6), but also because of the component of randomness and fortune associated to this modality: even the Grandmasters can gain or lose dozens of points during the weeks in this mode.
In general a non-professional player has different ratings based on the match time which can differ a lot, even 300 or 400 points.

time	white	black	observations
bullet	1568	1650	39
blitz	1599	1570	720
rapid	1561	1599	5910
classic	1528	1517	1025

Table 6: average rating on match time

However, results differ from forecasts, in fact, taking into account the different number of observations, the average ratings are very similar except for the classic one.
These results probably indicate that the dataset was collected based on similar ratings of players, but the ability of those who have 1570 points in bullet or blitz are certainly greater than the one of those who have 1570 points in classic or rapid.

2.3.4 Opening moves and ECO

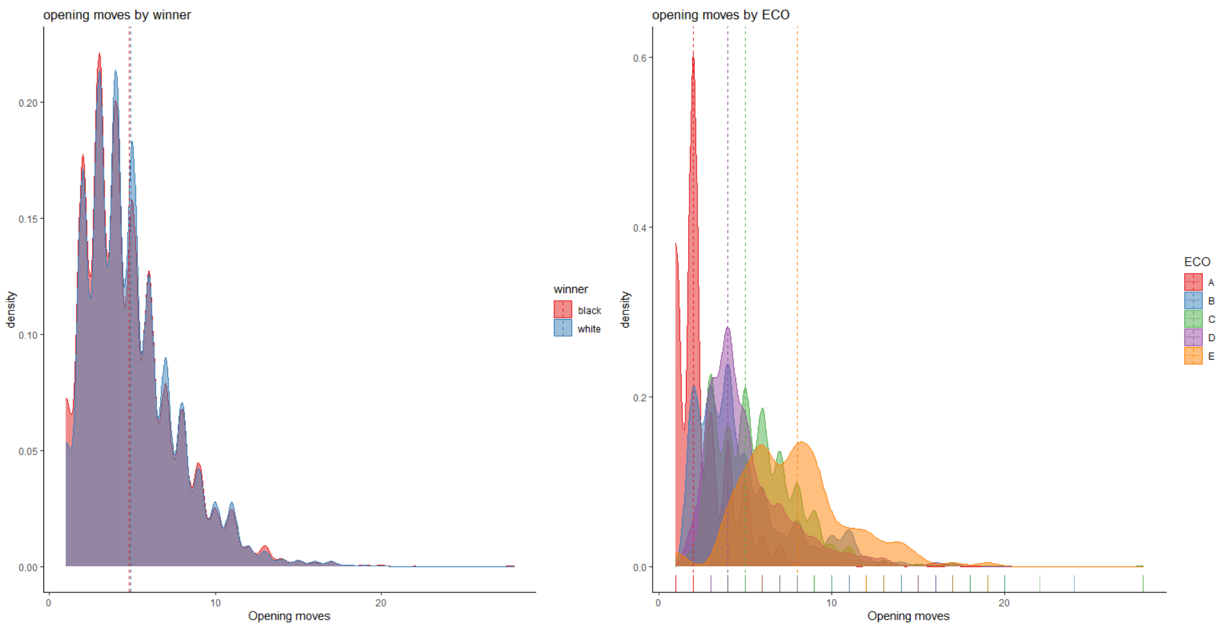


Figure 4: number of opening moves vs the response variable and ECO

the first graph shows the distribution of the different types of opening with respect to the response variable. Unfortunately, the distributions are completely overlapped, as the medians, which means that there is not a real difference in the response variable based on the number of opening moves.

When comparing the number of opening moves and the ECO code,it stands out how much A is different from the others, in particular A openings seems to lead the game to new lines very quickly compared to the other types of openings. This is important since playing out of theory early, when the pieces are not much developed, could influence the final result. On the other hand, the other openings seems to be more or less equally distributed, excluding the E one, but there are too few observations to make inference, as shown by the table below:

ECO code	%	ECO code	%
A	19.4	D	13.5
B	26.1	E	2.5
C	38.5	-	-

Table 7: ECO code frequencies

2.3.5 Time and Increment

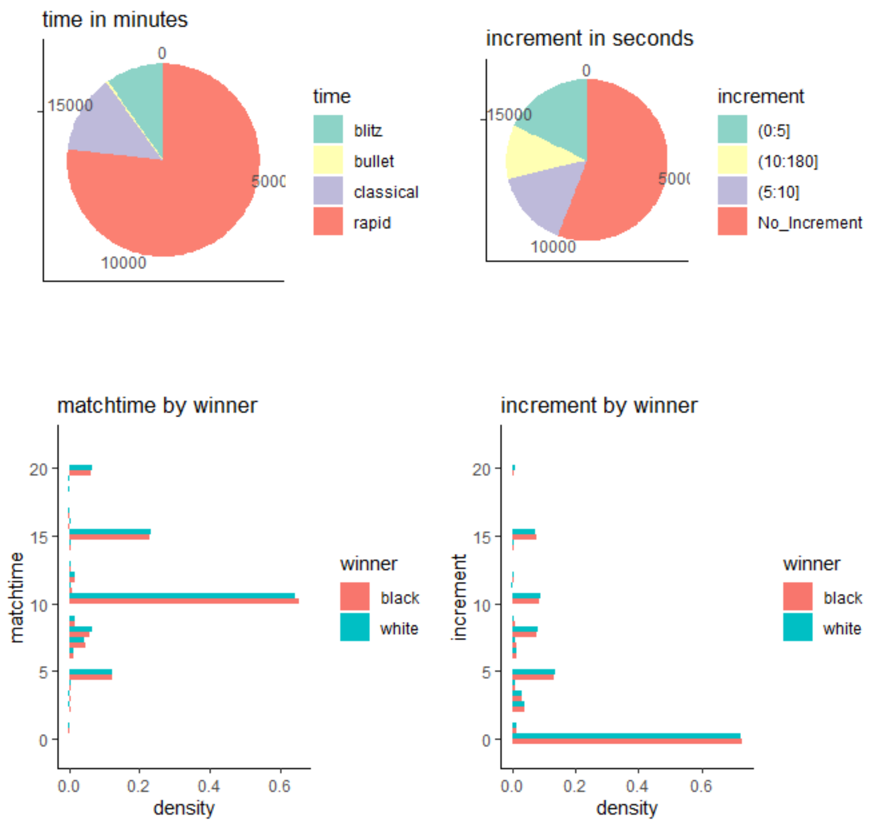


Figure 5: Pie charts of matchtime and increment above, relation between matchtime and increment with the response variable below

The histograms represent the distributions of *increment* and *matchtime* with respect to the response variable, they seems to be equally distributed. For visualisation purposes, the histograms have been restricted in [0,20], but the behavior is roughly the same for all the values.

The two pie charts represent the distributions of the times and the increments in the dataset.
As before, the time has been divided into 'bullet', 'blitz', 'rapid' and 'classic'. it is evident that the majority of the games played are rapid. Increments don't have an official distinction, it has been applied a reasonable one: 'No Increment', 'Increment between 0 and 5 seconds', 'Increment between 6 and 10 seconds', 'Increment between 11 and 180 seconds'.
As it can be seen in the second chart, more than 50% of games is played without increment, and about 30% is played with a small one.
Re-merging these two variables, it can be observed the most frequent games:

time(m) increment(s)	Occurrences
10 0	6979
15 0	1184
15 15	774
5 5	688
5 8	617
8 0	524
10 5	512
20 0	406
15 10	391
30 0	338

Table 8: most common matches type

2.3.6 Balanced and Usual Games

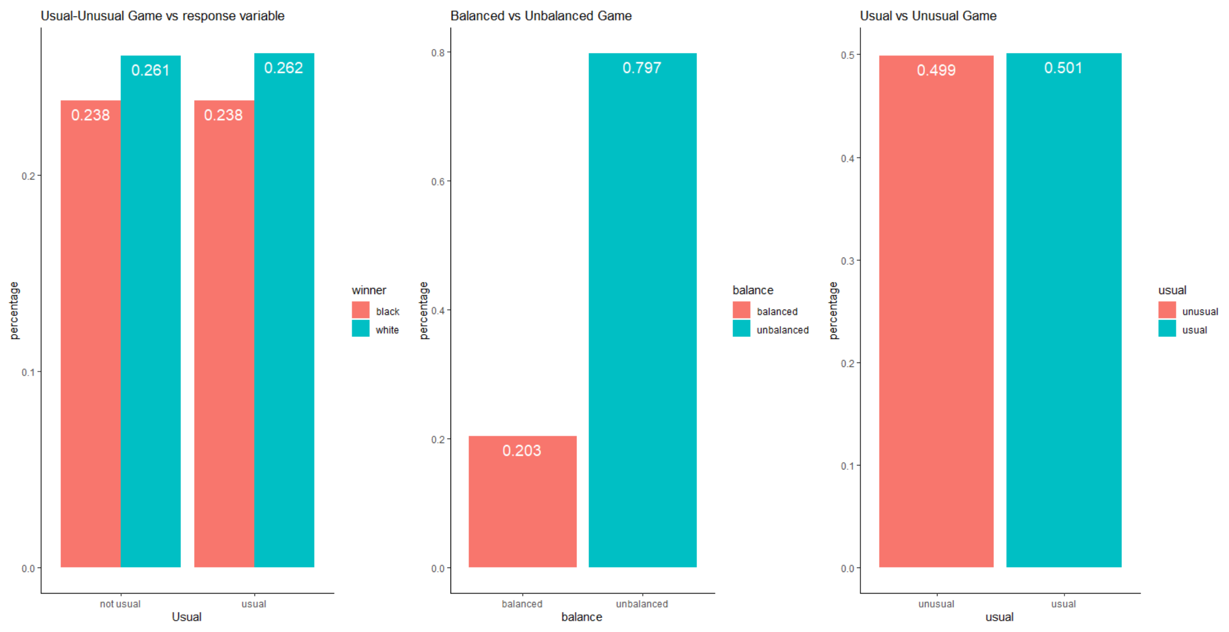


Figure 6: frequency of balanced an usual games

The first graph shows the relation between the response variable and usual-unusual games, representative of the variables *firstmove* and *firstresponse*. However there do not seem to be significant differences.

The second graph represents the percentage of balanced games in the dataset. A game has been considered balanced if $diffELO < 50$. This approximation contrasts with what has been said above regarding the definition of *diffELO*, but it is useful for visual purposes.
In short, a game is **balanced** if the two players has a similar raking, **unbalanced** otherwise.
Analyze this is crucial since for the most unbalanced matches the result should be fairly predictable.

The third graph shows the percentage of the usual games.
A **usual game** is a game in which *first move* and the *first response* are two of the most used ones. it is interesting to note that these 9 combinations are as frequent as the other 391 possible in the first 2 moves.

2.3.7 Game Moves

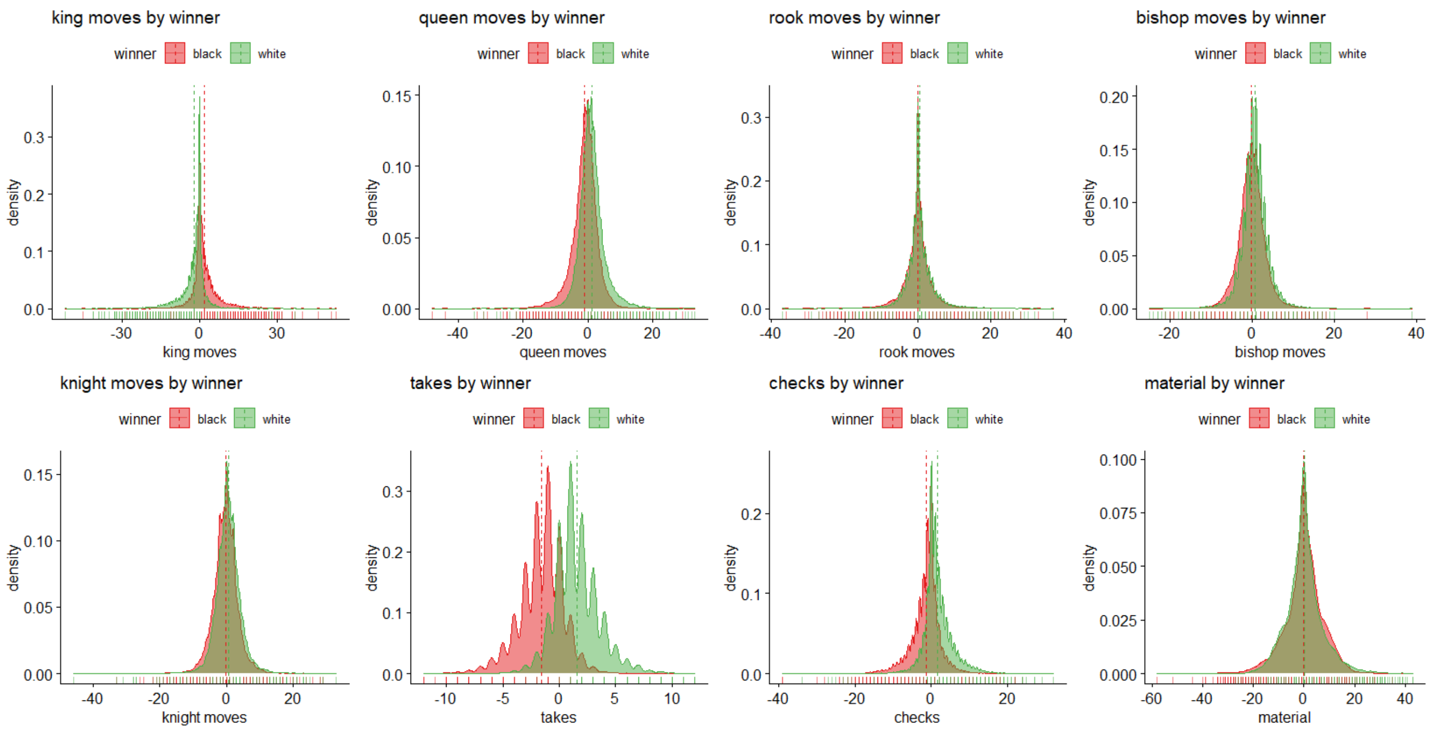


Figure 7: moves variable vs response variable

the figure above shows the different distributions of the *piece.moves* variables with respect to the response variable.
the variables **takes**, **king moves** and **checks** have a clear separation in the mean and in the distribution, which means that these variables will probably be very useful in classification.
Some other differentiation can be appreciated even in **bishop moves** and **queen moves**.

Table 9 shows some relation between *castle* and the response variable.

	White	Black
WC BC	0.50	0.50
WN BN	0.56	0.44
WC BN	0.54	0.46
WN BC	0.50	0.50

Table 9: Legend: W white, B black, C castled, N Not castled

Reading the table by rows it can be observed that if both players castle, the victory rate is the same, which means that there is a slight advantage for the black, since white has the 52.4% of victory rate in this dataset.

The same result is obtained when the white doesn't castle and the black does, which means that if black castle, white has the same chance of victory regardless of its castling.
Otherwise, if black doesn't castle the advantage for the white is noticeable whether it is castling or not.
In summary, the variable **b.castle** will probably be useful for the classification algorithms.

Below the heatmaps that represent the frequencies by each square of takes and checkmates. Since they are the variables with the most difference with respect to the response variable, it could be worth to analyze them briefly.

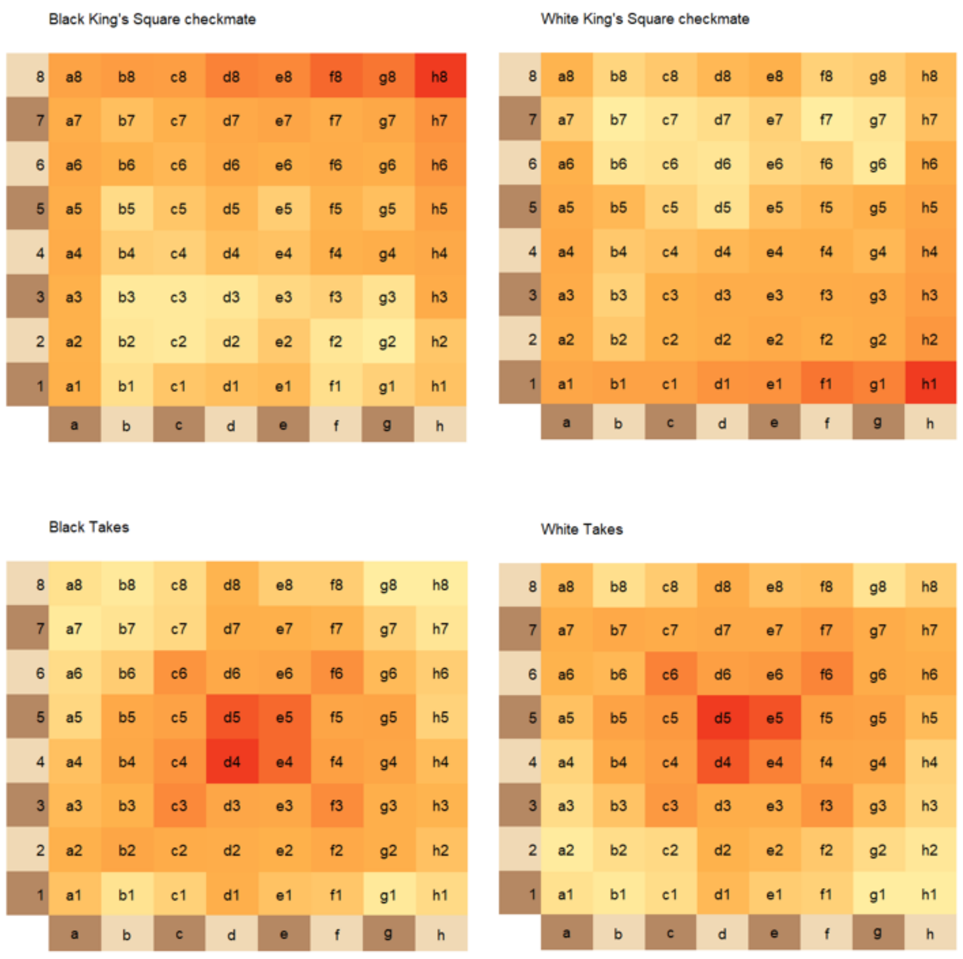


Figure 8: takes and checkmates heatmaps

The white and black kings are mostly checkmated in the lower left square of the respective player, this is probably caused by the so called *back-rank mate*, one of the most popular checkmate among amateurs.
The most common takes are concentrated in the center of the chessboard, which was predictable since in the first phases of the games the players always try to control the center of the chessboard, and so the squares d4, d5, e4, e5. Moreover, there are a lot of gambits that involve these squares. Then, it is interesting to note the high percentage of takes in the boxes c6, f6, c3, f3, squares usually occupied by knights in the firsts stages of a game.

3 Pre Modelling

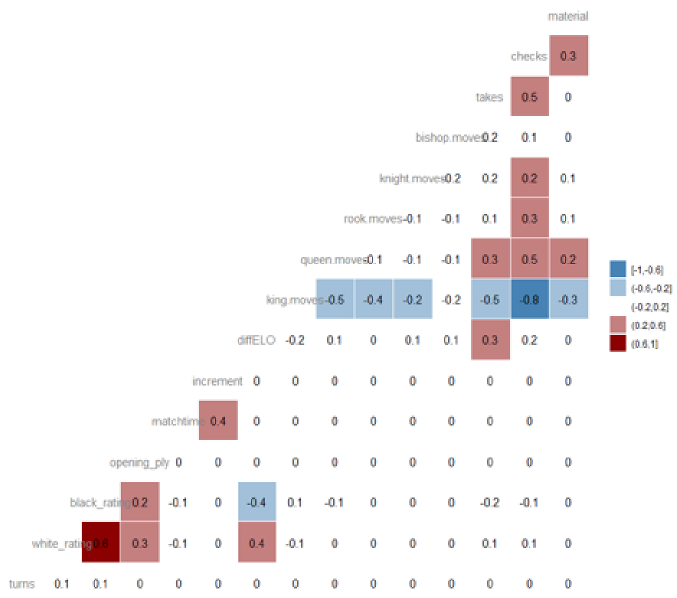
In this section will be performed some statistical analysis on the dataset, in order to eventually make changes to satisfy the assumptions of the algorithms where appropriate.
Not all the algorithms require the same assumptions, so in this stage the analysis will only be exploratory.

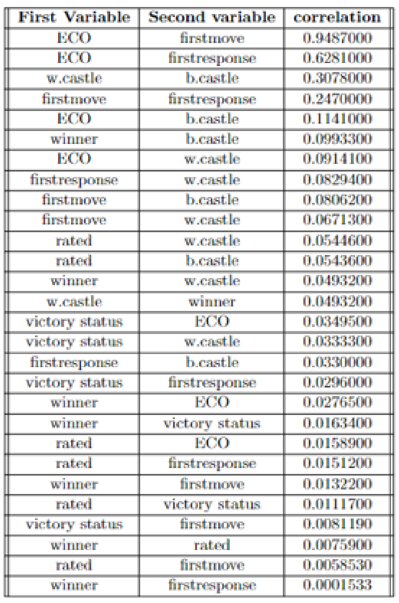
3.1 Scaling

Before proceeding with the analysis, the numeric variables have been scaled since they had different units of measure.

3.2 Correlation

The Dataset is composed by 9 numeric variables, 5 non-ordinal factors with 2 levels and one non-ordinal factor with 5 levels.
Correlation means 'degree of which a pair of variables are *linearly* related'. there are techniques that allow to find the correlations between mixed variables, as the *hetcor()* function on R, but it apply different criteria to different pair of variables.
Hence the analysis have been computed separately for numerical and factors. For the numerical it has been used the *Pearson correlation coefficient*, the *Cramer's V* for factors.
Below the HeatMap associated with the numerical correlation and the table with the factors correlation.





First Variable	Second variable	correlation
ECO	firstmove	0.9487000
ECO	firstresponse	0.6281000
w.castle	b.castle	0.3078000
firstmove	firstresponse	0.2470000
ECO	b.castle	0.1141000
winner	b.castle	0.0993300
ECO	w.castle	0.0914100
firstresponse	w.castle	0.0829400
firstmove	b.castle	0.0806200
firstmove	w.castle	0.0671300
rated	w.castle	0.0544600
rated	b.castle	0.0543600
winner	w.castle	0.0493200
w.castle	winner	0.0493200
victory status	ECO	0.0349500
victory status	w.castle	0.0333300
firstresponse	b.castle	0.0330000
victory status	firstresponse	0.0296000
winner	ECO	0.0276500
winner	victory status	0.0163400
rated	ECO	0.0158900
rated	firstresponse	0.0151200
winner	firstmove	0.0132200
rated	victory status	0.0111700
victory status	firstmove	0.0081190
winner	rated	0.0075900
rated	firstmove	0.0058530
winner	firstresponse	0.0001533

Figure 9: numeric correlation heatmap and factor correlation table

4 noticeable correlations:

- **ECO** with **firstmove** and **firstresponse**: the reason is clear, ECO encodes different opening types, *firstmove* and *firstresponse* do the same thing even if in a simplified way
- **checks** and **king.moves**: this negative correlation is unexpected since as a result of a check the player must either protect the king with another piece or move the king itself. This correlation means that when the difference between the number of white checks and the number of black checks increases, the difference between the number of white king moves and the number of black king moves decrease.
- **b.castle** and **w.castle**: It often happens that the two players castle in the same turn
- **white rating** and **black rating**: Lichess' software tends to pair players with similar rating.

3.3 Outliers

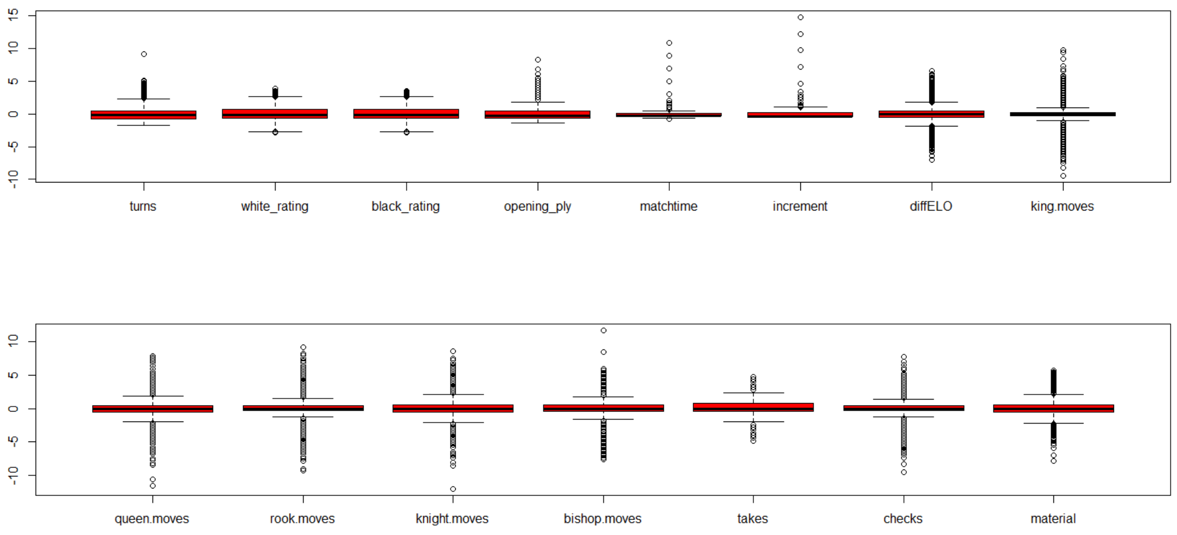


Figure 10: Boxplots of numerical variables

All the numeric variables have outliers, in particular *turns*, *diffELO*, and the *.moves* variables. The presence of outliers could be harmful for algorithms like the Logistic Regression, so before running it they will be treated in some ways. There are several ways to deal with them:

- they can be removed from the dataset, at the cost of a part of the observations
- they could be replaced with the mean or the median. But there are variables like diffELO that have a precise meaning in the dataset, replacing a diffELO = +300 with, for example, 30, will completely change the information.
- they can be capped: cap all the outliers to the q25-q75, this can be a valid option, even if would probably create an unusual density.
- the variables can be divided in classes, like previously done to visualize *matchtime* or *increment*, for example.

4 Modelling

4.1 Train and Test Set

the DataSet has been divided in train and test set:

- **Training Set** 12528 observations, 70% of the dataset
- **Test Set** 5223 observations, 30% of the dataset

the proportion of the response variable has been maintained:

	White	Black
Dataset	0.524	0.476
Training Set	0.524	0.476
Test Set	0.524	0.476

Table 10: Dataset proportions

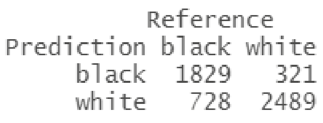
4.2 Decision Tree

4.2.1 Theoretical framework

Decision Trees are non parametric supervised learning methods used for both classification and regression. In classification, the response variable is qualitative.
Decision tree can handle multi-class problems and both numerical and categorical predictors.
It is robust to outliers, since the partitioning is based on the proportion of samples in the split ranges and not on absolute values. Intuitively, tree based methods cut the predictor space based on the maximal reduction of the *classification error rate*, and it is not important how far the observations are from the cut-line.
It is simple to implement and understand, trees can be visualized and interpreted easily.
On the other hand, this method can't compete with other algorithms in terms of prediction accuracy, and if the tree become too complex, it can overfit.

4.2.2 Decision tree without parameter tuning

Below, the confusion matrix of a first attempt.



```
            Reference
Prediction black white
     black  1829   321
     white   728  2489
```

Figure 11: Confusion Matrix

On the test set, the metrics are:

- **Accuracy** 80,4%
- **Specificity** 88,6%
- **Sensitivity** 71,5%

This model can classify correctly the results of 8 games over 10, it is interesting to observe the high difference between specificity and sensitivity. Basically, it makes more errors classifying the victories of black.

4.2.3 Decision tree with parameter tuning

There are several parameters that can be tuned in order to improve the performances of a decision tree, in this case it has been decided to tune the *mincriterion*, which is the levels of the statistic that must be exceeded in order to implement a split.
Figure 11 shows the changing in Accuracy with respect to different values of the *mincriterion.* The best parameter has been found at *mincriterion* = 0.65, which corresponds to the red dot in the figure. At *mincriterion* = 1 the accuracy falls since the algorithm can't make any split.

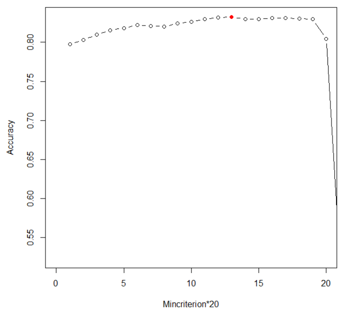
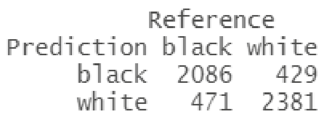


Figure 12: Mincriterion Tuning

After the tuning, the performances of the algorithm generally improved:



```
               Reference
Prediction black white
     black  2086   429
     white   471  2381
```

Figure 13: Confusion Matrix

On the test set, the metrics are:

- **Accuracy** 83,2%
- **Specificity** 84,7%
- **Sensitivity** 81,6%

The accuracy improved a bit, the sensitivity gained more than 10% at the cost of around 3% in Specificity.

Below the visualisation of the tree cut at *max depth* = 3 in order to make it more understandable.

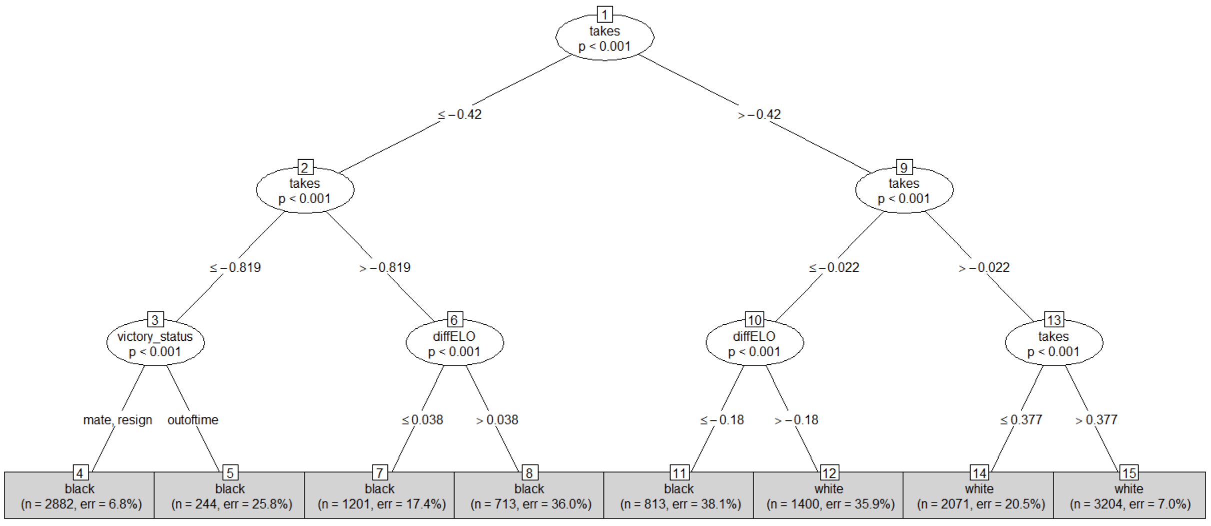


Figure 14: Decision Tree with max Depth = 3

It can be observed that the most important feature here are *takes* mainly, *diffELO* and *victory status*. The Explanatory power of *takes* was predictable since it has been observed how good is in sharply dividing the response variable.

4.3 Ensemble Methods

4.3.1 Theoretical Framework

Unlike the majority of the algorithms that tries to build a model with the highest possible accuracy, ensemble algorithms try to learn a large number of low-accuracy models, and then combine them to create an high-accuracy *meta-model*.
Low-accuracy models are trained by the so-called *weak-learners*: simple and fast algorithms that can't handle with too much complexity. The prediction power of these algorithms needs to be only slight better than random guessing.
An immediate example of weak learner is a simple decision tree, which stops the splitting after a few iterations.
The idea behind these methods is that if weak learners are sufficiently different one from each other, combining a sufficient number of them will generate an high-accuracy model.
The combining method can differ depending on the Algorithm, but in general it is a weighted voting process.
Ensemble methods can handle very well non linearity of the data and in general they produce very few errors. On the other hand, they can be computationally slow with high dimensional datasets, and they lack in explainability: they are like *black boxes* which produce great models but hard to understand.
The ensemble methods used here are the *random forest* and the *gradient boosting*

4.3.2 Random Forest

Random Forest is an ensemble learning paradigms derived from *bagging*, specifically designed to work with decision trees.

How it works
From the training set, the algorithm derives B random samples S_b. The sampling process is made *with replacement*, this adds bias to the model.
Then, it creates decision tree model f_b by each sample Sb, using a modified decision tree algorithms:
at each split in the learning process, the tree algorithm uses only a subset of the features, drawn randomly. This process avoids the correlation of the trees, which don't help in improving the accuracy.
After this process the algorithm has created B decision trees, the prediction of a new example is then obtained combining these models.

Model
It has been performed a cross validation and a grid search to tune the best number of features that have to be selected at each split.
The process provides *best mtry* $= 6$.
In figure 15 the different values of the accuracy by different number of features.

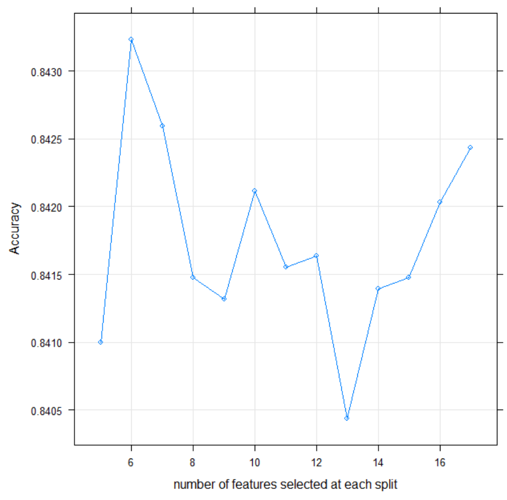


Figure 15: number of features tuning

Figure 16 shows the features importance in the random forest algorithm:

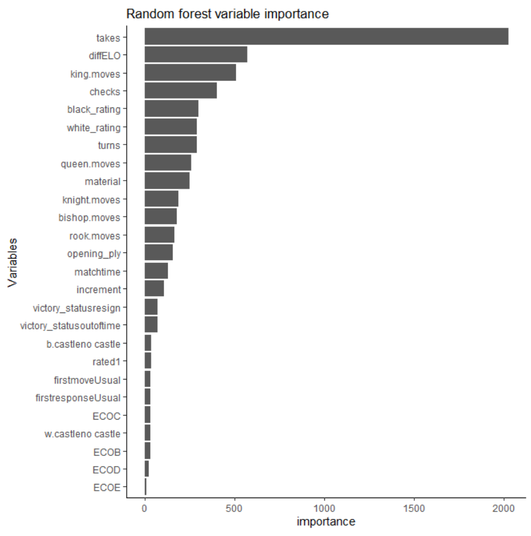
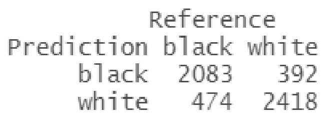


Figure 16: features importance

the most important variable is *takes* by far, followed by diffELO. These results are consistent with those of the decision tree.
It has to be noticed that in the first 10 positions, 6 variables derives from the course of the game, and 3 from the players' ratings.
Consistently with which has been seen in the visualisation section, the first positions are occupied by variables which are able to divide clearly the the response variable.
Moreover it has to be observed that victory status here seems to be marginal, in contrast with what has been seen in the decision tree model.

Prediction



```
          Reference
Prediction black white
     black  2083   392
     white   474  2418
```

Figure 17: Confusion Matrix

On the test set, the metrics are:

- **Accuracy** 83,8%
- **Specificity** 81,4%
- **Sensitivity** 86,0%

The accuracy is slightly better than the tuned decision tree's one.
The specificity decreases by about 3%, but the sensitivity gains around the 4%. The results improved less than expected.

4.3.3 Gradient Boosting

It has been performed the *XGBoost* algorithm, which is a more regularized form of Gradient Boosting.

How it works

Gradient boosting is a technique that produces an additive predictive model combining a series of weak learners, which typically, but not necessarily, are decision trees.
Each predictor tries to improve the previous one reducing the errors. This it actually done fitting a new predictor to the residual errors made by the previous predictor. So it works sequentially.
Fitting small trees to the residuals slowly improves the prediction function in areas where it does not perform well.

Model

The parameters required from the caret package to perform the algorithm are:

- **nrounds**: number of trees, if the number of trees is too high, gradient boosting may overfit.
- **eta**: the learning rate parameter, which controls the rate at which boosting learns
- **max depth**: the number of splits at each tree which control the complexity of the weak learners.
- **subsample**: fraction of the randomly selected training samples that will be used to train each tree
- **colsample by tree**: fraction of randomly selected features that will be used to train each tree.
- **gamma**: Minimum loss reduction required to make a further partition on a leaf node of the tree
- **min child weight**: Minimum sum of instance weight needed in a child

The aim of the last 6 parameters in the list is to control the tendency of this algorithm to overfit.

The parameter tuning made over this parameters selected the following optimal values:

Tuned Parameters	nrounds	eta	max depth	-
Values	1900	0.1	3	-
Constant Parameters	subsample	colsample by tree	gamma	min child weight
Values	0.5	0.5	0	1

Table 11: XGBoost parameter tuning

Figure 18 shows the variable importance for the XGBoost algorithm:
Consistently with Random forest, *takes* is the most important feature. Moreover it seems to be confirmed the pattern for which 6 out of 10 variables concern the course of the match and 3 out of 10 the ratings.
Turns here is the second variable in importance, while *king.moves* has lost importance.
Differently from the random forest, the importance seems to be more balanced between the first features.
Also in this case the variable *victory status* seems not to be much significant, unlike in the decision tree.

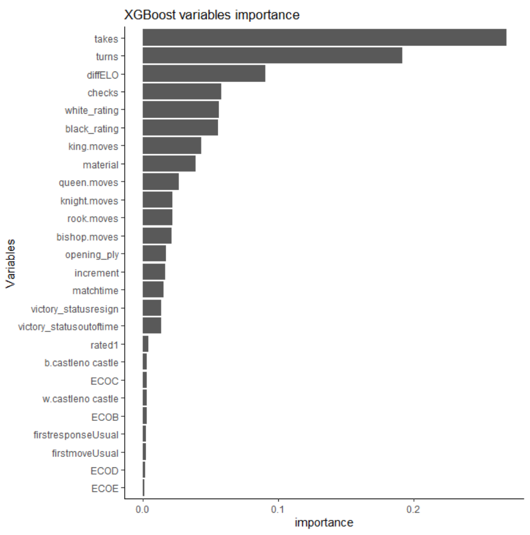
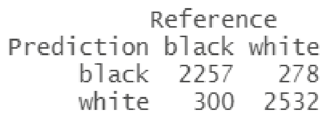


Figure 18: features importance

Prediction



```
          Reference
Prediction black white
     black  2257   278
     white   300  2532
```

Figure 19: Confusion Matrix

On the test set, the metrics are:

- **Accuracy** 89,2%
- **Specificity** 88,2%
- **Sensitivity** 90,1%

Performances have increased significantly compared to those of the random forest and decision tree.
The algorithm is able to classify around 9 games out of 10. Moreover, the computational time of this algorithm proved to be much lower than the random forest's one.
On the other hand, this method has the same problems in *explainability* of the Random forest.

4.4 Logistic Regression

The algorithms seen so far didn't required some specific assumption, the logistic regression instead is sensible to:

- Multicollinearity
- Outliers
- normality between numerical predictors and logit

Theoretical Framework

Logistic Regression can be used for both classification and regression, it estimates the probability of an event occurring. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In the case of a 2-classes problem, it can be defined the threshold at which to consider the observation as "1" rather than "0".
Logistic Regression is prone to overfitting, especially when it has to face high dimensional datasets. Hence it is good practice to use some regularisation methods.

Model Before running the final model, the logistic regression has required some intermediate step:

-First step: use the L1 regularisation to reduce the dimensionality of the dataset. The Lasso reduced the number of variables from 23 to 20, but 5 of these are parts of the categorical variables *ECO* and *Victory status*, which have been encoded.

-Second step: run a first model on the entire dataset, on which to verify the required assumptions.

-Third Step: Check for linearity between numerical features and logit:

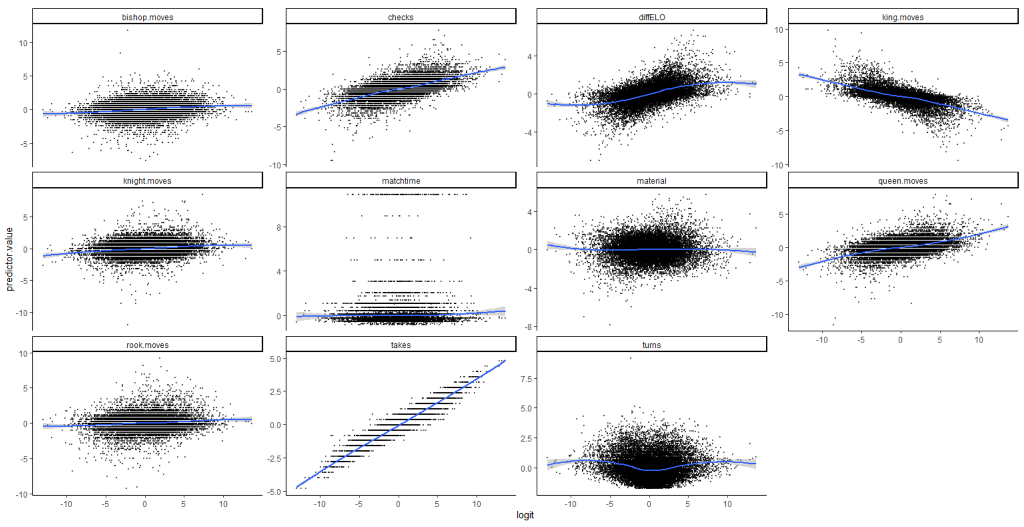


Figure 20: logit vs numerical predictors linearity

They all seem to be rather linearly associated with the response variable in logit scale.

-Fourth Step Outliers: Outliers have been detected using the Cook's distance, with a threshold defined as $t = \frac{4}{Number of Observations}$.
In Figure 21 the Cook's distance plot: the red dots are the observation identified as outliers and the blue line is the threshold. In this phase have been removed 1175 observations.

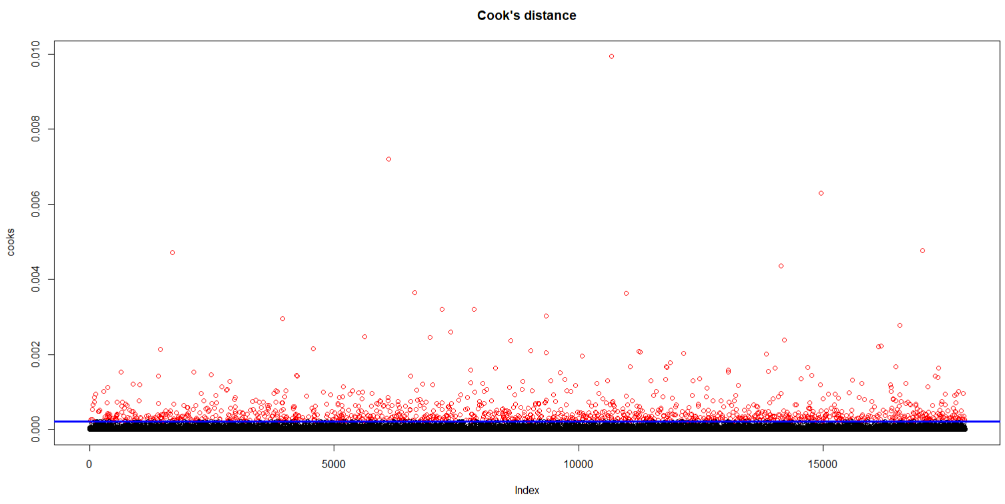


Figure 21: Cook's distance

Fifth Step: MultiCollinearity check:

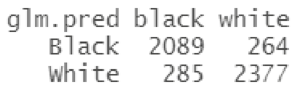
In table 12 can be observed the variance inflation factor of the variables.
VIF score of an independent variable represents how well the variable is explained by other independent variables. Lower the VIF, lower the multicollinearity.
A common rule of thumb states that there is not multicollinearity if
VIF < 5.
It can be observed that the highest VIF in the table is far from this value, therefore no variable has to be removed.

Feature	VIF	Feature	VIF
king.moves	3.300736	turns	1.185958
checks	2.608403	material	1.103580
rook.moves	2.155512	ECO:B	1.081109
queen.moves	2.128206	firstresponse: Usual	1.069516
knight.moves	1.848172	takes	1.053403
bishop.moves	1.625679	ECO: E	1.025244
victory status: outoftime	1.238996	diffELO	1.007080
victorystatus: resign	1.237470	matchtime	1.005010
b.castle	1.188224	-	-

Table 12: VIF standing

Prediction

The modified dataset is now composed of 21 variables and 16720 observations.
Before running the model, it has been divided in training and test set as the original one.
The threshold at which the observations are considered "white" rather than "black", it has been set at 52.4%, which represents the white victory rate in the dataset.



```
glm.pred black white
   Black  2089   264
   White   285  2377
```

Figure 22: Confusion Matrix

On the test set, the metrics are:

- **Accuracy** 88,7%
- **Specificity** 89,3%
- **Sensitivity** 88,4%

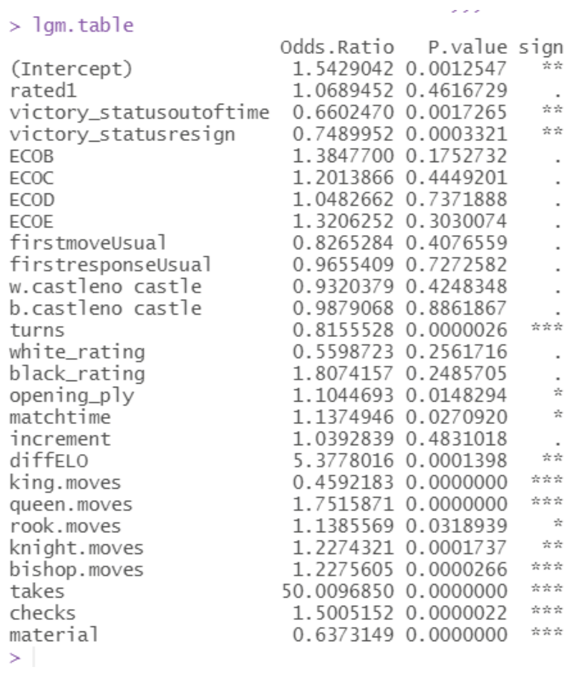
The results are very similar to the ones provided by XGBoost, but they are more explainable.

Results Interpretation

Figure 23 shows the results of the Logistic Regression. *Odds.Ratio* has been obtained taking the exponential of the log(odds), which are the estimates returned by the *glm* function. P-value and sign indicate whether the estimate is significantly different from 0.

The odds ratio are easily understandable: taking as example the variable *checks*: an increase of 1 unit of material multiplies the odds of a White victory by 1.5.
This reasoning has to be made only for the coefficient which are statistically significant, so with an associated P-value < 0.05.
Figure 24 shows the *most influential* variables.
the Plot has been made using the log-odds in order to preserve the values, since the exponential flattens the negative coefficients to 0. The blue bars indicate the positive coefficients, so the coefficient which have Odds Ratio > 1.
On the contrary,the red bars indicate the negative coefficients, the ones which have $0 <$ Odds Ratio < 1.
The most influential variables are *takes*, *DiffELO* and, negatively, *King-moves*. Since these 3 variables are built as differences, the interpretation is not immediate:

- **Takes**: it is the difference between white takes and black takes, which means that the White log odds increase when the difference between these two variables grows. In other words, if white takes a lot of black pieces, and black takes a few white pieces, the white victory chances raise.
- **DiffElo**: Similar interpretation: An increase in the difference between white rating and black rating leads to an increase in the white log odds. But since the difference is weighted, the white chances could raise of the same amount in a white 2400 vs black 2300 games such as in white 1800 vs black 1600 game.
- **king.moves**: This variable is defined as the differences between the white king moves and the black king moves. Which means that the white log odds decrease if white moves the king more than black.



```
> lgm.table
                         Odds.Ratio   P.value sign
(Intercept)               1.5429042 0.0012547   **
rated1                    1.0689452 0.4616729    .
victory_statusoutoftime   0.6602470 0.0017265   **
victory_statusresign      0.7489952 0.0003321   **
ECOB                      1.3847700 0.1752732    .
ECOC                      1.2013866 0.4449201    .
ECOD                      1.0482662 0.7371888    .
ECOE                      1.3206252 0.3030074    .
firstmoveUsual            0.8265284 0.4076559    .
firstresponseUsual        0.9655409 0.7272582    .
w.castleno castle         0.9320379 0.4248348    .
b.castleno castle         0.9879068 0.8861867    .
turns                     0.8155528 0.0000026  ***
white_rating              0.5598723 0.2561716    .
black_rating              1.8074157 0.2485705    .
opening_ply               1.1044693 0.0148294    *
matchtime                 1.1374946 0.0270920    *
increment                 1.0392839 0.4831018    .
diffELO                   5.3778016 0.0001398   **
king.moves                0.4592183 0.0000000  ***
queen.moves               1.7515871 0.0000000  ***
rook.moves                1.1385569 0.0318939    *
knight.moves              1.2274321 0.0001737   **
bishop.moves              1.2275605 0.0000266  ***
takes                    50.0096850 0.0000000  ***
checks                    1.5005152 0.0000022  ***
material                  0.6373149 0.0000000  ***
>
```

Figure 23: Logistic Regression Odds Ratio

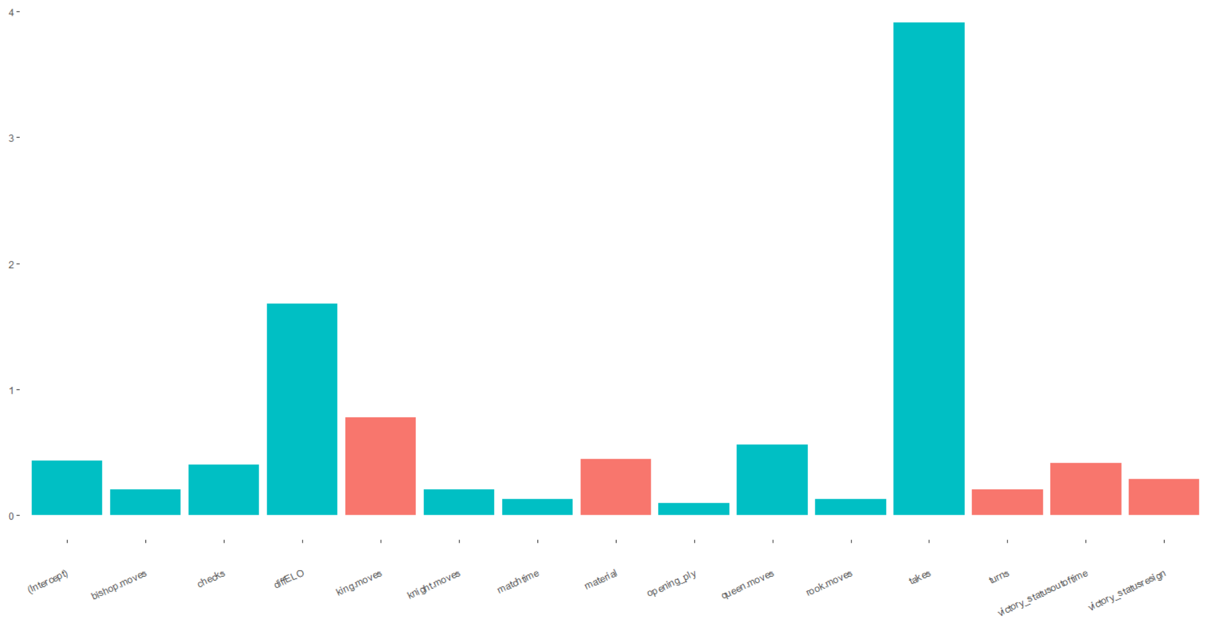


Figure 24: Most influent Variables

5 Conclusions

5.1 Algorithms

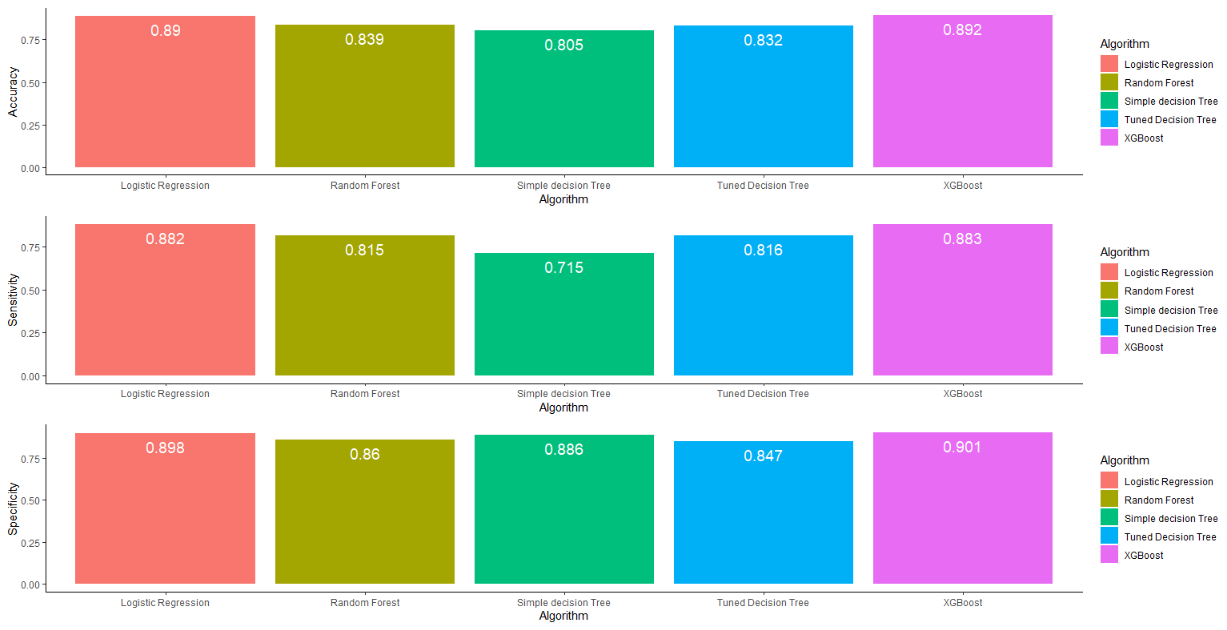


Figure 25: Algorithms performances

Figure 25 shows the algorithms performances in terms of accuracy, sensitivity and specificity.
The best performances are provided by the XGBoost, followed by the Logistic Regression which has only slight worse results.
However, weighting the computational time, the explainability and the comparable metrics, the best model it can be considered the one provided by the Logistic Regression.

5.2 For the Future

It would probably be difficult improve the performances of the algorithms without improving the analysis on the variable *moves*, it could be interesting to collect information about the pawns structure, the king safety, the number of square controlled by each player or the connection of the pieces. However doing this would change the focus more on the evaluation of the positions, which could be the foundation of a chess Engine.
Moreover, it could be interesting to apply the model to a dataset only composed of professional players matches, to study the changes in the performances of the model.

References

[1] Trevor Hastie, Rob Tibshirani, Jerome Friedman *The elements of Statistical Learning*

[2] Andriy Burkov *The Hundred-Page Machine Learning Book*

[3] Dataset: `https://www.kaggle.com/datasets/datasnaek/chess`

[4] FIDE official ELO standings: `https://ratings.fide.com/`

[5] Lichess analysis: `https://lichess.org/analysis`