Universiteit Antwerpen

**2020-2021**
**Machine Learning I**
**De Paepe Robin**
**MA Digital Text Analysis**

# *Report on Machine Learning Project : Predicting the winner of a chess game*

Machine Learning I exam project

The game of chess seems almost made for Machine Learning studies.

It is primarily a game of patterns, and data science is all about detecting patterns in data, which is why chess has been one of the most invested in areas of AI in the past. This project was an attempt at getting acquainted with the endless possibilities of predictive modelling by applying it to one of the most popular and complex board games ever made.

The dataset that has been used is a set of over 20.000 games collected from a selection of users on the site Lichess.org, which is an online platform were chess players can both practice and play chess. These games were collected using the Lichess API, which enables collection of any given users game history.  The dataset provides information about how the game ended, which moves are played, total game time, time increments etc.

For this particular model we decided to focus on the data described below:

- Rated (True/False) ; label
- Number of Turns ;numeric feature
- Winner (Black/White)
  The winner of the chess game. Draws were also noted; categorical feature
- Player rating ; numeric feature
  While the online ratings are not official ratings like FIDE, they work in a similar way. A chess rating is a system used in chess to estimate the strength of a player, based on their performance versus other players. Typical ratings range anywhere from 400 (Amateur/ not rated) to 3000 ( 2500 being a threshold for the title of Grandmaster (GM)).
- Opening Eco:

The Eco is a standardised code for notating documented openings like the Sicilian Defense, Vienna or Caro-Kann. By using the opening Eco, instead of the name of the opening itself, we were able the limit the number of unique values in the feature, which I believe helped the model perform better.

- Opening Ply
    - Number of moves in the opening phase, ; numeric feature

Some other data columns which could  be considered relevant have been purposely left out the model: The dataset also included all moves made in a particular game in standard chess notation. Although certainly useful, I decided to keep this data out of the model to simplify it, as well as keep a possibility open for further exploration of this dataset.

The main objective of the experiment was to attempt to predict the winner of a chess game (being the player with the white pieces or player with the black pieces). Since the dataset provides for a wide array of features to predicts such outcomes and considering that the data can be considered performative, I would have considered this to certainly be possible. The experiment itself can be considered to be mainly build to select, pre-process and finetune the data in order to to provide a model that could make the most accurate prediction possible. Since the data was fairly clean ('no missing values) and the data was relatively homogenic, the biggest task in the pre-processing phase was to encode the data to be able to be pre-processed by the various algorithms considered. For this I used a pipeline which was able to normalize the numerical data, as well as categorical data, using LabelEncoder and StandardScaler into a combined pipeline fit For this we used a pipeline which was able to fit both datatypes using LabelEncoder and StandardScaler.
The pipeline was then used for fitting the data into several standard algorithms. For this model, we tried checking the performance of the data on a Lineal Regression model, Support Vector Machine, Linear Support Vector, DeciscionTreeClassifier and final RandomForestClassifier.
After fitting the data, the initial test showed that the data in all models showed between a 50% to 65% accuracy.  Interestingly, almost all models showed a higher accuracy of predicting wins from white compared to black.

After the initial fitting, I went on to attempt to improve the accuracy of the models by looking at Feature reduction within the model. Although the overall accuracy did not seem to really improve , the DecisciontreeClassifier seemed to perform significantly on a particular set of features, which essentially eliminated all 'skill-based' features out of the data. Seeing as the DeciscionTreeClassifier performed vastly better under this particular feature selection I decided that this algorithm was best suited for the model. Finally, a grid search was executed on the DeciscionTreeClassifier and the second best performing model, which was the RandomForstClassifier. The grid search proved to only provide a marginal improvement over the standard model. Initially the algorithm performed even worse under the grid search due to a faulty selection of hyperparameter values.

Finally I went ahead and trained the model with the final optimized pipeline and executed a prediction test on the test data. The model performed similarly on the test data compared to the development data and attained about an 80% accuracy on the prediction values.

Ultimately the result were not what I was expecting them to be. This is certainly not a bad thing however since I would argue that the predictive model has still shown  some very useful insight into the chess algorithms: In essence, what a simple machine learning model like this is able to accomplish, is to provide a sort of ‘theoretical’ blueprint for wining chess games. If a players chooses a certain opening in a chess game, the model is able to predict with a  relatively high accuracy, if the opening would be winning or not. We explicitly state ‘theoretical’ here, since such accuracy greatly declines once we introduce the rating of both players. This is not surprising since a players rating correlates directly with his theoretical knowledge of chess as well as his tactical insight. Since we have no data about the relationship between matchmaking and the rating of the two players within this dataset, we are simply unable to make conclusions about the impact on the results. Judging from the results when comparing different algorithms, Decision Tree Learning seems to be particularly well suited for this kind of data. Instinctively this makes sense, since the game of chess essentially comes down to a series of ‘if-when’ questions.
I believe that certainly exists a possibility for expanding the accuracy of this model . I believe the feature ‘moves’ that is already present would provide such an improvement. This would require a much deeper depth of the tree learning model since the task of ‘labelling’ certain moves would become infinitely more  complex. However, seeing as machine learning models are actually being used on online chess platforms in real time, I believe it is certainly possible to do this. It has been shown by high-level chess players on these sites, that these models are actually very much based on these theoretical chess plays. Another option would be to Possible introduce the concept of ‘time’ into the model. Different chess formats are used on these online platforms and I cannot help but presume that the format a game has been played in highly correlates with the ‘skill-correlated’ data-features within the set.