Predicting the Outcome of a Chess Game Using the Random Forest and Support Vector Machine Algorithms

Ryne Swift

University of Manitoba

November 30, 2023

Abstract

This project is motivated by and incorporates the methods utilized by Héctor Pulido in his thesis for predicting the outcomes of chess games using machine learning.[1] In this project, machine learning techniques, specifically Support Vector Machines (SVM) and Random Forest algorithms, were used to predict the outcome of chess games after 20 movements for both black and white. The initial step involved downloading a database of chess games, which was accomplished by downloading a database from sourceforge.net.[2] Upon opening the database file with WinZip,[3] the software Scid enabled the selection of a subset of games in the database to be chosen and written into PGN notation,[4] with the subset consisting of games in which at least one player contained the phrase "so" in their name.

The subset of games was subsequently processed and cleaned in Python,[5] written into JSON with the json package in Python,[6] and split into 100 separate files contingent on the white and black players' ratings. In R,[7] and the integrated development editor RStudio,[8] each file was separated into two types of data, training and testing data, with the training data comprising 70% of the file, and the testing data accounting for the remaining 30%. Using the kernlab and randomForest packages in R,[9] [10] the SVM and Random Forest algorithms were applied to the training datasets to create predictive models for determining the outcomes of chess games, and the testing datasets were used to test the models.

[1] https://upcommons.upc.edu/bitstream/handle/2117/106389/119749.pdf?sequence=1&isAllowed=y

[2] https://sourceforge.net/projects/codekiddy-chess/

[3] https://www.winzip.com/en/pages/download/winzip-b3/?x-target=ppc&promo=ppc&utm_source=google&utm_mediu
dd-all-adwordsppc&utm_content=160010506971&utm_term=winzip&utm_id=20624671898&gclid=EAIaIQobChMIqP3R8L-

[4] https://sourceforge.net/projects/scid/

[5] https://www.python.org/

[6] found in Python's Visual Studio Code Integrated Development Editor

[7] https://www.r-project.org/

[8] https://posit.co/download/rstudio-desktop/

[9] https://cran.r-project.org/web/packages/kernlab/index.html

[10] https://cran.r-project.org/web/packages/randomForest/index.html

Contents

List of Figures

List of Tables

1 INTRODUCTION

1.1 Introduction

Due to the increasing power of modern machine learning algorithms, it is now possible to apply to these techniques to real world situations, specifically chess, using personal computers. The objective of this research project is to employ machine learning techniques, namely Support Vector Machines (SVM) and Random Forest, to predict the outcomes of chess games after 20 movements for both players.

The availability of chess libraries and databases online allowed for the download of an extensive chess database of chess games. To extract a dataset from the database, the phrase "so" functioned as a filter to retrieve games in which either the white or black player contained the term in their first or last name, and was selected for its ability to reduce the database into a dataset of satisfactory size.

Errors within the dataset prevented its extraction in Python, which necessitated the cleaning of the dataset prior to its analysis. Chess games that contained errors and games where at least one player's rating was missing were omitted, while additional information to improve our model's predictions for the outcomes of chess games were appended. The cleaned dataset was subsequently partitioned into 100 files depending on the white and black players' ratings to determine if the performance of various models increased within separate files.

The 100 files were subsequently processed into the machine learning algorithms to generate the predictive models. Through analysis of the results of the models, conclusions were generated which will be examined within the context of the predictor variables.

2 Project startup

To commence the process of analyzing chess games and predicting their outcomes, it was necessary to find an appropriate dataset. A database, authored by codekiddy2, was discovered and retrieved from the website sourceforge.net.

2.1 The database

2.1.1 WinZip

The database was formatted in a compressed WinZip file. This made it incompatible with the chess software that was utilized in this project to examine chess datasets and databases, known as Scid (Shane's Chess Information Database). Scid processes datasets that are stored in si4 or PGN format and was also downloaded from sourceforge.net. To convert the database from a compressed WinZip file into a format which is readable by Scid, the WinZip application was downloaded and applied to transform the compressed WinZip file to the si4 format.

2.1.2 Shane's Chess Information Database

The conversion of the database into si4 format allowed for the processing of it in Scid. Upon completion of processing the database in Scid, it was discovered that 6.5 million chess games were contained in the database. The number of games in the database significantly exceeded the amount required to develop training and testing datasets. Consequently, it was necessary to generate a filter to reduce the number of games the datasets consisted of.

The implementation of this filter was performed in Scid using its search function, and the keyword "so" was employed as a filter to diminish the 6.5 million chess games into approximately 498,000 games. The "so" filter retrieved the 6.5 million games from the database and returned the games which had at least one white or black player containing the phrase "so" in their first or last name.

2.1.3 PGN format

The si4 format is solely compatible with Scid, and programs frequently used to analyze and clean datasets, Python being an example, lack the capability to read files in si4 format. Because Python required the file to be in a readable format, it was necessary to convert the dataset containing 498,000 games into PGN format and export the dataset as a PGN file. The transformation of the format and the export of the PGN file were completed through use of the Scid software.

2.2 Elo

Chess players are assigned a numeric value named Elo upon completion of their first game or tournament. Elo provides an approximation of a player's strength, and is used to gauge the likelihood of the outcome of a game between two players. Assuming the Elo of two players accurately reflects their chess strength, a player possessing an Elo higher than their opponent has a greater likelihood of winning a game. For the reduced dataset, it will be assumed that the players' Elos are accurate indicators of their strength. Furthermore, the Elos for both colours in the dataset ranged from 1 to 3000.

3 Data cleansing and properties

Prior to the processing of the dataset into R, it was necessary to clean the dataset and add additional information to it. To clean the dataset, games exhibiting errors which prevented its loading by our software were expunged, and data entry errors were rectified. Upon completion of the tasks, additional information derived from the cleaned dataset was attached to it.

3.1 Python

Python was employed to examine and clean the dataset, and was selected for its libraries and tools which support the processing of files in PGN format. The integrated developer environment employed to run Python was Visual Studio Code.[11] The importation of the dataset into Python resulted in a selection of games producing errors that would halt the Python program. Because of the dataset's substantial size, the games which generated errors were removed in the process of cleaning the dataset, instead of being rectified. Approximately 17,000 chess games which generated errors were removed, resulting in a total of 482540 games in the cleaned dataset. Despite the removal of 17,000 games, the reduction in size of the cleaned dataset from the original dataset was not concerning, due to its significant size. The remaining games within the chess dataset had their columns extracted using functions in the 'chess.pgn' package in Python,[12] and were subsequently written to a JSON file using the 'json' package in Python.

[11] https://code.visualstudio.com/

[12] by unknown author(s), found in Python's Visual Studio Code Integrated Development Editor

3.2 Filtering the dataset

The potential for data entry errors of players' names within the database necessitated the creation of a function to identify and merge two similar names of the same colour. To accomplish this, a function utilizing a mathematical calculation was created to evaluate if two players of the same colour possessed similar names by evaluating the character strings of their names. The function operated by using the 'fuzz.ratio' function from the 'fuzzywuzzy' package in Python to calculate a number between 0 and 100 through a mathematical comparison of the two names.[13] If the calculated value was greater than or equal to a given number the function would recognize the two players as possessing identical names. Furthermore, the function would subsequently merge the two names into a single name and rewrite the names into the dataset. This process was performed for all players in both colours.

A delicate balance had to be struck between being overly meticulous in removing matching names, which would result in names which are similar to not be combined, and being not selective enough, which would result in dissimilar names being recognized as similar. Through exhaustive comparisons of strings, 90 was determined to be a sufficient threshold to achieve this balance.

[13] https://pypi.org/project/fuzzywuzzy/#description

```
print(fuzz.ratio("Ryn Swif", "Ryne Swift"))          # Two letters missing
print(fuzz.ratio("Ryne Swif", "Ryne Swift"))         # One letter missing

print()

print(fuzz.ratio("ryne swift", "Ryne Swift"))        # First letters are lowercase
print(fuzz.ratio("Ryne swift", "Ryne Swift"))        # Last name is lowercase

print()

print(fuzz.ratio("Ryne Swiftt", "Ryne Swift"))       # Extra t
print(fuzz.ratio("Rynee Swiftt", "Ryne Swift"))      # Extra e and t

print()

print(fuzz.ratio("RyneSwift", "Ryne Swift"))         # No space
print(fuzz.ratio("Ryne Swift", "Ryne Swift"))        # Same names
```

```
PROBLEMS 134   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')
89
95

80
90

95
91

95
100
PS C:\Users\ryne0\OneDrive\Documents\R\win-library\4.0\javaCode\Databases\Research Project>
```

Figure 3.1: Calculations of string similarities using fuzz.ratio

3.3 Final dataset

With the cleaned dataset, the process of extracting information from the chess games themselves within Python could now commence. Examples of the details for every game that were extracted from the dataset included: every board position for the games; the evaluation of each board position of a game, which was accomplished through using the 're' package in Python;[14] the average Elo of both black and white players across the dataset; and the results of the game. Subsequently, the necessary details were extracted from the reduced dataset with the assistance of the 'multiprocessing' package in Python,[15] and stored into a

[14] by unknown author(s), found in Python's Visual Studio Code Integrated Development Editor
[15] https://docs.python.org/3/library/multiprocessing.html

JSON file for further analysis in R.

Name	Value
Pawn	1
Bishop	3
Knight	3
Rook	5
Queen	9
King	200

Table 3.1: Values of chess pieces

fen_eval	white
sum_eval	white_win
white_elo	white_draw
black_elo	white_lose
black_avg_elo	black_win
black	black_draw
result	black_lose
white_avg_elo	

Table 3.2: Columns of the dataset

An explanation of the Table 3.2's variables is given below.

fen_eval: A list of the numerical evaluations of the game's board positions.

sum_eval: The summation of the numerical values of the fen_eval list for each game.

white_elo: The Elo of the white player in the game.

black_elo: The Elo of the black player in the game.

black_avg_elo: The average Elo of the black player across every game they played as black within the dataset.

black: The name of the black player.

result: The outcome of the game.

white_avg_elo: The average Elo of the white player across every game they played as white

within the dataset.

white: The name of the white player.

white_win: The cumulative frequency of wins as white for the white player across the dataset.

white_draw: The cumulative frequency of draws as white for the white player across the dataset.

white_lose: The cumulative frequency of losses as white for the white player across the dataset.

black_win: The cumulative frequency of wins as black for the black player across the dataset.

black_draw: The cumulative frequency of draws as black for the black player across the dataset.

black_lose: The cumulative frequency of losses as black for the black player across the dataset.

3.4 Limitations of the data and the hardware

3.4.1 Limitations within the data

The majority of the dataset consisted of real games, yet games existed which contained significant flaws. A selection of games was extracted from studies within chess books, meaning that these games were fictional. Furthermore, there is also the possibility that the games contained players who willingly did not play to the best of their ability, either through a lack of care and effort, or through bribery, which would lead to the outcome of the game being pre-determined. Modern computer chess programs could also be used to provide an unfair advantage that could potentially alter the game's outcome. For example, a player may take a bathroom break to consult their phone's chess program to inform them of the best

move. These flawed games were not uniquely indicated in the database, and because of the considerable size of the dataset, manually filtering these games out would cost a significant amount of time.

3.4.2 Hardware limitations

The dataset's size, in conjunction with the demands and structures of the necessary functions to extract information, caused its processing to be computationally expensive. The processing time was further extended by mistakes in the Python code, notably when such errors arose after running a program for a span of minutes or hours. Furthermore, due to the running of the Python programs on a personal computer, the programs terminated when the user inadvertently powered off the computer.

4 Data consolidation

4.1 Removal of short games

Because the primary objective of the project is to predict the outcomes of chess games after 20 movements by the black and white players, it was imperative to remove games from our dataset which failed to meet the required threshold. The removal of these games resulted in a reduction of our dataset by approximately 20,000 games. However, the exclusion of these games would have a negligible impact considering the significant size of the dataset. The final dataset prior to its separation contained 233,408 games.

4.2 Separating the dataset

The dataset was loaded into R using the 'jsonlite' package,[16] and future JSON datasets loaded into R were as well. To determine which white and black ratings would be most effectively predicted by the machine learning algorithms, 12 numbers were discovered using the "cut_number" function in R's "ggplot2" library.[17] These numbers would divide the dataset into 10 approximately even segments of data depending on the white player's rating. The numbers were subsequently entered into Python to partition the dataset into 10 segments, with the greatest of the white players' ratings going into one segment and the lowest into another. Subsequently, this process was repeated for the black players' ratings, and "cut_number" provided differing sets of 12 numbers for each of the 10 portions. Each of the 10 segments was thereafter divided into 10 more portions in Python, and written to a JSON file for future analysis and extraction into R. The plots of the 10 white grid ranges and 100 black and white grid ranges are presented below.

[16] https://cran.r-project.org/web/packages/jsonlite/index.html
[17] https://cran.r-project.org/web/packages/ggplot2/index.html

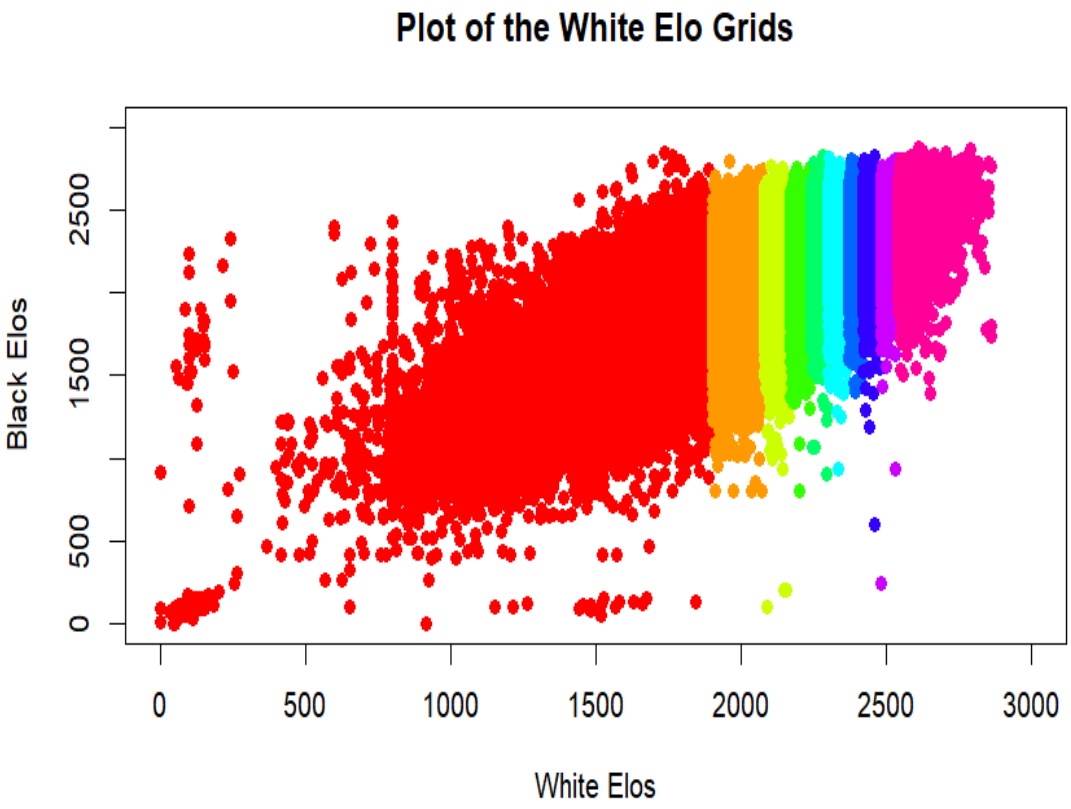

Figure 4.1: Plot of the Elos within the white grids

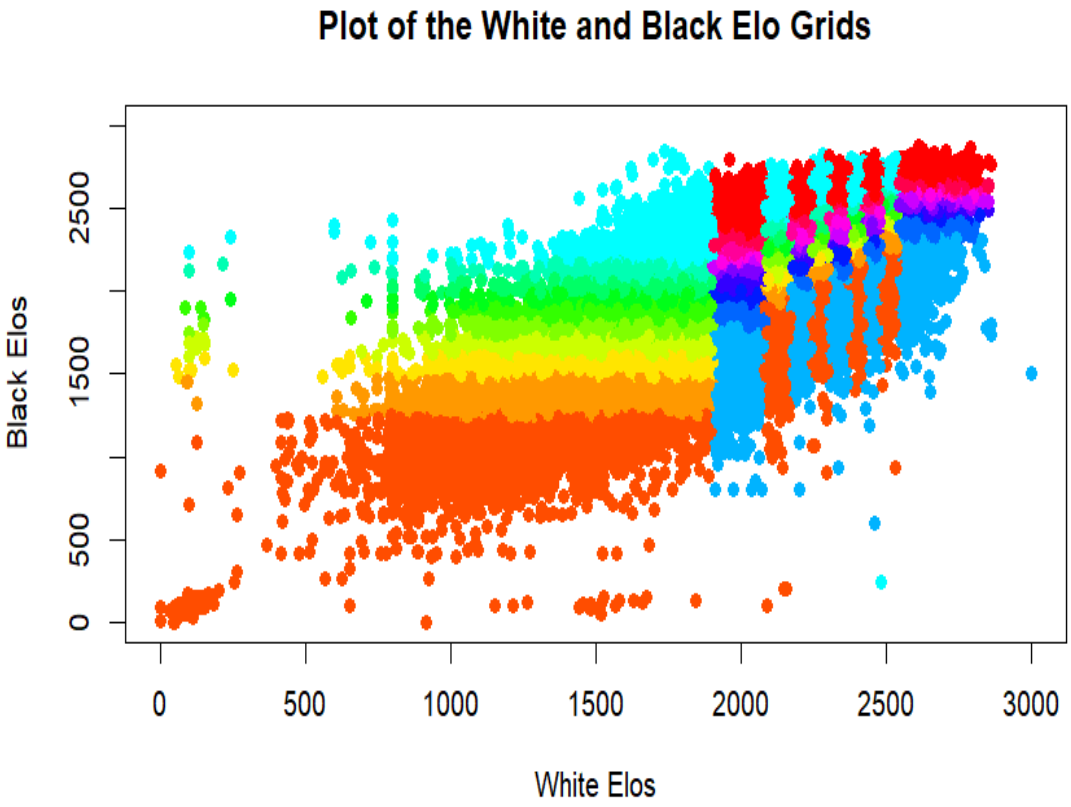

Figure 4.2: Plot of the Elos within the white and black grids

4.3 ChatGPT

ChatGPT was instrumental in assisting the diagnosis of errors and suggesting solutions while importing and cleaning the dataset.[18] It was consistently utilized throughout the project and expedited its completion by providing prompt answers to questions that would otherwise cost substantial time to resolve. The code which was recommended by ChatGPT and implemented into R or Python contained comments to provide appropriate credit to it.

[18]https://chat.openai.com/

5 Model implementation

5.1 Types of models

Two types of models for the Random Forest and SVM algorithms were constructed: models which incorporated wins, draws, and losses as possible outcomes; and models which incorporated wins and losses only as possible outcomes. This was conducted to determine how the algorithms would perform on the datasets given the presence or absence of draws as an outcome. The training and testing datasets for the models without draws contained two extra columns which were not assigned to the datasets of the models with draws.

5.2 Mechanics of the algorithms

5.2.1 Random Forest mechanics

A random subset of the data is sampled with replacement to ensure that each tree is trained using different data. Within each tree, a random subset of predictors is selected to avoid the introduction of bias in the results, which are the predicted outcomes of our chess games. The results of the trees are subsequently utilized by counting the most frequently observed result among the trees, which will be used to provide the final prediction for the dataset.[19] Furthermore, the function in R used for the implementation of Random Forest selects the model with the lowest out of bag error, which is the prediction error for Random Forest.

[19]New Machine Learning Algorithm: Random Forest

5.2.2 SVM mechanics

For the win and loss models, a line is created to separate the points within the training dataset into two classifiers, composing a binary decision of outcomes. If a data point resides within the first classifier, its predicted outcome is of the first classifier, and similarly, if it resides within the second classifier its predicted outcome will be of the second classifier.[20] Given that some of our models will encompass win, draw and loss as an outcome for a chess game, 3 different pairs of classifier will be made for these models, and the classifier among the three which is most frequently chosen will be the assigned outcome for a given training data point.[21] Moreover, the function in R which implements SVM selects the simulated model containing the least error.

5.3 Parameters of the Random Forest and SVM algorithms

To minimize model error, an attempt to determine the optimal number of trees to use in Random Forest was performed using the 10 white datasets. The endeavor involved using tree values which increased in increments of 1 from 50 to 300, inclusive. The function used to generate the Random Forest models was modified to present the best three tree parameters which yielded the lowest errors. However, the operation of the Random Forest algorithm on a subset of the 10 datasets failed to reach a consensus on the optimal number of trees to use. Due to this, R was allowed to choose the models with minimized errors by utilizing tree values which increased in increments of 1 ranging from 50 to 300, inclusive. A similarly extensive procedure was not applied to the SVM models because only five cost values (0.01, 0.1, 1, 10, and 100) were utilized to discover the value which yielded the lowest error.

[20] https://online.stat.psu.edu/stat857/node/211/
[21] https://online.stat.psu.edu/stat857/node/244/

5.4 Further dataset manipulation

5.4.1 Appending additional columns

Before the application of the machine learning algorithms to the dataset, additional columns were appended to the datasets to improve the future models. Our datasets contained columns that included draws, and as a result, these columns were excluded from the models without draws. It is anticipated that the inclusion of these columns will improve the accuracy of the predictions of the outcomes of chess games.

Columns added for both models	Description
Delta Average Elo	Difference of the white and black average Elos
Percentage Results	Percentage of results for fen, white and black
Delta Percentage	The percentage difference between win and lose and draw for white and black
Columns Added for Win and Loss Only	**Description**
Delta White	Difference between wins and losses for white
Delta Black	Difference between wins and losses for black

Table 5.1: Additional columns

5.4.2 Creation of training and testing datasets

To create our models, it was imperative to separate our datasets into testing and training data. Individual games within the datasets were randomly assigned within R to serve as either testing or training data, with approximately 70% of the original datasets designated for testing, and the remaining 30% for training.

5.5 ROC curve to justify Random Forest

Prior to the implementation of the Random Forest function on the datasets, it was necessary to determine if the Random Forest win and loss models created with the reduced datasets performed as effectively as the win and loss models created with the full datasets. This was a necessity due to how the Random Forest algorithm operates in R. To determine if the reduced datasets performed as effectively as the full datasets, plots of ROC (Receiving Operation Characteristic) curves were generated in R for both datasets using the roc function in the pROC library.[22]

The ROC curve can only be generated for binary data, limiting the plotting of the models' results to the win and loss models only. The ROC curve functions by plotting a line, and the proximity of the plotted line to the specificity and sensitivity values influences the value of the AUC (area under curve) value. A greater AUC value indicates a stronger performance of the model on the testing data. Two plots of ROC curves were constructed using the data of the entire dataset. The initial plot displays the ROC curve which employed the whole dataset, while the second plot displays the ROC curve which used a sample of the dataset.

[22]https://cran.r-project.org/web/packages/pROC/index.html

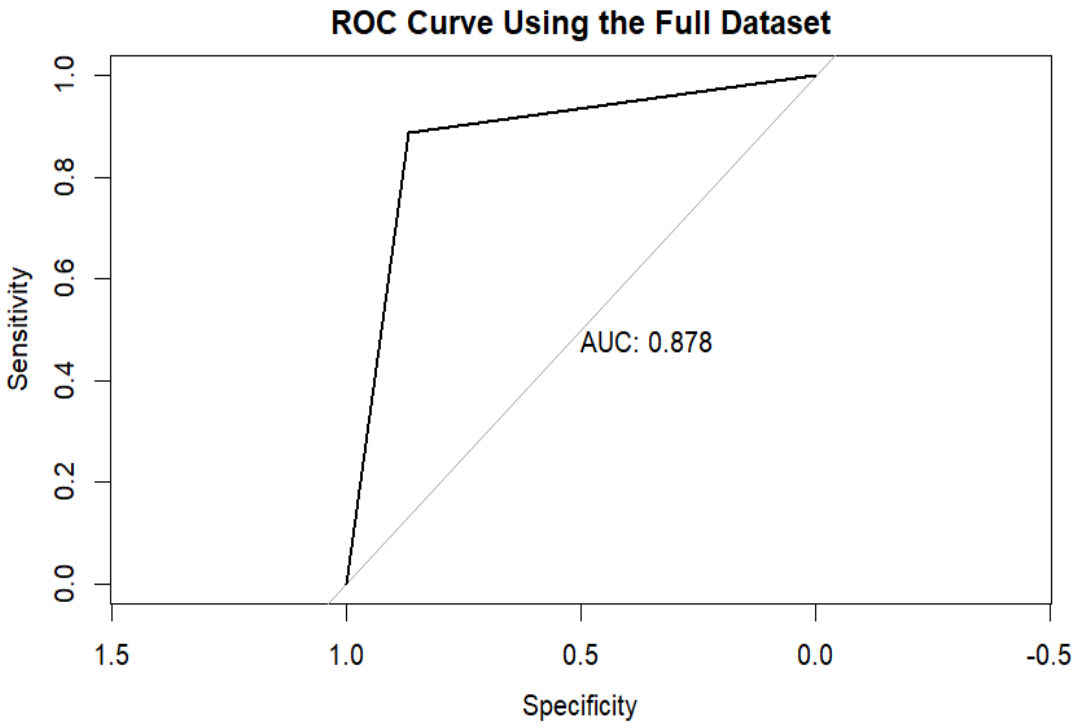

Figure 5.1: Plot of the ROC curve using the full dataset

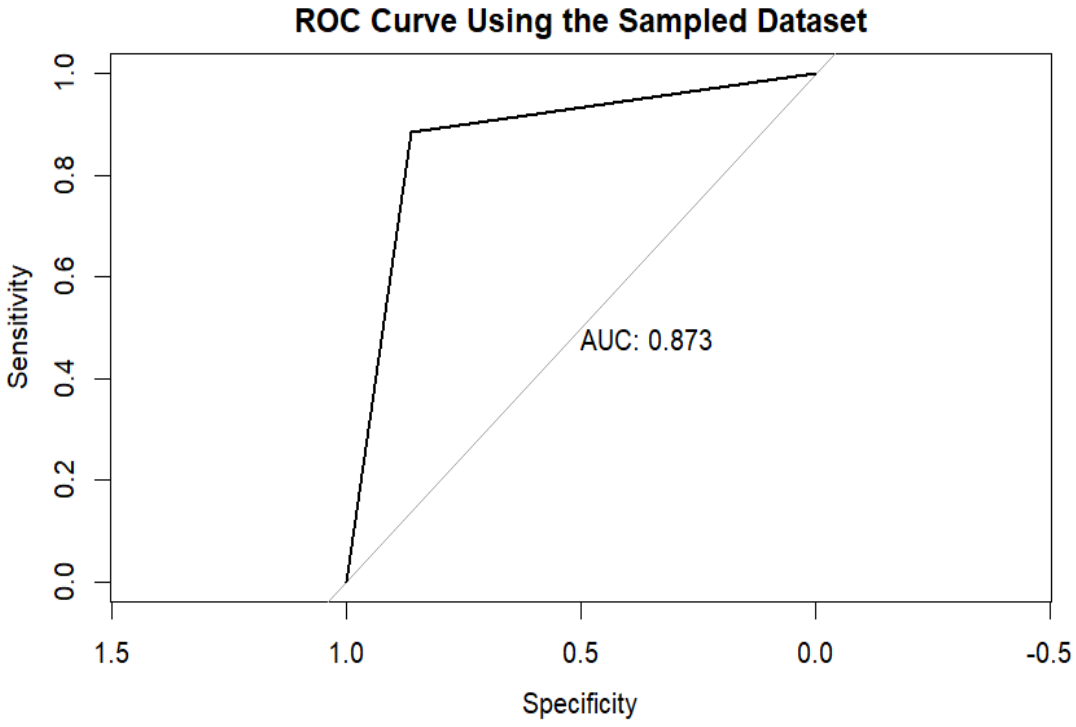

Figure 5.2: Plot of the ROC curve using the sampled dataset

From Figures 5.1 and 5.2, an AUC value of 0.878 for the full dataset and an AUC value of 0.873 for the sampled dataset are observed. The difference in AUC values between the full and sampled datasets is negligible. Due to this, we can safely use the reduced datasets.

5.6 Construction of the models

Because of the variability of Random Forest and SVM results across different seeds, the specific seed 427 was utilized to generate all models. The Random Forest and SVM functions were used to determine the model with the lowest error for each training dataset, and this was performed by generating models for each parameter value input into it. The models would subsequently be compared to determine which contains the lowest error, and the model possessing the lowest error would be selected as the optimal performer for the training dataset. By selecting the models containing the lowest error, the predictions for the outcomes of chess games will be improved.

5.7 Results

To determine the parameters utilized by our models, and the models' performance when applied to testing data, a data frame was created in R. The data frame displayed the model parameters and the results of the predictions on the test data for each dataset. For each dataset, the columns included in the data frame were: the model's error, the percent accuracy of the model on testing data for predicting black wins, the percent accuracy of the model on testing data for predicting draws, the percent accuracy of the model on testing data for predicting white wins, and the parameter value chosen to create each model. The parameter value consisted of either a tree value or a cost value, depending on which algorithm was used,

Random Forest or SVM. Furthermore, the column for the percent accuracy of the model on testing data for predicting draws was only included for the win, draw and loss model. Sample rows from the data frames for the models are presented below, in conjunction with plots of their errors.

Grid	errors	% Black Correct	% Draw Correct	% White Correct	Tree Used
2	0.10	89.52	92.31	87.46	63
3	0.11	86.48	90.09	84.51	67
33	0.28	71.90	67.61	73.04	200
41	0.19	89.29	92.91	78.75	96
61	0.15	100.00	93.94	82.96	98
70	0.33	67.76	67.53	100.00	222
75	0.39	58.00	53.92	62.57	116
85	0.38	72.34	55.86	63.37	117

Table 5.2: Random Forest model results with draws

Grid	errors	% Black Correct	% White Correct	Tree Used
2	0.07	93.48	92.42	84
3	0.07	91.24	92.92	100
33	0.14	91.41	84.20	148
41	0.09	75.61	91.70	77
61	0.06	100.00	93.66	58
70	0.13	84.58	70.00	87
75	0.19	81.82	85.75	252
85	0.16	80.33	85.57	207

Table 5.3: Random Forest model results without draws

Grid	errors	% Black Correct	% Draw Correct	% White Correct	Cost Used
2	0.08	85.31	90.79	87.70	10
3	0.11	84.40	88.84	82.67	1
33	0.23	73.53	68.60	72.43	1
41	0.16	75.00	91.03	78.47	1
61	0.11	62.50	84.00	86.53	10
70	0.25	69.33	64.10	36.73	10
75	0.35	61.33	56.44	61.89	1
85	0.33	60.87	59.48	63.67	1

Table 5.4: SVM model results with draws

Grid	errors	% Black Correct	% White Correct	Cost Used
2	0.06	89.16	94.96	10
3	0.06	90.73	88.05	10
33	0.13	83.76	85.19	1
41	0.06	81.40	93.46	10
61	0.04	89.47	93.76	1
70	0.11	89.20	66.67	10
75	0.14	69.35	80.00	10
85	0.14	80.30	83.37	1

Table 5.5: SVM model results without draws

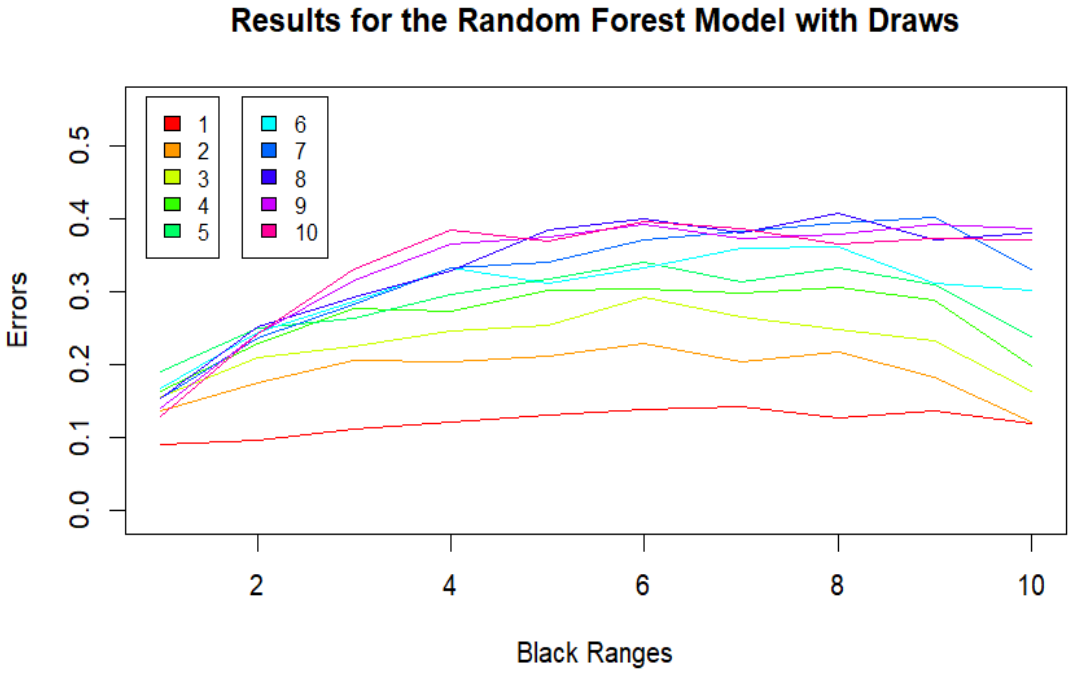

Figure 5.3: Plot of the errors of the Random Forest models with draws.

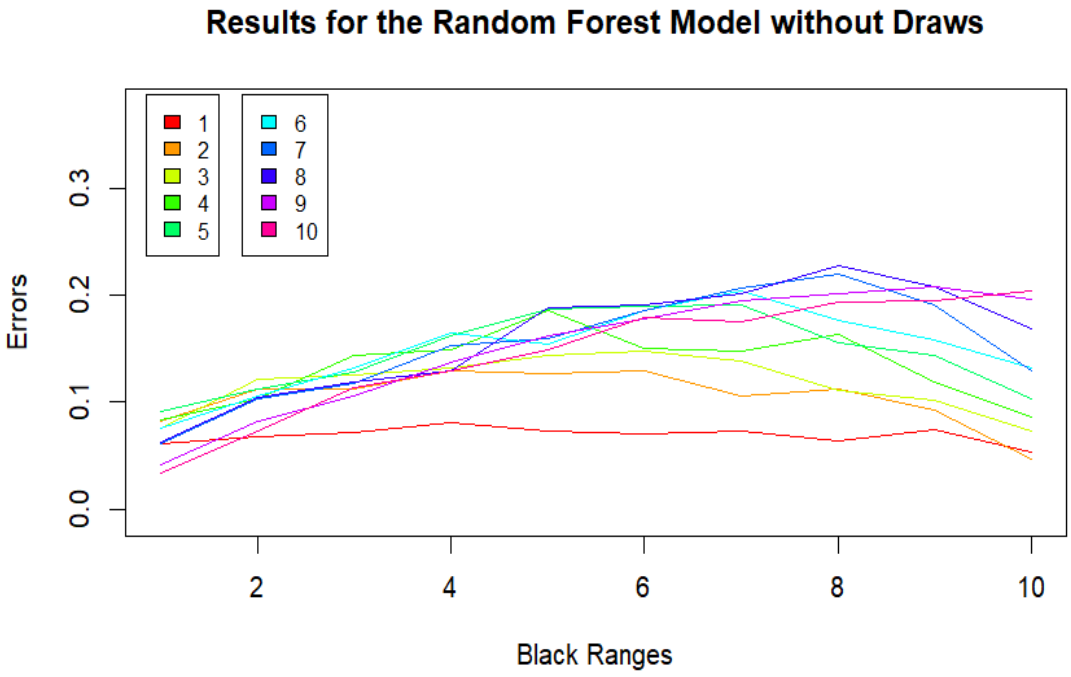

Figure 5.4: Plot of the errors of the Random Forest models without draws.

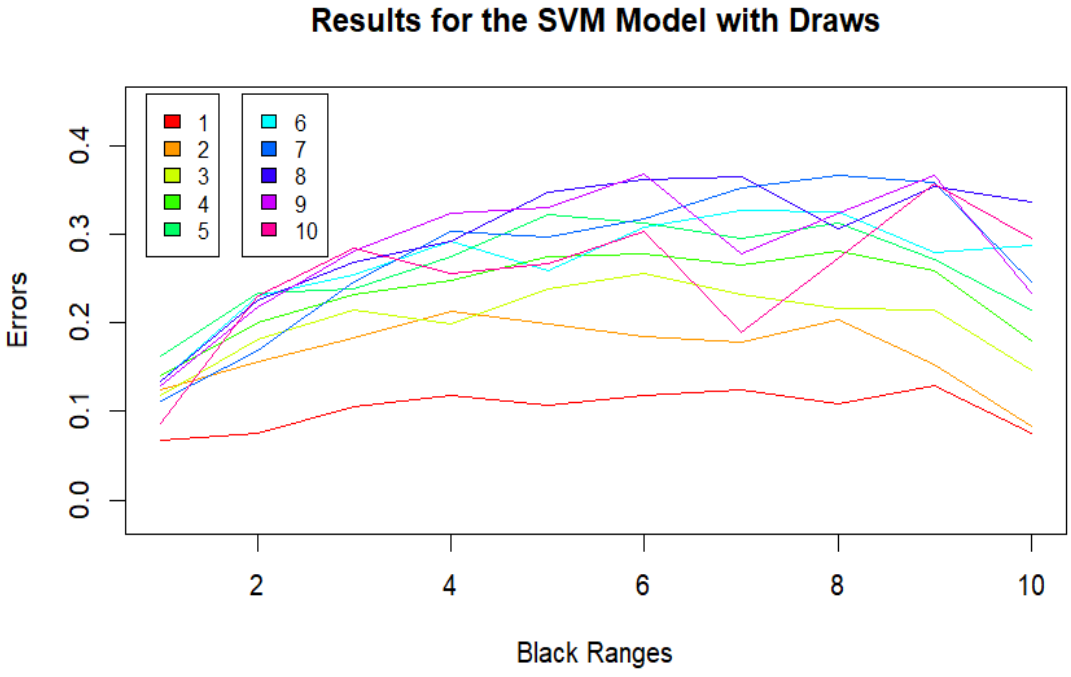

Figure 5.5: Plot of the errors of the SVM models with draws.

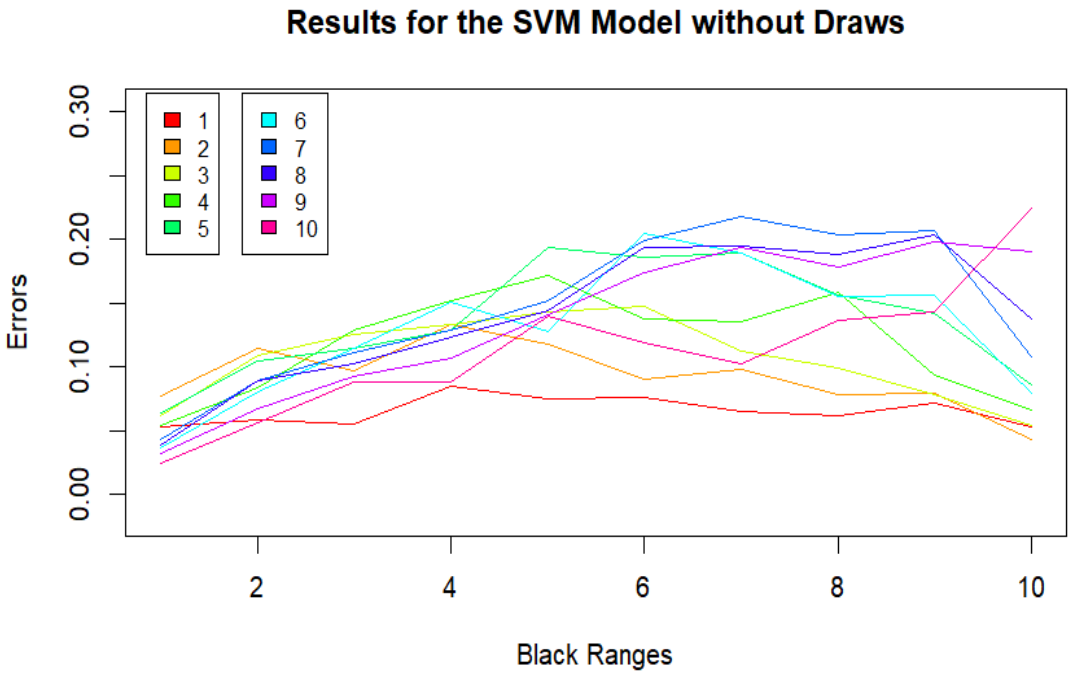

Figure 5.6: Plot of the errors of the SVM models without draws.

The complete data tables can be found in the appendix, and the export of the data tables was performed through the knitr package in R.[23] The colour representation on the plots demonstrate the errors for each white grid across their 10 black subgrids, with the errors of the first white grid being coloured red, the second orange, and so forth.

Visual examination of the plots reveals that the errors for the majority of white range models demonstrates a tendency to increase as the black range increases. It is of interest to note that the first white grid range remains consistently stationary across the black ranges for every type of model. The initial three white grid ranges contain the least errors across the models, while the final two white grid ranges consist of the most. Finally, the models without draws contain smaller errors than the models which include draws, and the Random Forest and SVM models performed similarly.

[23] https://cran.r-project.org/web/packages/knitr/index.html

The model errors consistently increase from the lowest white grid number, which contains the least errors, to the highest white grid numbers, which yield the greatest errors. This phenomenon is occurring because of a change in the importance of predictor columns. For instance, the column delta_average_elo will remain a reliable predictor across each white grid range. Conversely, columns such as sum_eval will begin as a reliable predictor for the lower grid ranges, but transition to a frequently unreliable predictor in the higher grid ranges.

The oscillations in the reliability of columns are attributed to the different skill level of lower and higher elo games within greater white elo and black elo grids. The rise in the Elo ratings of the two players creates an increased likelihood for a player to resign when they face a material disadvantage, producing a small sum_eval. Furthermore, it occurs with increased frequency in higher Elo games where sum_eval and fen_eval are 0, yet a player has a decisive advantage on the board, leading to the other player's resignation.

6 Conclusion

The objective of this project was the development of models to predict the outcomes of chess games. To achieve this goal, a database was identified, refined to create a reduced dataset which was subsequently cleaned and had additional details pertaining to the games appended. The dataset was separated into 100 segments, and each of which was subsequently partitioned into training and testing data. The generation of the training and testing data facilitated the possibility for the creation of the datasets' predictive models in R.

Using the datasets, the objective of the project was successfully obtained, and the generated models contained errors which were satisfactory. Upon examination of the error plots for the models, no significant difference between the Random Forest and SVM models was observed. However, a difference existed between the models which incorporated draws and the models which did not. The models utilized to predict the testing datasets with lower white Elos demonstrated superior performance than those used to predict testing datasets with higher Elos, which was due to the different skill levels of the white player.

Areas requiring improvement are the employment of a larger dataset, given that each of the 100 individual datasets contained an average number of approximately 2000 games. Moreover, it is conceivable that other machine learning algorithms will yield models with superior performance to the Random Forest and SVM models. Due to this, this possibility should be investigated. For the greater Elo ranges, the application of chess engines to correctly evaluate the board positions may also be applied. Such an approach would cost significant computation time, and should thus be exclusively applied to datasets containing large Elos. However, the inclusion of chess engines would create a variable which would serve as an excellent predictor for high Elo matches.

Appendices

Appendix A

A. Subgrid information

A.1 Number summary and genuine outcomes

Grid	% White Win	% Draw	% Black Win	mean	median
1	56.79	10.88	32.33	68.58	64
2	53.76	15.85	30.39	73.74	69
3	46.13	18.62	35.25	74.78	71
4	41.41	22.63	35.95	75.66	72
5	34.20	23.43	42.37	76.42	73
6	28.69	23.14	48.16	76.43	72
7	22.42	24.92	52.67	77.67	75
8	19.83	24.77	55.40	77.66	74
9	15.07	22.59	62.33	77.74	74
10	9.07	15.60	75.33	75.27	72
11	70.00	17.80	12.21	74.00	69
12	56.11	23.97	19.92	76.17	73
13	42.13	30.16	27.71	78.00	74
14	32.39	33.26	34.35	77.69	75
15	27.04	32.03	40.93	76.86	75
16	23.59	32.12	44.29	78.15	75
17	20.19	28.98	50.83	78.87	77
18	16.19	25.54	58.27	81.33	78
19	13.32	20.68	66.00	82.06	78
20	6.48	13.84	79.68	78.38	74
21	69.16	18.84	12.00	75.43	71
22	53.87	28.62	17.51	77.54	73
23	48.06	29.97	21.96	77.97	74
24	38.14	33.30	28.56	78.59	76
25	30.79	35.84	33.37	79.62	77
26	25.69	34.43	39.88	77.67	76
27	23.16	32.94	43.90	80.10	78
28	17.52	30.29	52.19	81.42	79
29	13.14	26.64	60.21	82.26	79
30	7.84	17.97	74.18	79.47	75

Grid	% White Win	% Draw	% Black Win	mean	median
31	70.96	19.05	9.99	75.68	71
32	59.83	25.90	14.27	78.77	75
33	48.78	31.13	20.09	78.31	75
34	40.34	33.73	25.93	77.97	75
35	35.26	36.13	28.61	78.01	76
36	27.44	37.15	35.41	79.93	79
37	22.70	36.17	41.13	80.30	78
38	18.17	35.57	46.25	80.59	78
39	14.38	31.31	54.31	81.11	78
40	8.30	21.74	69.95	81.02	78
41	71.40	18.74	9.87	76.80	73
42	59.01	26.12	14.87	79.12	75
43	49.63	31.49	18.88	79.65	77
44	42.56	35.90	21.55	79.03	77
45	35.71	38.83	25.47	76.57	75
46	29.82	39.75	30.42	77.04	76
47	24.53	41.11	34.37	78.36	76
48	18.81	39.31	41.88	79.76	78
49	15.78	35.71	48.51	80.24	78
50	9.13	25.50	65.37	81.44	79
51	78.27	14.91	6.82	76.54	72
52	65.43	23.60	10.97	80.33	75
53	55.15	30.34	14.51	81.07	79
54	47.95	34.35	17.70	79.59	77
55	39.19	38.04	22.77	78.66	77
56	33.86	42.75	23.39	75.10	73
57	23.41	47.40	29.18	75.90	75
58	17.61	45.39	37.00	77.78	78
59	17.91	37.43	44.66	81.02	78
60	10.56	33.12	56.32	83.88	80
61	80.04	14.22	5.75	76.40	73
62	65.64	24.06	10.30	80.94	78
63	59.53	28.65	11.82	79.62	77
64	50.68	33.94	15.38	79.21	76
65	42.08	41.36	16.55	78.41	77
66	31.33	49.62	19.05	72.78	72
67	25.87	51.03	23.09	74.02	73
68	21.04	48.78	30.18	73.82	73
69	17.04	46.03	36.93	77.80	76
70	11.11	41.16	47.73	81.13	80

Grid	% White Win	% Draw	% Black Win	mean	median
71	81.54	12.64	5.82	75.77	72
72	66.57	23.55	9.88	80.96	77
73	60.14	28.75	11.11	82.02	78
74	53.05	34.93	12.02	80.53	78
75	41.16	44.16	14.68	77.55	76
76	34.38	49.31	16.31	74.58	73
77	26.20	53.10	20.70	72.77	72
78	22.28	52.72	25.00	74.25	73
79	18.28	51.01	30.71	77.44	76
80	12.87	48.29	38.85	79.52	76
81	84.66	12.00	3.34	74.85	71
82	70.21	22.63	7.16	80.31	77
83	61.70	28.75	9.54	82.40	79
84	53.23	34.54	12.23	79.77	77
85	43.55	43.96	12.49	77.98	75
86	38.30	46.02	15.68	77.61	76
87	28.93	52.50	18.57	76.82	76
88	26.38	52.72	20.90	76.40	75
89	20.78	56.85	22.37	75.99	75
90	16.04	52.34	31.62	80.02	78
91	85.90	10.61	3.49	76.35	72
92	70.87	23.06	6.06	82.29	78
93	59.55	31.61	8.84	83.72	79
94	50.97	39.60	9.43	83.27	79
95	44.90	44.17	10.93	81.07	79
96	40.02	48.56	11.42	78.97	77
97	34.60	52.42	12.98	79.05	76
98	28.35	56.25	15.40	77.53	76
99	26.87	55.88	17.25	81.92	79
100	22.61	55.50	21.89	83.15	79

Table A.1: Number summary and genuine outcomes

A.2 Datasets' sizes and Elo ranges

Grid	Games	White minimum	White maximum	Black minimum	Black maximum
1	2375	1	999	1	999
2	2391	1000	999	1251	1480
3	2312	100	989	1481	1610
4	2474	100	993	1611	1730
5	2448	100	999	1731	1820
6	2296	1005	931	1821	1910
7	2288	1045	995	1911	1990
8	2511	1000	971	1991	2070
9	2355	100	910	2071	2170
10	2334	100	939	2171	2840
11	2214	1921	2080	1000	962
12	2315	1921	2080	1781	1910
13	2184	1921	2080	1911	2010
14	2407	1921	2080	2011	2080
15	2234	1921	2080	2081	2130
16	2046	1921	2080	2131	2170
17	2476	1921	2080	2171	2220
18	2101	1921	2080	2221	2270
19	2003	1921	2080	2271	2350
20	2279	1921	2080	2351	2795
21	2547	2081	2180	100	939
22	2584	2081	2180	1931	2030
23	2498	2081	2180	2031	2090
24	2745	2081	2180	2091	2150
25	2448	2081	2180	2151	2200
26	2418	2081	2180	2201	2240
27	2879	2081	2180	2241	2290
28	2349	2081	2180	2291	2340
29	2666	2081	2180	2341	2410
30	2692	2081	2180	2411	2767

Grid	Games	White minimum	White maximum	Black minimum	Black maximum
31	1995	2181	2240	1090	800
32	1966	2181	2240	2011	2080
33	2341	2181	2240	2081	2150
34	1879	2181	2240	2151	2200
35	2153	2181	2240	2201	2250
36	2219	2181	2240	2251	2300
37	1834	2181	2240	2301	2340
38	2003	2181	2240	2341	2390
39	2144	2181	2240	2391	2450
40	2051	2181	2240	2451	2757
41	2420	2241	2310	1071	911
42	2834	2241	2310	2061	2140
43	2667	2241	2310	2141	2200
44	2190	2241	2310	2201	2240
45	2769	2241	2310	2241	2290
46	2745	2241	2310	2291	2340
47	2190	2241	2310	2341	2380
48	2434	2241	2310	2381	2420
49	2682	2241	2310	2421	2480
50	2595	2241	2310	2481	2817
51	2058	2311	2370	1253	936
52	2307	2311	2370	2101	2180
53	2183	2311	2370	2181	2230
54	2358	2311	2370	2231	2280
55	1826	2311	2370	2281	2320
56	2077	2311	2370	2321	2370
57	2496	2311	2370	2371	2420
58	1891	2311	2370	2421	2460
59	2203	2311	2370	2461	2520
60	2046	2311	2370	2521	2773
61	2660	2371	2430	1293	2150
62	2541	2371	2430	2151	2220
63	2613	2371	2430	2221	2270
64	2793	2371	2430	2271	2320
65	2197	2371	2430	2321	2360
66	2230	2371	2430	2361	2400
67	3019	2371	2430	2401	2450
68	2665	2371	2430	2451	2500
69	2123	2371	2430	2501	2540
70	2598	2371	2430	2541	2800

Grid	Games	White minimum	White maximum	Black minimum	Black maximum
71	2151	2431	2490	1188	600
72	2583	2431	2490	2201	2280
73	1790	2431	2490	2281	2320
74	2364	2431	2490	2321	2370
75	2335	2431	2490	2371	2410
76	2381	2431	2490	2411	2450
77	1987	2431	2490	2451	2490
78	2289	2431	2490	2491	2530
79	2261	2431	2490	2531	2570
80	2142	2431	2490	2571	2820
81	2605	2491	2560	1500	936
82	2599	2491	2560	2261	2340
83	2375	2491	2560	2341	2390
84	2032	2491	2560	2391	2420
85	3000	2491	2560	2421	2460
86	2905	2491	2560	2461	2500
87	1608	2491	2560	2501	2520
88	3164	2491	2560	2521	2560
89	2339	2491	2560	2561	2600
90	2315	2491	2560	2601	2805
91	2154	2561	3000	1390	2330
92	2331	2561	2801	2331	2420
93	1804	2561	2820	2421	2460
94	2296	2561	2851	2461	2500
95	2217	2561	2831	2501	2530
96	2246	2561	2851	2531	2560
97	2203	2561	2827	2561	2590
98	1931	2561	2812	2591	2620
99	1898	2561	2851	2621	2650
100	2235	2561	2862	2651	2876

Table A.2: Datasets' sizes and Elo ranges

B. Models information

B.1 Random Forest model results without draws

Grid	errors	% Black Correct	% White Correct	Tree Used
1	0.06	92.95	92.61	139
2	0.07	93.48	92.42	84
3	0.07	91.24	92.92	100
4	0.08	88.85	91.63	145
5	0.07	89.41	93.83	255
6	0.07	94.07	90.91	167
7	0.07	92.62	96.49	104
8	0.06	91.12	93.55	69
9	0.07	92.60	97.62	64
10	0.05	92.11	100.00	52
11	0.08	92.16	93.58	62
12	0.11	87.50	90.34	201
13	0.11	85.13	84.02	94
14	0.13	85.45	90.76	206
15	0.13	85.16	92.25	82
16	0.13	90.40	92.63	79
17	0.11	87.89	89.29	61
18	0.11	87.53	86.67	178
19	0.09	87.67	95.24	134
20	0.05	93.74	NA	77
21	0.08	90.62	91.69	120
22	0.12	81.03	91.69	99
23	0.13	84.85	87.17	200
24	0.13	87.92	82.21	168
25	0.14	85.71	87.83	163
26	0.15	85.94	82.07	249
27	0.14	89.28	83.22	75
28	0.11	86.20	90.00	154
29	0.10	91.25	93.33	58
30	0.07	91.74	100.00	54

Grid	errors	% Black Correct	% White Correct	Tree Used
31	0.08	90.00	94.24	63
32	0.10	81.36	88.09	99
33	0.14	91.41	84.20	148
34	0.15	78.29	76.34	64
35	0.19	82.26	80.27	66
36	0.15	81.32	78.57	166
37	0.15	84.15	75.53	81
38	0.16	83.93	86.21	83
39	0.12	88.59	86.49	65
40	0.09	93.43	100.00	58
41	0.09	75.61	91.70	77
42	0.11	90.12	87.58	50
43	0.13	88.29	87.94	154
44	0.16	86.84	82.33	121
45	0.19	77.37	85.19	266
46	0.19	81.54	81.39	236
47	0.19	81.75	80.00	177
48	0.16	86.90	72.32	132
49	0.14	86.34	78.48	50
50	0.10	88.95	80.00	67
51	0.08	100.00	91.08	89
52	0.11	96.15	88.40	74
53	0.13	85.48	87.96	85
54	0.16	77.78	84.37	74
55	0.15	77.70	87.24	162
56	0.19	78.21	77.47	171
57	0.20	81.18	82.89	55
58	0.18	83.51	72.15	70
59	0.16	83.19	85.71	214
60	0.13	89.67	75.00	68
61	0.06	100.00	93.66	58
62	0.10	79.55	92.22	59
63	0.12	84.09	88.02	72
64	0.15	83.33	88.75	107
65	0.16	83.13	85.85	132
66	0.19	75.19	72.14	259
67	0.21	79.53	76.95	132
68	0.22	78.48	79.41	171
69	0.19	76.64	73.33	85
70	0.13	84.58	70.00	87

Grid	errors	% Black Correct	% White Correct	Tree Used
71	0.06	100.00	92.41	60
72	0.10	82.61	90.58	56
73	0.12	85.00	86.91	72
74	0.13	85.00	84.42	57
75	0.19	81.82	85.75	252
76	0.19	77.89	77.42	54
77	0.20	68.79	80.00	74
78	0.23	73.80	72.63	73
79	0.21	79.78	76.07	77
80	0.17	83.53	80.00	59
81	0.04	NA	96.42	70
82	0.08	81.82	92.25	93
83	0.11	86.96	89.65	65
84	0.14	89.29	87.17	153
85	0.16	80.33	85.57	207
86	0.18	82.29	81.34	230
87	0.20	76.62	73.33	112
88	0.20	74.12	79.56	68
89	0.21	77.55	79.01	213
90	0.20	79.31	77.53	152
91	0.03	100.00	95.34	79
92	0.07	NA	91.40	56
93	0.11	66.67	87.02	60
94	0.13	72.73	87.53	281
95	0.15	81.25	85.83	72
96	0.18	76.67	85.52	196
97	0.18	80.70	82.33	64
98	0.19	90.67	79.36	174
99	0.20	72.32	81.28	64
100	0.20	80.69	70.35	223

Table B.1 Random Forest model results without draws

B.2 SVM model results without draws

Grid	errors	% Black Correct	% White Correct	Cost Used
1	0.05	90.56	92.43	10.0
2	0.06	89.16	94.96	10.0
3	0.06	90.73	88.05	10.0
4	0.08	89.09	90.61	1.0
5	0.07	92.24	87.06	1.0
6	0.08	94.83	85.00	1.0
7	0.06	90.19	84.15	1.0
8	0.06	92.92	95.18	1.0
9	0.07	93.30	97.56	1.0
10	0.05	94.82	94.12	1.0
11	0.08	92.45	93.88	1.0
12	0.11	83.62	92.23	1.0
13	0.10	85.29	86.86	10.0
14	0.13	86.06	87.36	1.0
15	0.12	89.40	83.82	1.0
16	0.09	86.41	83.19	1.0
17	0.10	88.24	82.02	1.0
18	0.08	90.55	71.70	10.0
19	0.08	89.58	86.21	1.0
20	0.04	95.71	71.43	1.0
21	0.06	84.38	93.27	1.0
22	0.11	85.44	87.53	1.0
23	0.12	78.21	86.34	1.0
24	0.13	81.42	86.86	1.0
25	0.14	84.43	84.90	1.0
26	0.15	85.27	84.72	1.0
27	0.11	88.60	78.77	10.0
28	0.10	85.88	82.35	1.0
29	0.08	90.02	80.85	10.0
30	0.05	92.80	75.00	1.0

Grid	errors	% Black Correct	% White Correct	Cost Used
31	0.05	92.31	86.96	1.0
32	0.08	83.93	88.13	10.0
33	0.13	83.76	85.19	1.0
34	0.15	80.28	83.40	1.0
35	0.17	79.38	80.45	1.0
36	0.14	82.15	84.62	10.0
37	0.14	83.03	78.79	1.0
38	0.16	83.87	78.87	1.0
39	0.09	89.23	87.23	10.0
40	0.07	93.98	100.00	1.0
41	0.06	81.40	93.46	10.0
42	0.10	80.81	89.63	1.0
43	0.11	82.03	85.93	1.0
44	0.13	69.17	81.73	10.0
45	0.19	78.64	85.38	0.1
46	0.19	84.00	79.92	1.0
47	0.19	84.73	74.83	1.0
48	0.16	84.78	81.82	1.0
49	0.14	83.74	89.23	1.0
50	0.09	89.54	84.62	1.0
51	0.04	78.26	94.00	10.0
52	0.08	73.81	90.66	10.0
53	0.11	92.59	86.78	1.0
54	0.15	78.12	84.55	1.0
55	0.13	79.31	84.26	10.0
56	0.20	78.86	77.91	0.1
57	0.19	78.15	77.18	1.0
58	0.16	83.62	79.75	10.0
59	0.16	83.92	75.00	1.0
60	0.08	88.00	70.59	10.0
61	0.04	89.47	93.76	1.0
62	0.09	79.17	90.79	1.0
63	0.11	84.44	88.14	1.0
64	0.13	86.73	85.15	1.0
65	0.15	85.11	85.16	1.0
66	0.20	73.23	79.84	1.0
67	0.22	77.29	72.24	0.1
68	0.20	78.11	71.69	1.0
69	0.21	76.69	85.25	1.0
70	0.11	89.20	66.67	10.0

Grid	errors	% Black Correct	% White Correct	Cost Used
71	0.04	93.33	92.41	1.0
72	0.09	93.33	89.73	1.0
73	0.10	71.88	88.86	1.0
74	0.12	73.17	86.68	1.0
75	0.14	69.35	80.00	10.0
76	0.19	75.00	80.12	1.0
77	0.19	72.97	86.19	1.0
78	0.19	80.90	75.00	10.0
79	0.20	77.94	78.85	1.0
80	0.14	83.53	78.38	10.0
81	0.03	60.00	95.91	1.0
82	0.07	85.00	93.05	1.0
83	0.09	77.27	90.25	1.0
84	0.11	74.42	86.39	1.0
85	0.14	80.30	83.37	1.0
86	0.17	76.19	80.38	1.0
87	0.19	75.95	80.72	1.0
88	0.18	73.89	81.02	10.0
89	0.20	71.62	76.97	1.0
90	0.19	79.57	67.78	1.0
91	0.02	75.00	97.22	1.0
92	0.06	25.00	90.84	10.0
93	0.09	83.33	93.55	10.0
94	0.09	50.00	86.48	10.0
95	0.14	78.57	86.05	1.0
96	0.12	68.75	83.71	10.0
97	0.10	65.38	82.07	100.0
98	0.14	71.83	82.07	10.0
99	0.14	75.27	73.58	10.0
100	0.22	76.35	73.62	1.0

Table B.2 SVM model results without draws

B.3 Random Forest model results with draws

Grid	errors	% Black Correct	% Draw Correct	% White Correct	Tree Used
1	0.09	87.97	96.55	84.36	85
2	0.10	89.52	92.31	87.46	63
3	0.11	86.48	90.09	84.51	67
4	0.12	85.76	87.04	80.83	75
5	0.13	84.57	90.00	88.06	269
6	0.14	88.92	82.13	88.00	127
7	0.14	86.43	84.15	82.89	209
8	0.13	85.74	87.98	89.66	53
9	0.14	85.53	88.27	96.00	254
10	0.12	88.17	76.36	100.00	98
11	0.14	81.97	91.71	84.48	121
12	0.17	71.43	86.90	81.52	97
13	0.21	78.37	80.75	81.68	234
14	0.20	76.35	79.39	75.00	201
15	0.21	77.30	73.31	81.32	58
16	0.23	78.79	76.59	77.14	230
17	0.20	78.42	74.66	77.97	119
18	0.22	76.74	71.88	69.23	86
19	0.18	82.49	79.63	80.00	96
20	0.12	84.75	86.05	NA	51
21	0.16	80.00	92.17	78.31	128
22	0.21	71.21	81.13	80.59	270
23	0.22	67.07	78.67	71.50	135
24	0.25	75.00	72.70	75.62	151
25	0.25	70.63	73.95	73.21	142
26	0.29	75.07	70.70	70.77	147
27	0.27	76.67	73.23	72.64	125
28	0.25	75.42	72.67	75.47	115
29	0.23	75.04	79.22	68.18	162
30	0.16	83.96	85.87	NA	94

Grid	errors	% Black Correct	% Draw Correct	% White Correct	Tree Used
31	0.16	85.71	92.67	74.94	262
32	0.23	80.77	71.08	78.70	89
33	0.28	71.90	67.61	73.04	200
34	0.27	69.75	75.58	68.83	161
35	0.30	69.34	70.19	70.74	208
36	0.30	68.45	63.90	65.31	170
37	0.30	73.06	69.09	60.24	187
38	0.31	64.50	66.49	57.58	130
39	0.29	69.09	63.83	66.67	90
40	0.20	78.00	80.39	NA	82
41	0.19	89.29	92.91	78.75	96
42	0.25	75.24	69.71	74.91	88
43	0.26	71.97	73.09	71.21	169
44	0.30	69.01	67.14	64.40	109
45	0.32	59.92	66.56	66.67	130
46	0.34	69.00	68.86	59.52	201
47	0.31	68.54	57.20	65.25	291
48	0.33	68.14	63.75	60.61	206
49	0.31	73.15	62.35	79.41	280
50	0.24	78.01	75.26	NA	242
51	0.17	100.00	97.62	82.00	68
52	0.24	62.16	78.43	74.61	81
53	0.29	70.91	63.28	68.73	134
54	0.33	68.32	65.22	66.98	214
55	0.31	67.42	57.74	68.67	170
56	0.33	63.50	66.67	57.54	209
57	0.36	67.39	59.62	63.25	178
58	0.36	65.99	61.11	68.09	293
59	0.31	65.13	64.06	52.46	109
60	0.30	67.49	61.11	NA	110
61	0.15	100.00	93.94	82.96	98
62	0.24	80.95	76.38	73.63	94
63	0.28	63.89	68.79	74.42	173
64	0.33	73.49	53.70	69.68	243
65	0.34	65.66	60.61	65.64	140
66	0.37	52.86	64.06	63.97	297
67	0.38	68.75	57.38	56.96	220
68	0.39	66.44	59.49	53.96	187
69	0.40	60.98	56.90	68.57	136
70	0.33	67.76	67.53	100.00	222

Grid	errors	% Black Correct	% Draw Correct	% White Correct	Tree Used
71	0.15	NA	97.06	81.26	102
72	0.25	100.00	59.77	72.62	61
73	0.29	60.87	76.32	71.77	99
74	0.33	57.14	59.30	67.80	236
75	0.39	58.00	53.92	62.57	116
76	0.40	63.00	58.75	61.20	276
77	0.38	60.31	56.25	55.56	180
78	0.41	59.26	63.54	57.36	278
79	0.37	59.13	54.85	68.89	116
80	0.38	62.85	60.94	50.00	201
81	0.14	NA	86.67	83.77	50
82	0.24	100.00	67.21	75.26	99
83	0.31	100.00	67.57	68.17	105
84	0.37	71.43	59.05	61.48	62
85	0.38	72.34	55.86	63.37	117
86	0.39	68.18	59.67	61.64	158
87	0.37	55.41	58.84	58.39	93
88	0.38	66.83	62.12	60.96	281
89	0.39	56.25	57.64	62.22	173
90	0.39	64.73	59.25	93.75	174
91	0.13	NA	100.00	84.97	52
92	0.24	NA	74.36	75.39	238
93	0.33	100.00	58.23	67.77	74
94	0.38	100.00	51.63	64.79	262
95	0.37	71.43	55.44	62.50	298
96	0.40	72.73	63.33	57.51	196
97	0.39	72.22	56.31	64.80	285
98	0.37	78.57	60.45	56.46	233
99	0.37	55.32	65.70	60.00	85
100	0.37	59.41	58.30	71.01	168

Table B.3 Random Forest model results with draws

B.4 SVM model results with draws

Grid	errors	% Black Correct	% Draw Correct	% White Correct	Cost Used
1	0.07	88.14	94.71	85.28	10.0
2	0.08	85.31	90.79	87.70	10.0
3	0.11	84.40	88.84	82.67	1.0
4	0.12	87.59	86.75	81.59	1.0
5	0.11	86.00	86.32	80.11	1.0
6	0.12	87.38	86.18	81.30	1.0
7	0.12	87.43	83.17	81.54	1.0
8	0.11	86.99	82.01	88.73	1.0
9	0.13	86.65	82.76	85.29	1.0
10	0.08	90.99	82.86	65.38	10.0
11	0.12	85.94	90.15	84.20	1.0
12	0.16	79.85	79.84	80.67	1.0
13	0.18	81.86	80.16	75.12	1.0
14	0.21	76.10	77.66	73.81	0.1
15	0.20	81.29	76.00	74.77	1.0
16	0.18	81.21	69.26	74.70	1.0
17	0.18	76.69	77.45	74.47	1.0
18	0.20	81.19	76.42	72.09	1.0
19	0.15	84.09	80.73	66.67	1.0
20	0.08	89.14	79.41	33.33	10.0
21	0.12	69.77	87.56	85.08	10.0
22	0.18	78.70	77.22	75.45	1.0
23	0.21	68.93	75.00	79.94	1.0
24	0.20	75.20	73.78	77.09	10.0
25	0.24	77.21	70.04	75.54	1.0
26	0.26	76.64	67.86	68.97	1.0
27	0.23	74.07	72.92	73.02	1.0
28	0.22	76.91	70.59	72.73	1.0
29	0.21	75.54	77.56	71.79	1.0
30	0.15	86.13	74.12	50.00	1.0

Grid	errors	% Black Correct	% Draw Correct	% White Correct	Cost Used
31	0.14	83.78	89.70	81.62	1.0
32	0.20	79.66	78.32	76.53	1.0
33	0.23	73.53	68.60	72.43	1.0
34	0.25	65.19	67.40	72.95	1.0
35	0.28	67.76	66.07	74.37	1.0
36	0.28	74.45	66.97	66.46	1.0
37	0.27	70.33	69.73	63.83	1.0
38	0.28	68.49	71.02	58.82	1.0
39	0.26	73.86	75.22	71.43	1.0
40	0.18	80.94	84.00	61.54	1.0
41	0.16	75.00	91.03	78.47	1.0
42	0.23	81.11	81.48	72.39	1.0
43	0.24	76.67	67.46	73.33	1.0
44	0.27	69.75	56.60	71.13	1.0
45	0.32	65.99	67.63	63.13	0.1
46	0.31	70.25	63.50	58.70	1.0
47	0.29	67.03	65.89	65.87	1.0
48	0.31	70.17	64.59	65.28	1.0
49	0.27	67.54	70.97	50.82	1.0
50	0.21	76.17	78.38	53.33	1.0
51	0.13	87.50	95.29	81.57	1.0
52	0.23	75.61	70.87	75.40	1.0
53	0.25	73.33	65.07	68.57	1.0
54	0.29	66.04	60.42	66.99	1.0
55	0.26	60.96	66.04	66.30	1.0
56	0.31	66.67	62.10	66.25	1.0
57	0.33	70.04	58.02	58.17	1.0
58	0.32	62.29	64.47	56.86	1.0
59	0.28	69.13	52.68	62.07	1.0
60	0.29	71.90	69.91	44.44	1.0
61	0.11	62.50	84.00	86.53	10.0
62	0.17	69.81	66.38	75.17	10.0
63	0.25	71.74	65.44	71.99	1.0
64	0.30	65.22	69.23	68.74	1.0
65	0.30	60.40	66.84	63.73	1.0
66	0.32	61.24	59.93	60.73	1.0
67	0.35	66.38	57.40	57.14	1.0
68	0.37	61.06	52.27	59.68	1.0
69	0.36	60.26	57.25	42.00	1.0
70	0.25	69.33	64.10	36.73	10.0

Grid	errors	% Black Correct	% Draw Correct	% White Correct	Cost Used
71	0.13	92.31	79.07	83.77	1.0
72	0.23	69.05	82.35	69.74	1.0
73	0.27	65.38	69.86	71.65	1.0
74	0.29	65.85	61.44	64.93	1.0
75	0.35	61.33	56.44	61.89	1.0
76	0.36	61.70	58.49	62.21	1.0
77	0.36	62.61	59.86	58.43	1.0
78	0.31	58.72	56.93	62.50	10.0
79	0.35	57.65	60.96	68.06	1.0
80	0.34	62.98	55.47	42.86	1.0
81	0.13	NA	96.77	85.92	1.0
82	0.22	47.62	89.13	76.26	1.0
83	0.28	60.00	71.21	67.93	1.0
84	0.32	74.29	52.86	66.14	1.0
85	0.33	60.87	59.48	63.67	1.0
86	0.37	65.22	57.23	56.01	1.0
87	0.28	72.58	53.41	63.16	10.0
88	0.32	69.75	59.23	52.06	10.0
89	0.37	63.96	57.54	60.44	1.0
90	0.23	68.16	60.64	53.25	100.0
91	0.09	88.89	72.41	91.00	10.0
92	0.23	50.00	76.92	75.66	1.0
93	0.28	52.38	50.82	68.71	1.0
94	0.26	59.52	55.61	64.03	10.0
95	0.27	54.76	58.97	62.36	10.0
96	0.30	70.00	53.24	62.71	10.0
97	0.19	51.56	59.09	63.64	100.0
98	0.27	54.72	60.79	61.24	10.0
99	0.36	66.67	62.89	58.82	1.0
100	0.30	48.21	59.96	56.58	10.0

Table B.4 SVM model results with draws

BIBLIOGRAPHY

Alludo, "Winzip". WinZip. Accessed July 10, 2023. https://www.winzip.com/en/pages/download/winzip-b3/?x-target=ppc&promo=ppc&utm_source=google&utm_medium=cpc&utm_campaign=wz-dd-all-adwordsppc&utm_content=160010506971&utm_term=winzip&utm_id=20624671898&gclid=EAIaIQobChMIqP3R8L_4gQMVuSyzAB0PnAVeEAAYASAAEgLSdvD_BwE.

arwagner et al, "Scid". SourceForge. Accessed July 10, 2023. https://sourceforge.net/projects/scid/.

Breiman, Leo, et al., "randomForest" package in R. R. Accessed October 4, 2023. https://cran.r-project.org/web/packages/randomForest/index.html.

codekiddy2, "15 Million Games Chess Database." SourceForge. Accessed July 10, 2023. https://sourceforge.net/projects/codekiddy-chess/.

Cohen, Adam, "fuzzywuzzy" package in Python. fuzzywuzzy 0.18.0. Accessed July 11, 2023. https://pypi.org/project/fuzzywuzzy/#description.

Karatzoglou, Alexandros, et al., "kernlab" package in R. R. Accessed October 4, 2023. https://cran.r-project.org/web/packages/kernlab/index.html.

Liu, Yanli, et al. "New Machine Learning Algorithm: Random Forest". International Conference on Information Computing and Applications, no. 3 (2012): 246-252.

Microsoft Corporation, "Visual Studio Code". Visual Studio Code. Accessed July 10, 2023. https://code.visualstudio.com/.

Ooms, Jeroen, "jsonlite" package in R. R. Accessed July 24, 2023. https://cran.r-project.org/web/packages/jsonlite/index.html.

OpenAI, "ChatGPT". ChatGPT. Accessed July 12, 2023 through to November 13, 2023. https://chat.openai.com/.

Posit and Allaire, Joseph J. , "RStudio Desktop". Accessed July 24, 2023. https://posit.co/download/rstudio-desktop/.

Pulido, Héctor, "Predicting the Outcome of a Chess Game by Statistical and Machine Learning Techniques". UPCommons. Accessed July 10, 2023 through to November 13, 2023. https://upcommons.upc.edu/bitstream/handle/2117/106389/119749.pdf?sequence=1&isAllowed=y

Python Software Foundation, "Python". Python. Accessed July 10, 2023. https://www.python.org/.

R Core Team, "R". R. Accessed July 24, 2023. https://www.r-project.org/.

Robin, Xavier, et al., "pROC" package in R. R. Accessed November 5, 2023. https://cran.r-project.org/web/packages/pROC/index.html.

Unknown author(s), "chess.pgn" package in Python. Accessed July 11, 2023. Found in Python's Visual Studio Code Integrated Development Editor.

Unknown author(s), "json" package in Python. Accessed July 11, 2023. Found in Python's Visual Studio Code Integrated Development Editor.

Unknown author(s), "Lesson 10: Support Vector Machines". PennState Eberly College of Science. Accessed November 12, 2023. https://online.stat.psu.edu/stat857/node/211/.

Unknown author(s), "multiprocessing" package in Python. Python. Accessed August 17, 2023. https://docs.python.org/3/library/multiprocessing.html.

Unknown author(s), "re" package in Python. Python. Accessed August 17, 2023. https://docs.python.org/3/library/multiprocessing.html.

Unknown author(s), "10.5 - Multiclass SVM". PennState Eberly College of Science. Accessed November 12, 2023. https://online.stat.psu.edu/stat857/node/244/.

Wickham, Hadley, et al., "ggplot2" package in R. R. Accessed October 3, 2023. https://cran.r-project.org/web/packages/ggplot2/index.html.

Xie, Yihui, et al., "knitr" package in R. R. Accessed November 5, 2023. https://cran.r-project.org/web/packages/knitr/index.html.