



Minnesota State University, Mankato

## Cornerstone: A Collection of Scholarly and Creative Works for Minnesota State University, Mankato

---

All Graduate Theses, Dissertations, and Other  
Capstone Projects

Graduate Theses, Dissertations, and Other  
Capstone Projects

---

2021

### Classification of Chess Games: An Exploration of Classifiers for Anomaly Detection in Chess

Masudul Hoque

*Minnesota State University, Mankato*

Follow this and additional works at: <https://cornerstone.lib.mnsu.edu/etds>



Part of the [Applied Statistics Commons](#), and the [Artificial Intelligence and Robotics Commons](#)

---

#### Recommended Citation

Hoque, M. (2021). Classification of chess games: An exploration of classifiers for anomaly detection in chess [Master's thesis, Minnesota State University, Mankato]. Cornerstone: A Collection of Scholarly and Creative Works for Minnesota State University, Mankato. <https://cornerstone.lib.mnsu.edu/etds/1119>

This Thesis is brought to you for free and open access by the Graduate Theses, Dissertations, and Other Capstone Projects at Cornerstone: A Collection of Scholarly and Creative Works for Minnesota State University, Mankato. It has been accepted for inclusion in All Graduate Theses, Dissertations, and Other Capstone Projects by an authorized administrator of Cornerstone: A Collection of Scholarly and Creative Works for Minnesota State University, Mankato.

CLASSIFICATION OF CHESS GAMES  
An exploration of classifiers for anomaly detection in chess

By

Masudul Hoque

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

Applied Statistics, MS

In

Master's in Applied Statistics

Minnesota State University, Mankato

Mankato, Minnesota

May 2021

April 2<sup>nd</sup> 2021

## CLASSIFICATION OF CHESS GAMES

An exploration of classifiers for anomaly detection in chess

Masudul Hoque

This thesis has been examined and approved by the following members of the student's committee.

---

Advisor

---

Committee Member

---

Committee Member

# Table of Contents

<b>Chapter 1 Introduction .....</b>	<b>1</b>
1.1 Introduction.....	1
1.2 Literature Review .....	2
1.3 Research Problem Statement.....	6
<b>Chapter 2 Data.....</b>	<b>8</b>
2.1 Data Source.....	8
2.2 Data Description.....	8
2.3 Data Preprocessing .....	11
2.4 Data Manipulations .....	13
<b>Chapter 3 Methodology .....</b>	<b>15</b>
3.1 Classification.....	15
3.1.1 Discriminant Analysis .....	15
3.1.1.a Linear Discriminant Analysis .....	17
3.1.1.b Quadratic Discriminant Analysis.....	21
3.1.2 K Nearest Neighbour .....	24
3.1.3 Multinomial Logistic Regression .....	27
3.2 Model Validation .....	32
3.2.1 Confusion Matrix.....	34
3.2.2 K Fold Cross-Validation.....	39
3.2.3 Leave One Out Cross-Validation.....	41
3.3 Technology Use.....	42

<b>Chapter 4 Results</b>	<b>44</b>
4.1 Phase I	44
4.1.1 Linear Discriminant Analysis	51
4.1.2 Quadratic Discriminant Analysis	54
4.1.3 Multinomial Logistic Regression	57
4.1.4 K nearest Neighbour	62
4.1.5 Phase I Summaries	70
4.2 Phase II	77
4.2.1 Linear Discriminant Analysis	81
4.2.2 Quadratic Discriminant Analysis	85
4.2.3 Multinomial Logistic Regression	89
4.2.4 K nearest Neighbour	94
4.2.5 Phase II Summaries	102
4.3 Phase III	109
<b>Chapter 5 Conclusions</b>	<b>118</b>
5.1 Conclusion	118
5.2 Future Work	121
<b>References</b>	<b>123</b>
<b>Appendix</b>	<b>123</b>
Python Codes	127
R codes	130

# CLASSIFICATION OF CHESS GAMES

## An exploration of classifiers for anomaly detection in chess

Masudul Hoque

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE OF  
MASTER OF SCIENCE IN APPLIED STATISTICS

MINNESOTA STATE UNIVERSITY, MANKATO

MANKATO, MINNESOTA

MAY 2021

### ABSTRACT

Chess is a strategy board game with its inception dating back to the 15th century. The Covid-19 pandemic has led to a chess boom online with 95,853,038 chess games being played during January, 2021 on lichess.com. Along with the chess boom, instances of cheating have also become more rampant. Classifications have been used for anomaly detection in different fields and thus it is a natural idea to develop classifiers to detect cheating in chess. However, there are no specific examples of this, and it is difficult to obtain data where cheating has occurred. So, in this paper, we develop 4 machine learning classifiers, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Multinomial Logistic Regression, and K-Nearest Neighbour classifiers to predict chess game results and explore predictors that produce the best accuracy performance. We use Confusion Matrix, K Fold Cross-Validation, and Leave-One-Out Cross-Validation methods to find the accuracy metrics. There are three phases of analysis. In phase I, we train classifiers using 1.94 million over the board game as training data and 20 thousand online games as testing data and obtain accuracy metrics. In phase II, we select a smaller pool of 212 games, select additional predictor variables from chess engine evaluation of the moves played in those games and check whether the inclusion of the variables improve performance. Finally, in phase III, we investigate for patterns in misclassified cases to define anomalies. From phase I, the models are not performing at a utilizable level of accuracy (44-63%). For all classifiers, it is no better than deciding the class with a coin toss. K-Nearest Neighbour with  $K = 7$  was the best model. In phase II, adding the new predictors improved the performance of all the classifiers significantly across all validation methods. In fact, using only significant variables as predictors produced highly accurate classifiers. Finally, from phase III, we could not find any patterns or significant differences between the predictors for both correct classifications and misclassifications. In conclusion, machine learning classification is only one useful tool to spot instances that indicates anomalies. However, we cannot simply judge anomalous games using only this method.

# Acknowledgements

I would first like to thank my thesis advisor Dr Iresha Premarathna of the Department of Mathematics & Statistics at Minnesota State University-Mankato. Dr Premarathna was always available to provide invaluable insight, guidance, and answer any question about my research or writing. She consistently allowed this paper to be my work but steered me in the right direction whenever I felt lost.

I would also like to acknowledge Dr. Mezbahur Rahman and Dr. Deepak Sanjel of the Department of Mathematics & Statistics at Minnesota State University-Mankato as esteemed members of the thesis committee, who were wonderful teachers in many of my courses. I am gratefully indebted to their teachings and valuable comments on this thesis.

I must express my very profound gratitude to my mother, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. I would also like to thank my friends for lending me sympathetic ears and acting as a support group to keep me sane throughout my life as a graduate student. Thank you.

Finally,

Author

Masudul Hoque

# Chapter 1: Introduction

## 1.1 Introduction

Chess is one of the most popular boardgames and is steeped in history. It is an abstract strategy game without hidden information emerging around the 15th century in Europe, evolving from older games from India and Persia (Chaturanga). Starting from the 19th century, the game became more competitive and professional. And with the rapid improvement of chess-playing computer engines, the new era of chess began in the 1990s. It is one of the most famous competitive game with hundreds of tournaments played around the world.

The Covid-19 pandemic, starting at the end of 2019, has created an unprecedented threat to global health and requires people to stay inside to prevent the spread of the disease.

Lockdowns have led to a lot of people picking up chess again, leading to a boom online with thousands of games being played every single day. The chess-playing website Lichess.org reported 95,853,038 games played in January 2021. [1]

Alongside the chess boom, instances of cheating in chess have also increased. Chess.com, a popular online chess playing website, reports more than 500 account closures every day for chess engine use. [2] Chess engines are computer programs that analyze chess positions and generate a list of the strongest moves and is a common educational tool in chess. But if used during a game, it can offer an unfair advantage to a player.



Even in the highest professional chess leagues and tournaments happening online, suspicions and paranoia of cheating have become frequent. One of the most recent high profile cases occurred when Armenian Grandmaster Tigran Petrosian was accused of using chess engine assistance in the final game in the chess.com Pro Chess League. The chess.com fair play team assessed and concluded the accusations to be correct which led to the Armenian Eagles being stripped of their victory, and Tigran Petrosian being banned for life on the chess.com server. [3]

Cheating detection in online chess games is a challenging and multifaceted problem. In this paper, we hope to develop classifiers that can predict chess game results with accuracy, based on certain variables. Misclassifications by an accurate classifier can provide us with observations to study further to find patterns that can be considered anomalous.

## 1.2 literature review

Anomaly detection is the identification of rare observations which raise suspicions by differing significantly from the majority of the data.

[4] Typically the anomalous items will translate to some kind of problem such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are also referred to as outliers, novelties, noise, deviations and exceptions. [5]

Machine learning techniques have been used as a tool for anomaly detection, especially for network intrusion and fraud detections. [6]

Especially Supervised machine learning that can categorize observations into predefined classes, also known as classifiers have been used for Handwriting recognition, Object recognition in computer vision, Pattern recognition, Speech recognition, Spam detection, and most importantly anomaly detection [7].

This is why it is a natural idea to develop classifiers to use as anomaly detection for online chess games. Chess games have three labelled categorical outcomes: the White player wins, the Black player wins, or the game is a Draw. We can develop classifiers that can predict the results of a chess game [8] based on certain features that are available to us. We intend to use the information available in large chess game databases to train machine learning classifiers. Then wrong classifications can be further investigated to find patterns that can be considered anomalies.

Information such as the skill level of the contending players can be valuable to predict game results, as stronger skilled players are likelier to win games. The ELO rating system is widely used in chess to calculate an estimate of the strength of a player and is adopted by FIDE. The ELO rating system is a method for calculating the relative skill level of players in zero-sum games (such as chess, table tennis, board games and esports). The rating is not a absolute strength of a player, rather it is a system that calculates the probable outcome of the game based on the difference in ratings. Two players with equal ratings are expected to win 50% of the games against each other. While a player rated 100 points greater than their opponents is expected to win 64% of the games. With a 200 points difference, the expected score

for the stronger player is to win 76% of the games and so on. The difference in ELO points also dictates how many rating points a player will earn for defeating the other player. [9]

The rating ranges from 2500+ for most of the Grandmasters (GM) and World Champions, 2400-2500 for International Masters (IM), 2200-2400 for FIDE Masters (FM) and Candidate Masters (CM), 1200-2000 for Class D/C/B/A players while rating below 1200 is for novices. [8] If the players have registered with FIDE, and have played some rated games in a FIDE chess event, they may obtain their ELO rating. Many chess websites employ similar rating systems to rate their players, which is obtainable in chess databases. And ELO ratings have been used to predict chess game results before. [10]

The total number of turns played during a chess game may be a useful predictor to help predict results. In a two-player, sequential game such as chess, each player's turns are often referred to as half moves. In chess, a full move is when both players complete their turns. But the move counting begins when white players move. Thus, in a 30-move chess game, if the game ends on the white players turn there were 59 half moves, while if the game ends on the black players turn then there were 60 half moves. Longer games may lead to mistakes being made by either player due to loss of focus, thus we can consider using it as a predictor variable.

Finally, chess databases also include the chess moves played in a game in algebraic notation. It is readable to chess engines that can evaluate the strength of moves.

These evaluations can be a useful feature for prediction. But analyzing the games is a time-consuming process and not easy to automate. Additionally, it is limiting to only use chess engine analysis results as a basis for cheating detection. [11]

To understand some of the terms we shall discuss next we need to understand some Chess terminologies.

*Centipawn* is the unit measure of advantage in chess. A centipawn is equal to 1/100 of the value of the pawn (thus, centi-pawn). In chess, the pieces have material value, with pawns weighing 1 point, bishops and knights weighing 3 points, rooks are 5 points and the queen is 9 points. The king has indiscernible value as it cannot be captured, but checkmating the king wins you the game. These values play no formal role in the game but are useful to players, and essentials in computer chess, to evaluate positions.

Inaccuracies indicate suboptimal moves being played, which often leads to 50 to 100 centipawn loss. Mistakes are clear bad moves that may lead to loss of material (i.e., losing a piece). Blunders are the worst possible moves that can lead to a loss of the game. Each move by both players can be evaluated by chess engines and then the advantages are calculated and a centipawn loss or gain is outputted. An average centipawn loss is the average centipawn lost over the moves. Lower average centipawn loss is usually indicating fewer inaccuracies, mistakes and blunders. Although, one blunder can still cost a player with less average centipawn to lose.

Chess-playing website lichess.com provides the opportunity to evaluate a sequence of chess moves played in a game using Stockfish 13+, a free and popular chess engine. The engine analyses each move and provides the centipawn loss or gain for each move. In tandem, it provides the number of inaccuracies, mistakes, blunders and average centipawn loss of both the white player and the black player. On a smaller scale, we can check whether these additional variables can help improve the accuracy of predictions.

## 1.3 Research problem statement

As discussed before, since classifications have been used for anomaly detection in other fields, it is a natural idea to develop classifiers to detect anomalous games in chess. Since instances of cheating are more prevalent in online chess games, it is reasonable to use over-the-board games to train the machine learning classifiers and test their accuracy on online games in the hopes to detect anomalies.

However, there are no such prior examples of classifiers used for anomaly detection in chess. Additionally, it is difficult to obtain data where cheating has occurred, as most websites either don't record or cannot share the data publically due to legal or other reasons. Furthermore, simulating such data is also arduous as there are often no clear established patterns due to the lack of public data. Therefore, it is a challenge to verify that such a process will satisfactorily work.

Thus, it is more practical in this paper to create accurate classifiers and figure out predictors that best help in that regard. Furthermore, we can perhaps study the

misclassifications and obtain a picture of what anomalies in a chess game may look like.

The whole analysis is divided into **three** phases.

**Phase I:** We preprocess the large chess game database into a dataset for analysis.

We split the dataset into two parts, over-the-board games as the training dataset and online games as the testing dataset. We then train several machine learning classifier models, modelled with 3 predictors, and obtain their accuracy measurements to compare their performances.

**Phase II:** We randomly select a smaller subset of the over-the-board chess game dataset and analyze the sequence of chess moves played on a chess engine to obtain the frequency of inaccuracies, mistakes, blunders and the average centipawn loss for both players and use them as additional predictors in the same models and check for improvements in accuracy performances.

**Phase III:** We look at the misclassifications by the most accurate classifier for the larger online test dataset in phase I and the smaller dataset in phase II and investigate for patterns that may establish what can be considered anomalies.

# Chapter 2: Data

## 2.1 Data source

The over the board chess game data was originally obtained from a chess game database called SCID (Shane's Chess Information Database). It contains around **3.5 million over-the-board chess games**. Over-the-board games can be considered offline games. It is available as a combined text file on kaggle.com. More details about the documentation for the chess dataset can be obtained on this link. [12] We use this set of data as the training set in phase I.

The online chess games data is obtained from lichess.com, a popular online chess playing website. It contains **20 thousand chess games** with similar information as the over-the-board database aforementioned. This data was collected using the Lichess API into a CSV file and it is available in Kaggle at this link. [13]

We use this set of data as the testing set in phase I.

## 2.2 Data description

The text file database contained several columns of observations detailing game attributes. Those ending in “\_c” are indicator variables explaining whether the corresponding attribute is corrupted or missing. If true, then the observation is corrupted. If false, then it is not corrupted. The descriptions of the columns are as follows,

1. t: Position of the game in the original PGN file.
2. date: Date at which the game was played (the format is year.month.day).
3. result: Game result specified inside brackets in the PGN file. The value can be 1, 0 or -1 corresponding to white win, draw or loose, respectively.
4. welo: ELO of white player (an integer number).
5. belo: ELO of black player (an integer number).
6. len: Number of moves in the game (for some games it may be zero!)
7. date\_c: Whether date (yyyy.mm.dd) is corrupted or missing.
8. resu\_c: Whether result is corrupted or missing.
9. welo\_c: Whether the ELO is corrupted or missing.
10. belo\_c: black ELO is corrupted or missing.
11. edate\_c: Whether the event date is corrupted or missing. The event where the game was held (if there is one).
12. setup: This attribute may be setup\_true or setup\_false. If it is true, then the game initial position is specified. This is used when playing Fischer Random Chess for example.
13. fen: This attribute may be fen\_true and fen\_false. It is related to the setup and explains the initial setup of the pieces.
14. resu2\_c: In the original file, the result is provided in two places. At the end of each sequence of moves and in the attributes part. This attribute indicates if the result is (is not) properly provided after the sequence of moves.



15. `oyrange`: This attribute may be `oyrange_true` or `oyrange_false`. This attribute is false only for games with dates in the range of years [1998,2007]. The `oyrange` means out of year range.

16. `bad_len`: Whether the game length is corrupted or missing.

17. `###`: After this, we can find the sequence of moves. Each move has a number and a letter W (white) or B (black) indicating the th-move of the white or black player, respectively. For example, W1.d4 B1.d5 W2.c4 B2.e6 W3.Nc3 B3.Nf6...

In the online games, data from lichess contains the following attributes,

1. `id`: Game ID number of the chess game.
2. `rated`: Whether the game is rated or not.
3. `created_at`: Game started at time.
4. `last_move_at`: Game ended at time.
5. `turns`: Number of half moves. This is equivalent to the `len` attribute from over-the-board database.
6. `victory_status`: Whether the game ended from resignation, checkmate, running out of time etc.
7. `winner`: Whether the game resulted in white, black or draw. This is equivalent to the `result` attribute from earlier.
8. `increment`: Amount of time allotted with increment per turn for a timed game.
9. `white_id`: White player's User ID on the website.

10. `white_rating`: ELO rating of the white player. Equivalent to `welo` attribute from earlier.
11. `black_id`: Black player's User ID on the website.
12. `black_rating`: ELO rating of the black player. Equivalent to `belo` attribute from earlier.
13. `moves`: Sequence of moves played in the game in Standard Chess Notation.  
Similar to the information after `###` in the Over-the-board database.
14. `opening_eco`: Standardized Code for any given opening played on the game.
15. `opening_name`: Name of the opening played.
16. `opening_ply`: Number of moves in the opening phase.

In the next section, we describe the procedures used to organize and extract relevant information from these databases.

## 2.3 Data Preprocessing

The over-the-board chess game database is parsed using Python, where the text information is separated into columns. The separated data is then saved as a CSV file to be processed in R. In R, we filter the data to remove all the corrupted and missing observations, using the indicator attributes ending with `"_c"` (if true then they are removed). Since these variables hold no other information about the games except if the information in them is corrupted or not, we do not consider them for the final dataset.

Additionally, we only consider standard chess games and ignore all other chess variants by filtering using the setup attribute. After all the preprocessing and filtration, the remainder over-the-board dataset contains about **1.9 million chess games** with the following features,

1. t = Position of the game in the database
2. result = Outcome of the game. The variable has three levels, White wins, Black wins, and Draw.
3. welo = ELO rating of white player. An integer number ranging from 1 to 2851.
4. belo = ELO rating of black player. An integer number ranging from 1 to 2851.
5. length = Number of turns or half moves in the game. Ranges from 1 to 600.

As for the online chess games database, we separate winner, white\_rating, black\_rating and turns variables which are equivalent to result, welo, belo, and length variables respectively. There are no missing or corrupted values in this dataset and all the games standard chess games. This dataset contains **20 thousand chess games**.

Finally, we also randomly select a small pool of games from the over-the-board database for chess engine analysis of their sequence of moves. We evaluate the games over at lichess.com, which uses Stockfish 13+, a free chess engine for move evaluation and returns certain summaries. Lichess.com allows only a certain number of evaluations per day to save server resources. Due to this restriction and time constraint, we ran the chess engine evaluations over several days and stopped

after obtaining information for 212 games (although with more time and resources such as additional accounts, it will be possible to obtain more in the future).

Along with the aforementioned variables, we gather the following new predictors from the chess engine evaluations for each of the games,

6. w.inaccuracies = Number of inaccurate moves played by the white player.

7. w.mistakes = Number of mistakes made by the white player.

8. w.blunders = Number of blunders made by the white player.

9. w.acl = The average centipawn loss of the white player.

10. b.inaccuracies = Number of inaccurate moves played by the black player.

11. b.mistakes = Number of mistakes made by the black player.

12. b.blunders = Number of blunders made by the black player.

13. b.acl = The average centipawn loss of the black player.

For this smaller dataset we then have 212 observations with 11 predictor variables and one response variable with 3 categories.

## 2.4 Data Manipulations

Before we begin training machine learning models, we need to perform a few data manipulations. Since the features selected from the dataset have differences between ranges, it is useful to standardize the observations. Additionally, since we

shall be using some distance-based machine learning models such as KNN, standardization will make certain that all variables contribute equally in similarity measurements.

For our purpose, we use the popular Z-score standardization. It is done by subtracting the mean,  $\mu$ , and dividing by the standard deviation,  $\sigma$  for each predictor.

$$Z = \frac{X_i - \mu}{\sigma} \text{ for all } i = 1, 2, 3, \dots, n$$

We perform this standardization for over-the-board database and online database separately, as they are from different sources.

Afterwards, we append these two datasets together to create the data frame for training machine learning classifiers in phase I. The 1942330 over-the-board offline games are selected as training dataset, while 20058 online games are selected as testing dataset. Thus the total dataset contains 1962388 total observations with 3 predictors and one response variable with 3 categories.

Furthermore, we perform standardization for the smaller dataset with 212 observations. We split that dataset into training and testing dataset as well. Seventy percent of the data (148 observations) is randomly selected for the training set, while the remainder (64 observations) is selected as a testing dataset.

# Chapter 3: Methodology

## 3.1 Classification

Classification is a technique concerned with separating distinct sets of objects or observations and with allocating new objects or observations to groups defined previously. The goal of classification is to sort observations into two or more labelled classes and obtain a rule which can be used to optimally assign new observations to the established labelled classes. [14]

Our problem of predicting game outcomes directly relates to classification. We shall use a few commonly used classification techniques and algorithms to classify chess games into the three possible results. The objective of this phase to compare the accuracies of several techniques and use the best one for the next phase of classifying online games.

There are several machine learning classification techniques we shall use for our purposes. We shall be discussing them in detail below.

### 3.1.1 Discriminant Analysis

Discrimination analysis is a technique which finds a set of prediction equation based on independent variables, which allows for distribution of observations into different categories, groups or classes of the same type. The goal is to describe, either graphically or algebraically, the differential features of objects or

observations from several known collections. We obtain discriminants, whose numerical values are such that the collections are separated as much as possible.

[14]

Discriminant Analysis can be considered as a parallel to multiple linear regression analysis. The procedures used to perform a discriminant analysis is similar to multiple linear analysis, such as, we can plot each independent variable versus the categorical dependent variable, go through a variable selection for modelling and determine which independent variables are significant and beneficial, and conduct a residual analysis to determine the accuracy of the discriminant equations.

The main difference between regression and discriminant analysis being that regression analysis deals with the continuous dependent variable, while discriminant analysis deals with categorical dependent variables.

The mathematics of discriminant analysis is related closely to one-way MANOVA with the roles of the variables reversing. The categorical variable in MANOVA becomes the dependent variable in discriminant analysis, and the dependent variables in MANOVA become the independent variable in the discriminant analysis. The discriminant analysis also shares assumptions with MANOVA. The assumptions are as follows,

1. Multivariate normality: Independent variables are normal for each category of the grouping variable. [15][16]

2. Homogeneity of variance/covariance (homoscedasticity): Variances among group variables are the same across levels of predictors. [15] It is suggested, however, that linear discriminant analysis be used when covariances are equal, and that quadratic discriminant analysis may be used when covariances are not equal. [14]

3. Multicollinearity: An increased correlation between predictor variables may lead to a decrease in predictive power. [16]

4. Independence: Observations are assumed to be randomly sampled, and a value on one variable for an observation is assumed to be independent of the value of other variables for all other observations. [15][16]

It has been suggested that discriminant analysis is relatively robust to slight violations of these assumptions, [17] and it has also been shown that discriminant analysis may still be reliable when using dichotomous variables (where multivariate normality is often violated). [18]

### 3.1.1.a Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant [14]. It is a discriminant classification technique that finds a linear combination of features that classifies objects or observations into labelled classes. The resulting combination will be used as a linear classifier.



Say, we want to estimate  $P(Y|X)$ , where  $Y$  is the categorical response variable and  $X$  is the predictor variables. Instead of directly estimating that, we may estimate,

$\hat{P}(X|Y)$ , which is the distribution of  $X$  given the response,

$\hat{P}(Y)$ , which estimates how likely each category of response are.

Thus, using the Bayes rule, we can obtain the estimate,

$$\begin{aligned}\hat{P}(Y = k | X = x) &= \frac{\hat{P}(X = x | Y = k) \hat{P}(Y = k)}{\hat{P}(X = x)} \\ &= \frac{\hat{P}(X = x | Y = k) \hat{P}(Y = k)}{\sum_{j=1}^k \hat{P}(X = x | Y = j) \hat{P}(Y = j)}\end{aligned}$$

Where  $k$  is the particular response class or category of the response variable  $Y$ .

Linear Discriminant Analysis is the special case of the above strategy where,

$$P(X|Y = k) \sim N(\mu_k, \Sigma)$$

That is, within each class the features follow a multivariate normal distribution with the mean depending on the class and common covariance  $\Sigma$ .

Here, the probabilities of the response being  $k$ ,  $P(Y = k)$  are estimated by the fraction of training samples of class  $k$ .

Suppose that,  $P(Y = k) = \pi_k$

And  $P(X = x|Y = k)$  follows multivariate normal distribution with pdf,

$$f_k(x) = \frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1} (x-\mu_k)}$$

Where  $\mu_k$  the mean of the  $x$  is's for category  $k$  and  $\Sigma$  is the common covariance matrix.

Then according to the Bayes rule, the probability of category  $k$  given  $x$  is,

$$P(Y = k|X = x) = \frac{f_k(x)\pi_k}{P(X = x)}$$

Now expanding  $f_k(x)$  we obtain,

$$P(Y = k|X = x) = \frac{\frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1} (x-\mu_k)} \pi_k}{P(X = x)}$$

Since some of the terms do not depend on the response  $k$ , we can consider them as a constant  $C$ , and write,

$$P(Y = k|X = x) = C \pi_k e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1} (x-\mu_k)}$$

Now taking natural logarithm on both sides,

$$\ln P(Y = k|X = x) = \ln C + \ln \pi_k - \frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)$$

Where the constant  $\ln C$  will be same for each category,  $k$ .

Now the goal is to maximize the terms without  $\ln C$ , that is to maximize  $\ln \pi_k -$

$$\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k) \text{ over } k.$$

We can summarize the expression,  $\ln \pi_k - \frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k)$

$$= \ln \pi_k - \frac{1}{2}[x^T \Sigma^{-1}x + \mu_k^T \Sigma^{-1}\mu_k] + x^T \Sigma^{-1}\mu_k$$

$$= C'' + \ln \pi_k - \frac{1}{2} \mu_k^T \Sigma^{-1}\mu_k + x^T \Sigma^{-1}\mu_k \quad [\text{setting } \frac{1}{2}x^T \Sigma^{-1}x \text{ as constant } C'']$$

Thus we can establish the objectives called discriminant functions as,

$$\delta_k(x) = \ln \pi_k - \frac{1}{2} \mu_k^T \Sigma^{-1}\mu_k + x^T \Sigma^{-1}\mu_k$$

Which at a value of  $x$ 's can predict the response with the highest  $\delta_k(x)$ .

To estimate the final discriminant function we perform the following steps,

1. We obtain the decision boundaries where the discriminant functions of two classes agree at a set of points of  $x$ , so that,

$$\delta_k(x) = \delta_l(x)$$

$$\text{Or, } \ln \pi_k - \frac{1}{2} \mu_k^T \Sigma^{-1}\mu_k + x^T \Sigma^{-1}\mu_k = \ln \pi_l - \frac{1}{2} \mu_l^T \Sigma^{-1}\mu_l + x^T \Sigma^{-1}\mu_l$$

2. We estimate  $\pi_k$  as the fraction of training samples of class  $k$ , such that,

$$\hat{\pi}_k = \frac{n(i | y_i = k)}{n}$$

3. We estimate the parameters  $\mu_k$  and  $\Sigma$  for  $f_k(x)$ .

$$\hat{\mu}_k = \frac{\sum_{y_i=k} x_i}{n(i | y_i = k)}$$

And for multiple predictors we can compute the vectors of deviations,

$(x_1 - \widehat{\mu}_{y_1}), (x_2 - \widehat{\mu}_{y_2}), \dots, (x_n - \widehat{\mu}_{y_n})$  and calculate the sample covariance matrix to estimate covariance matrix,  $\Sigma$ . The sample covariance can be estimated as,

$$\hat{\Sigma} = \frac{1}{n-1} \sum_{i=1}^n (x_{i\cdot} - \hat{\mu}_l)(x_{i\cdot} - \hat{\mu}_l)^T$$

Then we can define the final decision rule, for an input of values, the class is predicted to be with the largest,

$$\hat{\delta}_k(x) = \ln \hat{\pi}_k - \frac{1}{2} \hat{\mu}_k^T \hat{\Sigma}^{-1} \hat{\mu}_k + x^T \hat{\Sigma}^{-1} \hat{\mu}_k$$

With the decision boundaries  $\{x : \delta_k(x) = \delta_l(x)\}, l \leq k$ . [19]

### 3.1.1.b Quadratic Discriminant Analysis

Quadratic discriminant analysis (QDA) is another commonly used discriminant technique. It is most useful for classifying cases where the population is multivariate Normal with unequal covariance matrices. [14]

The process behind Quadratic Discriminant analysis is similar to Linear discriminant analysis, with a key difference, which is that the covariance matrices are not equal. i.e.,

$$\Sigma_i \neq \Sigma_j \text{ for } i \neq j$$

Thus according to the Bayes rule (from earlier), the probability of category  $k$  given  $x$  is,

$$P(Y = k|X = x) = \frac{f_k(x)\pi_k}{P(X = x)}$$

Now expanding  $f_k(x)$  we obtain,

$$P(Y = k|X = x) = \frac{\frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)} \pi_k}{P(X = x)}$$

But now,  $\Sigma_i \neq \Sigma_j$ , thus the equation becomes,

$$P(Y = k|X = x) = \frac{\frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)} \pi_k}{P(X = x)}$$

Since some of the terms do not depend on the response  $k$ , we can consider them as a constant  $C$ , and write,

$$P(Y = k|X = x) = C \frac{1}{|\Sigma_k|^{\frac{1}{2}}} \pi_k e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$$

Taking logarithm on both sides,

$$\ln P(Y = k|X = x) = \ln C + \ln \pi_k - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2} \ln |\Sigma_k|$$

Where the constant  $\ln C$  will be same for each category,  $k$ .

Now the goal is to maximize the terms without  $\ln C$ , that is to maximize  $\ln \pi_k -$

$$\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2} \ln |\Sigma_k| \text{ over } k.$$

Now, summarizing the expression,

$$\begin{aligned} \ln \pi_k - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2} \ln |\Sigma_k| \\ = \ln \pi_k - \frac{1}{2} [x^T \Sigma_k^{-1} x + \mu_k^T \Sigma_k^{-1} \mu_k] + x^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \ln |\Sigma_k| \\ = \ln \pi_k - \frac{1}{2} \mu_k^T \Sigma_k^{-1} \mu_k + x^T \Sigma_k^{-1} \mu_k - \frac{1}{2} x^T \Sigma_k^{-1} x - \frac{1}{2} \ln |\Sigma_k| \end{aligned}$$

Thus, we can establish the quadratic discriminant functions as,

$$\delta_k(x) = \ln \pi_k - \frac{1}{2} \mu_k^T \Sigma_k^{-1} \mu_k + x^T \Sigma_k^{-1} \mu_k - \frac{1}{2} x^T \Sigma_k^{-1} x - \frac{1}{2} \ln |\Sigma_k|$$

Where the function is quadratic in  $x$ .

To estimate the final quadratic discriminant function we perform the following steps,

1. We obtain the decision boundaries where they are 0s of quadratic functions.
2. We find the estimates  $\hat{\mu}_k$  and  $\hat{\Sigma}_k$  for each response classes separately.
3. We estimate  $\hat{\pi}_k$ .

Then we can define the final decision rule, for an input of values, the class is predicted to be with the largest,

$$\delta_k(x) = \ln \hat{\pi}_k - \frac{1}{2} \hat{\mu}_k^T \hat{\Sigma}_k^{-1} \hat{\mu}_k + x^T \hat{\Sigma}_k^{-1} \hat{\mu}_k - \frac{1}{2} x^T \hat{\Sigma}_k^{-1} x - \frac{1}{2} \ln |\hat{\Sigma}_k|$$

[19]

In the case that data is not normal, we can transform the data to conform to be close to normal and test for the equality of covariance matrices. However, we can simply apply linear or quadratic discriminant analysis without worrying about the distributional assumptions of the population, and hope that it performs reasonably well. In this case it is important to check the performance of the classification model.

[14]

For our purpose, we apply these two discriminant techniques (we transform the data beforehand to make them close to normal) and evaluate the performances of the classification procedures for comparison with other techniques.

### 3.1.2 K-Nearest Neighbour

K-Nearest Neighbour (KNN) classification is one of the most simple and fundamental classification techniques. It is a non-parametric classification method first developed by Evelyn Fix and Joseph Hodges in 1951,[20] and later expanded by Thomas Cover. [21] It was developed from the need to perform discriminant analysis when reliable parametric estimates of probabilities are unknown or difficult to ascertain. [22]

The KNN algorithm calculates the distance between a new data point and all the points in your data set and predicts the class in which the new data point belongs to. This technique is instance-based learning where learning happens at prediction time and requires no parameters to tune. [23]

The common technique to measure the distance between a novel example and test example is known as the Minkowski distance,

$$D(X,Y) = \left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

Which can be considered as a generalization of both the Euclidean distance and the Manhattan distance. Minkowski distance is typically used with p.

When p = 1, we obtain the Manhattan distance,

$$D(X,Y) = \sum_{i=1}^n |x_i - y_i|$$

When p = 2, we obtain the Euclidean distance,

$$D(X,Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

In the limiting case of p reaching infinity, we obtain the Chebyshev distance,



$$\lim_{p \rightarrow \infty} D(X, Y) = \lim_{p \rightarrow \infty} \left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p} = \max_i (|x_i - y_i|)$$

The general steps of the KNN algorithm can be described as,

1. Training examples are set as vectors in a multidimensional feature space, each with a class label.
2. At the start of the training phase, we store the feature vectors and class labels of the training samples.
3. During the classification phase, we define a constant k and an unlabeled vector (a test point) is classified by assigning the label which is most frequent among the k training sample nearest to that test point. A commonly used distance metric for continuous variables in this step is Euclidean distance.
4. We repeat step 3 until all unlabeled vectors are classified.

The choice of k depends on the data. Larger values of k reduce the variance, but often at the cost of increased bias. [24] It can also lead to overfitting and reduce the generalization capability of the trained algorithm.

In our case, we compare the performance of the models among k = 5, k = 7, and k = 9.

Model validation methods such as cross-validation can be used to tune models to optimize the trade-off. We use K fold cross-validation to find the optimum k which provides the highest accuracy in predictions.

### 3.1.3 Multinomial Logistic regression

A commonly used regression model for classification is the Logistic Regression model, used when the outcome variable is binary. The model can be modified to handle the case where the output variable has more than two classes. In this process, the goal is to estimate the probability of choosing each of the outcome classes as well as to estimate the odds of the class choice in a functional form of the covariates and derive odds ratios for the choice of different plans. [25]

The estimated multinomial logistic regression coefficients are expressed in terms of the reference level.

An important feature of the multinomial logit model is that it estimates  $k-1$  models, where  $k$  is the number of levels of the outcome variable.

We assume the categories of the outcome variable,  $Y$ , to be coded as  $0, 1, 2, \dots, k$ , in the  $k$  outcome category model we shall need a total  $(k-1)$  number of logit functions. A reference category is chosen, say,  $Y = 0$ , and then form logit functions for all other categories,  $Y = 1, Y = 2, \dots, Y = k$  to compare against the reference category.

To develop the model, we assume to have  $p$  covariates and one constant term  $x$ , of length  $p+1$ , where  $x_0 = 1$ . Then we denote the logit functions as

$$g_1(x) = \ln \left[ \frac{\Pr(Y = 1|x)}{\Pr(Y = 0|x)} \right] = \beta_{10} + \beta_{11}x_1 + \beta_{12}x_2 + \cdots + \beta_{1p}x_p = \mathbf{x}'\boldsymbol{\beta}_1$$

$$g_2(x) = \ln \left[ \frac{\Pr(Y = 2|x)}{\Pr(Y = 0|x)} \right] = \beta_{20} + \beta_{21}x_1 + \beta_{22}x_2 + \cdots + \beta_{2p}x_p = \mathbf{x}'\boldsymbol{\beta}_2$$

.

.

.

$$g_k(x) = \ln \left[ \frac{\Pr(Y = k|x)}{\Pr(Y = 0|x)} \right] = \beta_{k0} + \beta_{k1}x_1 + \beta_{k2}x_2 + \cdots + \beta_{kp}x_p = \mathbf{x}'\boldsymbol{\beta}_k$$

We can then develop a general expression for the conditional probability in a  $k$  category model,

$$\pi_j(x) = \Pr(Y = j|x) = \frac{e^{g_j(x)}}{\sum_{j=0}^k e^{g_j(x)}}$$

Where the vector  $\beta_0 = 0$  and  $g_0(x) = 0$ . [25]

Now let us assume the response variable  $Y$  is coded as follows,

If  $Y = 0$  then  $Y_0 = 1, Y_1 = 0, Y_2 = 0, \dots, Y_k = 0$ ;

If  $Y = 1$  then  $Y_0 = 0, Y_1 = 1, Y_2 = 0, \dots, Y_k = 0$ ;

If  $Y = 2$  then  $Y_0 = 0, Y_1 = 0, Y_2 = 1, \dots, Y_k = 0$ ;

•  
•  
•

If  $Y = k$  then  $Y_0 = 0, Y_1 = 0, Y_2 = 0, \dots, Y_k = 1$ ;

Thus, for any value of  $Y$ , the  $\sum_{j=0}^k Y_j = 1$ .

Then using these notations, we can establish the conditional likelihood function for a sample of  $n$  independent observations as,

$$l(\beta) = \prod_{i=1}^n [\pi_0(x_i)^{y_{0i}} \pi_1(x_i)^{y_{1i}} \pi_2(x_i)^{y_{2i}} \dots \pi_k(x_i)^{y_{ki}}]$$

Taking natural log of the likelihood function, we get the log likelihood as follows,

$$L(\beta) = \ln l(\beta)$$

$$= \sum_{i=1}^n [y_{0i} \ln \pi_0(x_i) + y_{1i} \ln \pi_1(x_i) + y_{2i} \ln \pi_2(x_i) + \dots + y_{ki} \ln \pi_k(x_i)]$$

Now using  $\pi_j(x) = \Pr(Y = j|x) = \frac{e^{g_j(x)}}{\sum_{j=0}^k e^{g_j(x)}}$

$$= \sum_{i=1}^n [y_{0i} g_0(x_i) + y_{1i} g_1(x_i) + y_{2i} g_2(x_i) + \dots + y_{ki} g_k(x_i)$$

$$- \ln(e^{g_0(x_i)} + e^{g_1(x_i)} + e^{g_2(x_i)} + \dots + e^{g_k(x_i)})]$$

And since,  $g_0(x) = 0$  then,

$$= \sum_{i=1}^n [y_{1i}g_1(x_i) + y_{2i}g_2(x_i) + \dots + y_{ki}g_k(x_i) - \ln(1 + e^{g_1(x_i)} + e^{g_2(x_i)} + \dots + e^{g_k(x_i)})]$$

The likelihood equations are obtained by taking the first partial derivative of the log-likelihood functions with respect to each of the  $2(p+1)$  unknown parameters.

The simplified general form of the derivative is,

$$\frac{\delta L(\beta)}{\delta \beta_{jp}} = \sum_{i=1}^n x_{pi}(y_{ij} - \pi_{ij})$$

For  $j = 1, 2, \dots, k$  and  $p = 0, 1, 2, \dots, p$ , with  $x_{oi} = 1$  for each subject.

The Maximum likelihood estimator  $\hat{\beta}$  is obtained by setting the equations equal to zero and solving for  $\hat{\beta}$ . [25] But note that there are no close form solutions for the equations. We can use iterative numerical approaches, such as, Multivariate Newton's Method, to find the solutions.

The Odds Ratio is a widely used measure of association for logistic regression. It approximates how many times likely or unlikely it is for the outcome to be present for change in the predictor variables. The Odds ratio can be obtained by exponentiation of the multinomial logit coefficients for continuous variables, such that,

$$OR_j = e^{\hat{\beta}_j}$$

The Odds Ratio of a coefficient indicates how the odds of the outcome falling in the comparison group compared to the odds of the outcome falling in the reference group according to the changes with the variable in question.

An  $OR > 1$  indicates that the odds of the outcome falling in the comparison group relative to the odds of the outcome falling in the reference group increases as the predictor variable increases. In other words, the comparison outcome is more likely.

An  $OR < 1$  indicates that the odds of the outcome falling in the comparison group relative to the odds of the outcome falling in the reference group decreases as the predictor variable increases. Thus, the reference outcome is more likely. [25]

We can also obtain the test statistic  $z$  from the ratio of the coefficient and the standard error of the respective predictor, that is,

$$z_{ij} = \frac{\hat{\beta}_{ij}}{s.e(\hat{\beta}_{ij})}$$

Where  $i = 1, 2, \dots, k$  (model) and  $j = 1, 2, 3, \dots, p$  (parameters).

From the test statistic, P-values can also be computed. For a given alpha value,  $z$  and P-value determine whether or not the null hypothesis can be rejected. The null hypothesis, in this case, is if a particular predictor's regression coefficient is equal to zero, given that the rest of the predictors are in the model. If the p-value is less than

alpha then we may reject the null hypothesis and the parameter estimate is considered to be significant in the model.

In multinomial logistic regression, the significance of a parameter estimate is limited to the model in which the parameter estimate was calculated, such that the significance of a  $\hat{\beta}$  in the model for 'Black wins' vs 'Draw' cannot be assumed to hold for 'Black wins' vs 'White wins'.

## 3.2 Model Validation

After training and building a model, we are interested in determining the performance of the model.

The steps to validate the model are usually,

1. Split the data into training and test sets.
2. Use the training dataset to train the model.
3. Use the test dataset to validate the trained model by estimating performance metrics.

From this process, we are interested to know how accurate the model is in predicting the outcome for novel observations that were not used to train the model. To determine the performance we may consider using overall accuracy and the kappa statistic as the common performance metric for all models.

**Accuracy:** For a comparative performance measurement for classification, we use overall accuracy. It is a measure of how often the classifier is correct. It is calculated as the proportion of observations the model correctly classified, such that,

$$Accuracy ( Overall) = \frac{Correct\ Classifications}{Total\ Population}$$

However, only using accuracy to compare performances of models is not reliable. So, in addition to accuracy, we observe the Kappa statistic.

**Cohen's Kappa:** This is a statistic to measure how well the classifier performed as compared to how well it would have performed simply by chance. In other words, a model will have a high Kappa score if there is a big difference between the accuracy and the null error rate. It is a variation of accuracy that is corrected for category imbalances.

Cohen's Kappa, symbolized by the lower case Greek letter,  $\kappa$ , [26] is a robust statistic useful for either interrater or interrater reliability testing. It can range from  $-1$  to  $+1$ , similar to the correlation coefficient, where  $0$  represents the amount of agreement that can be expected from random chance, and  $1$  represents perfect agreement between the raters. While Kappa values below  $0$  are possible, it is unlikely to happen in practice. [26] As with all correlation statistics, the Kappa is a standardized value and thus can be interpreted similarly across multiple studies, which makes it a useful metric for comparison across models.

Cohen's Kappa estimate can be interpreted as follows,



Table 01. Cohen's Kappa interpretation table

<b>Value of Kappa</b>	<b>Level of Agreement</b>	<b>% of Data that are Reliable</b>
0-.20	None	0-4%
.21-.39	Minimal	4-15%
.40-.59	Weak	15-35%
.60-.79	Moderate	35-63%
.80-.90	Strong	64-81%
Above.90	Almost Perfect	82-100%

Calculation of Cohen's Kappa may be performed according to the following formula:

$$\kappa = \frac{\text{Pr}(o) - \text{Pr}(e)}{1 - \text{Pr}(e)}$$

Where,  $\text{Pr}(o)$  represents the actual observed agreement, and

$\text{Pr}(e)$  represents chance agreement. [27]

### 3.2.1 Confusion Matrix

A model validation technique to evaluate the performance of a classifier is to observe the confusion matrix or the error matrix. It is a cross-tabular layout of the performance of a machine learning classifier algorithm. Each row of the table represents the instances in a predicted category of the response variable, while each column represents the instances in an actual category of the response.

A typical 2x2 confusion matrix can be expressed in the following way,

Table 02. Example of a Confusion Matrix

Predicted Class	Actual Class (Reference)	
	Positive	Negative
Positive	A	B
Negative	C	D

Where, A is the number of True Positive (TP).

B is the number of False positives (FP) [Also known as Type I error]

C is the number of False Negative (FN) [Also known as Type II error]

D is the number of True Negative (TN)

The first performance measurement we consider for performance measurement is the accuracy. Accuracy is calculated as,

$$Accuracy (Overall) = \frac{A + D}{A + B + C + D}$$

The Kappa statistic is also calculated using,

$$\kappa = \frac{\Pr(o) - \Pr(e)}{1 - \Pr(e)}$$

Where,  $\Pr(o) = \frac{A+D}{A+B+C+D}$ , represents the observed accuracy, and

$$\Pr(e) = \frac{\frac{(A+C)(A+B)}{A+B+C+D} + \frac{(B+D)(C+D)}{A+B+C+D}}{A+B+C+D}, \text{ represents the expected accuracy.}$$

A p-value can be calculated using McNemar's test is also computed. The overall accuracy rate is computed along with a 95 percent confidence interval and a one-sided test to see if the accuracy is better than the "no information rate," which is taken to be the largest class percentage in the data. This process is carried out using software. [28]

From the confusion matrix we can calculate a few additional performance measurements. [28] They are listed below.

**Sensitivity (True positive rate):** A measure of the proportion of True positives that are correctly classified. It tells us how often a classifier predicts positive when the response of the observation is actually positive. It is also often known as Recall, another metric.

$$Sensitivity = \frac{A}{A + C}$$

$$Recall = \frac{A}{A + C}$$

**Specificity (True negative rate):** A measure of the proportion of negatives that are correctly classified. It tells us how often a classifier predicts negative when the response of the observation is actually negative.

$$Specificity = \frac{D}{B + D}$$

**Prevalence:** A measure of proportion of True positives and total observations. It tells us how often positives occurs in the sample.

$$Prevalence = \frac{(A + C)}{(A + B + C + D)}$$

**Positive Predictive Value (Precision):** A measure of the proportion of correct true positive predictions and total number of positive predictions. This tells us how often the classifier is correct when predictive positives.

$$PPV = \frac{Sensitivity * Prevalence}{Sensitivity * Prevalence + [(1 - specificity)(1 - prevalence)]}$$

$$Precision = \frac{A}{A + B}$$

**Negative Predictive Value:** A measure of proportion of True negatives and total number of negative predictions.

$$NPV = \frac{Sensitivity * (1 - Prevalence)}{(1 - Sensitivity) * Prevalence + Specificity * (1 - Prevalence)}$$

**Detection rate:** Measure of proportion of True positives and Total observations.

$$Detection Rate = \frac{A}{A + B + C + D}$$

Detections prevalence: A measure of proportion of positive predictions and total observations.

$$Detection\ Prevalence = \frac{A + B}{A + B + C + D}$$

**Balanced accuracy:** It is the average of the proportions of correct classifications of each classes individually. It is calculated by the average of sensitivity and specificity.

$$Balanced\ Accuracy = \frac{sensitivity + Sppecificity}{2}$$

**F Score:** This metric measures the weighted average (harmonic mean) of the true positive rate (recall) and precision. The general formula for calculating the F Score is,

$$F_h = \frac{(1 + h^2)(Precision * Recall)}{h^2(Precision * Recall)}$$

We usually look at the F score for  $h = 1$ , known as the F1 score. It is calculated as follows,

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{A}{A + \frac{B + C}{2}}$$

For more than two classes, these results are calculated comparing each factor level to the remaining levels (i.e., a "one versus all" approach). [28]

## 3.2.2 K-fold Cross-Validation

Cross-validation is a statistical technique to validate a model and find the performance measurements. It is a technique that provides a measure of model performance for out of model data predictions. Cross-validation is also known as a resampling method as it involved fitting the same method multiple times using different subsets of the data.

There are many different cross-validation techniques, differing in method and complexity. K-fold Cross-validation is a robust method for estimating accuracies. In K-fold cross-validation, the training set is randomly partitioned into k equal-sized portions or folds. Then the accuracy is calculated.

The algorithm for K-fold Cross-Validation is as follows,

1. Data is split into k subsets or folds.
2. Reserve one subset and train the model on all other subsets.
3. The trained model is tested against the reserved subset and accuracy and kappa statistic is calculated.
4. The process is repeated until each of the k subsets has served for testing.
5. The average accuracy of the k resampled process is calculated. This metric will be known as cross-validation accuracy.

A graphic showing how the K Fold CV process operates is provided below,

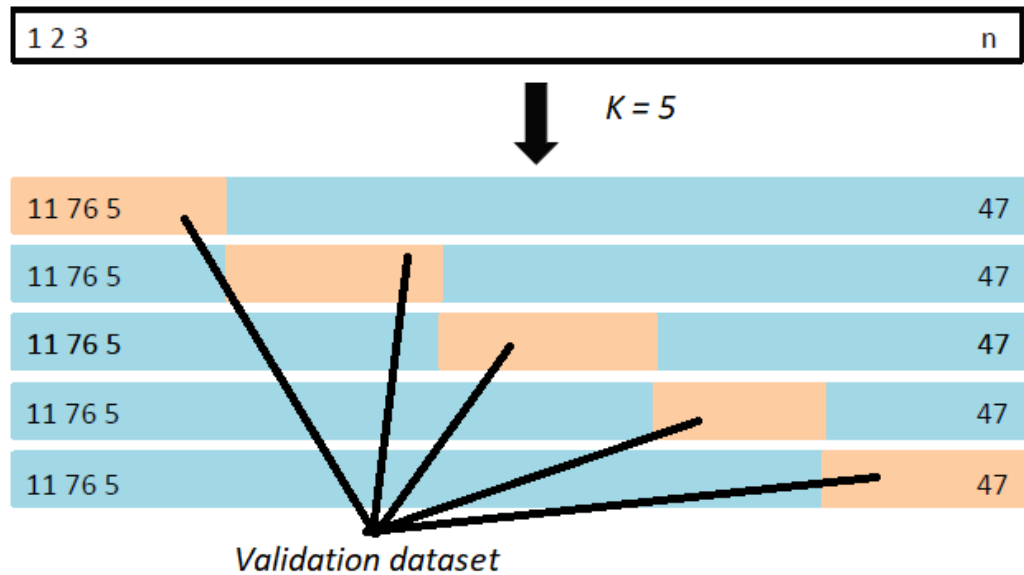


Figure 01. The validation process in K Fold Cross-Validation with 5 folds. [31]

The choice of  $K$  in this method is predicated by the bias-variance tradeoff. Lower values of  $K$  leads to lower variance, but higher bias. Whereas an increase in  $K$  cause the bias to lower, but variance increases. The solution in this situation is to strike the right balance between bias and variance.

However, in practice, we typically set  $K = 5$  or  $K = 10$ , as these values have been shown empirically to yield accuracy metrics that generally don't have a high bias or high variance. [29]

In this paper, we compare the performance of the models among  $k = 10$ .

An advantage of the K-fold cross-validation process is that it is very robust, computationally inexpensive, and comparatively more accurate than evaluating the confusion matrix.

### 3.2.3 Leave One Out Cross-Validation

Leave-One-Out Cross-Validation (LOOCV) is a particular case of K-fold cross-validation where  $k = n$ , that is, each observation serves as a subset of the data.

The algorithm of Leave One Out Cross-Validation is as follows,

1. Leave one data observation out and build the model with the remaining data set.
2. The trained model is tested against the data point that was left out in step 1 and accuracy and kappa statistic is calculated.
3. The process is repeated for all data points.
4. The average accuracy of the resampled processes is calculated.

A graphic showing how the LOOCV process operates is provided below,



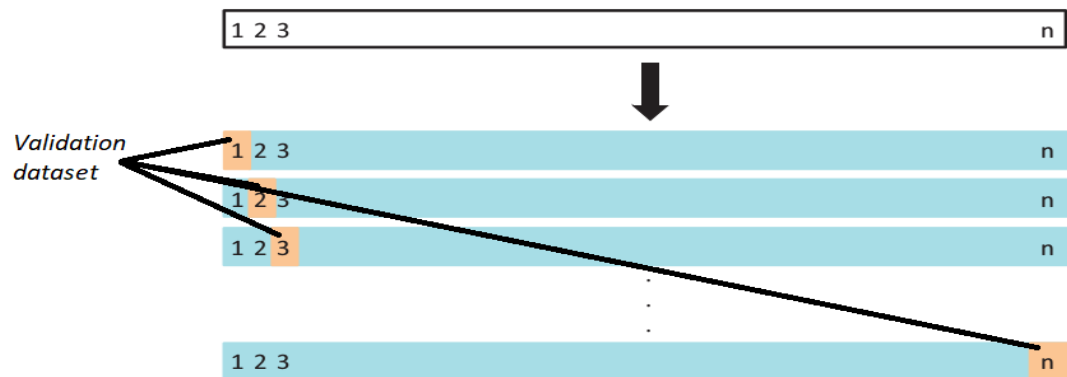


Figure 02. The validation process of Leave One Out Cross-Validation. [31]

The advantage of the LOOCV process is that, since we use all data points, we reduce potential bias. However, the disadvantage of the process also lies in that fact. Since the process is repeated as many times as there are data points, the process is computationally expensive when  $n$  is extremely large. Moreover, if there are outliers in the data, it may result in higher variation in the prediction accuracy as in each iteration we test the model performance against one data point.

### 3.3 Technology use

For data preprocessing, organization, and analysis we employ Python and R studio. Parsing the chess game database and processing it into a comma-separated values file was carried out using python libraries numpy and pandas. The python coding was written and executed in Spyder.

The rest of the data organizing and analysis was carried out in R.

The Classification and Regression Training package, authored by Max Kuhn, contains several functions for training and plotting classification and regression models. We extensively use functions and operations available in the caret package. It contains functions to help in preprocessing the data. Furthermore, it includes functions that calculate the performance measurements of trained models. [28] We also use the nnet package, which includes functions to run multinomial logistic regression. In conjunction with caret, we can code and execute the multinomial logistic regression technique as a classifier.

# Chapter 4: Results

## 4.1 Phase I

Let us first take a look at the summaries of the selected attributes from the Over-Board chess games. The data contains 1942330 observations.

result	welo	belo	length
black:554707	Min. : 1	Min. : 1	Min. : 1.00
draw :672476	1st Qu.:2132	1st Qu.:2130	1st Qu.: 55.00
white:715147	Median :2295	Median :2293	Median : 75.00
	Mean :2242	Mean :2239	Mean : 76.97
	3rd Qu.:2430	3rd Qu.:2430	3rd Qu.: 96.00
	Max. :2851	Max. :2851	Max. :600.00

Figure 03. Summary table of the over the board chess game attributes.

The bar plot showing the frequencies of the result of the games is,

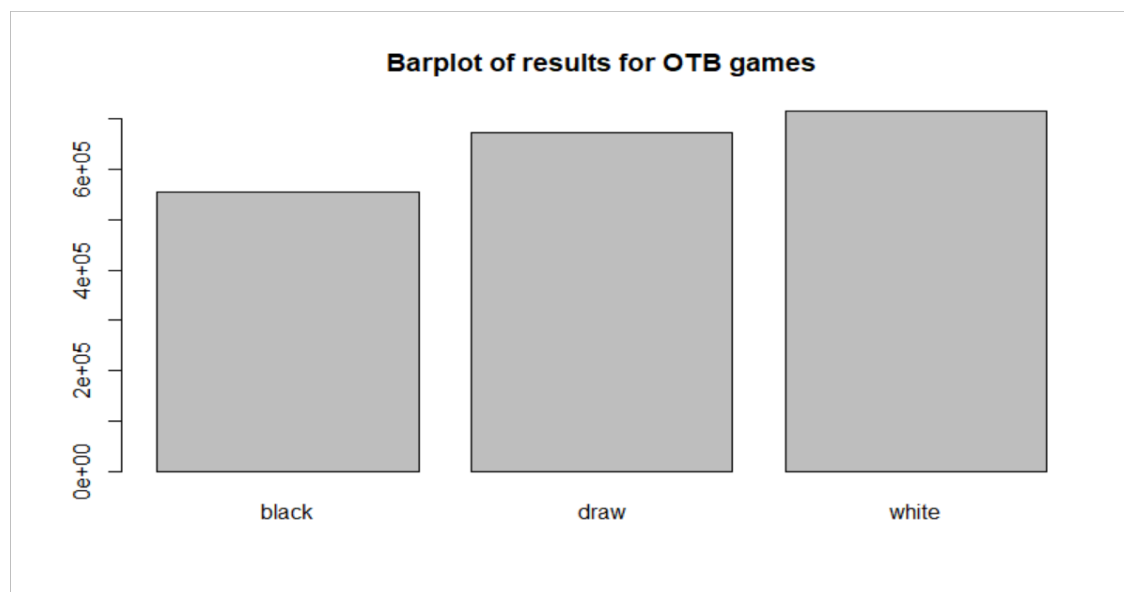


Figure 04. Bar plot of the results of the Over-the-board chess games.

As we can see, the proportions of each level of the response, results, are slightly different. We can obtain the frequencies and proportions of the result responses,

Table 03. Frequency and proportions of result responses for OTB games.

Result level	Black wins	Draw	White wins
Frequency	554707	672476	715147
Proportion	0.2855884	0.3462213	0.3681903

Here, we see that white players won more than black players, with draws being proportionally in the middle. It is consistent with chess theory that the white players start the game with an inherent advantage due to having the first move. [30]

These winning percentages are consistent and often occur in over the board chess tournaments. For example, during the World Blitz Championship 2009, approximately 26.40% of the games were drawn, 38.96% were won by White and 34.63% were won by Black. [31]

We may want to balance the responses before training, but since the training dataset is quite large and the proportions are close enough, it should not create any severe bias during training. Additionally, it would enable us to train the machine in accordance with the real-world situation.

The histograms of the predictor attributes are as follows,

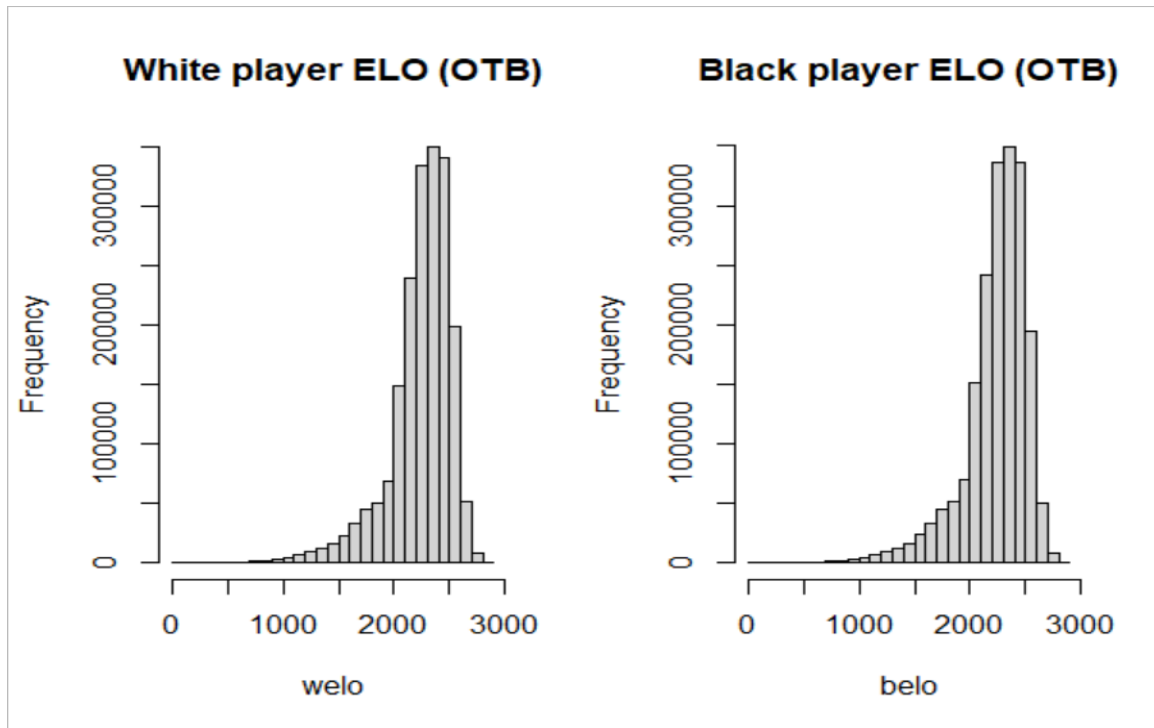


Figure 05. Histogram of white and black players ELO rating for Over the Board chess games.

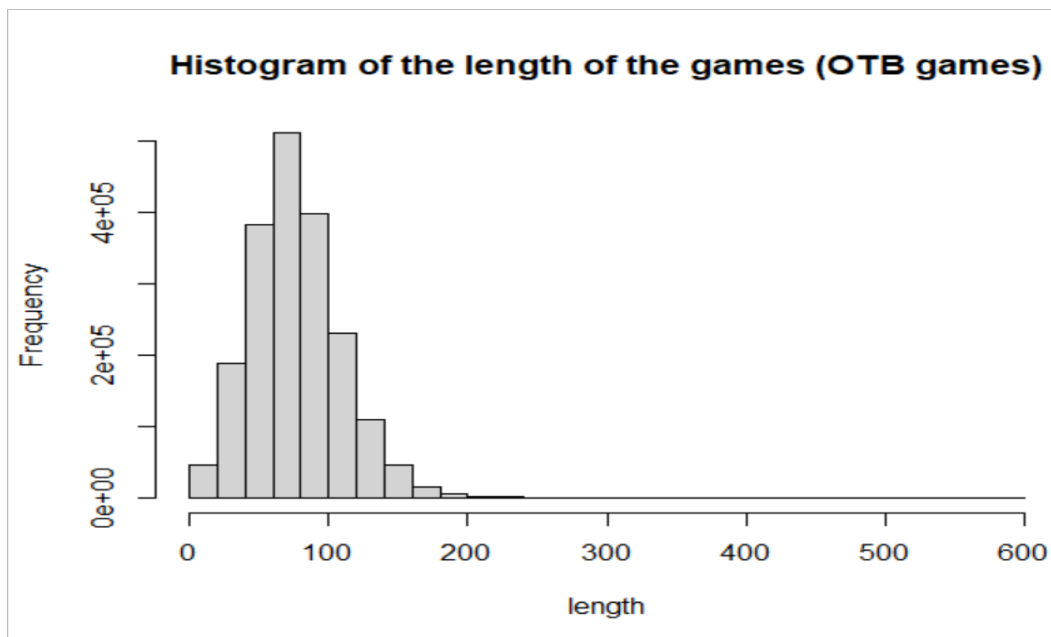


Figure 06. Histogram of number of turns or length of Over the Board chess games.

Next, we take a look at the online chess games database. The data contains 20058 observations. The summaries of the attributes for this data is as follows,

result	welo	belo	length
black: 9107	Min. : 784	Min. : 789	Min. : 1.00
draw : 950	1st Qu.:1398	1st Qu.:1391	1st Qu.: 37.00
white:10001	Median :1567	Median :1562	Median : 55.00
	Mean :1597	Mean :1589	Mean : 60.47
	3rd Qu.:1793	3rd Qu.:1784	3rd Qu.: 79.00
	Max. :2700	Max. :2723	Max. :349.00

Figure 07. Summary table of the over the Online chess game attributes.

The bar plot showing the frequencies of the result of the games is,

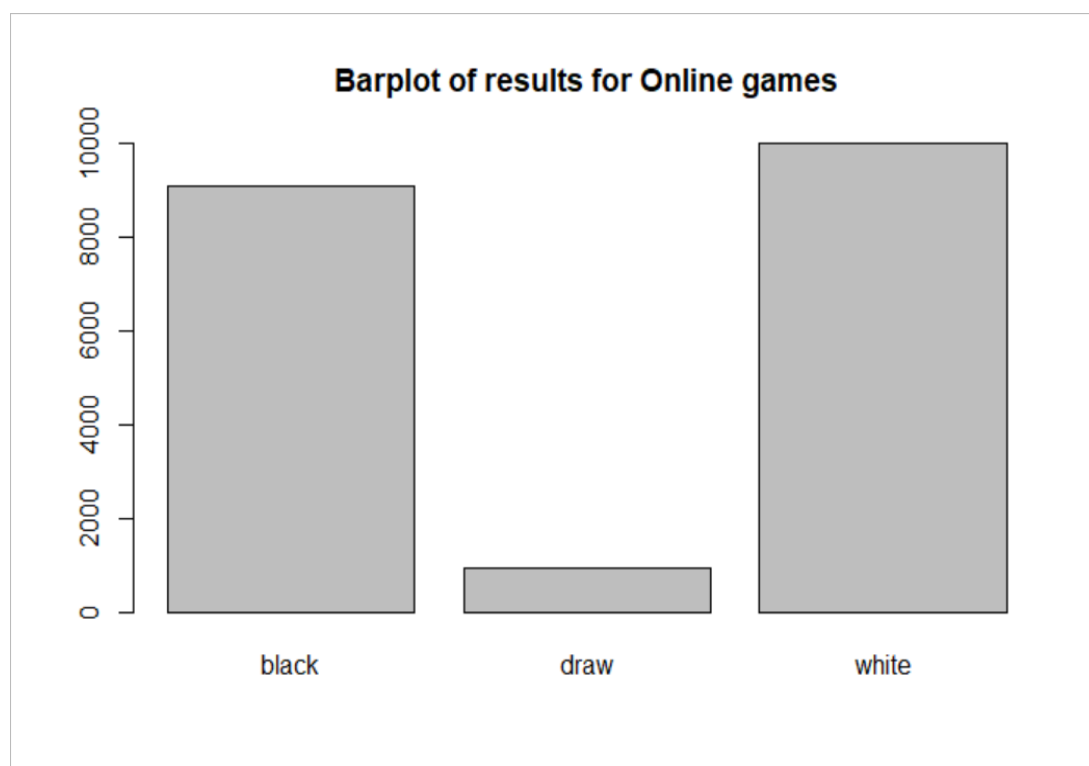


Figure 08. Bar plot of the results of the online chess games.

Here, we can see the number of games drawn in this database is much less compared to black or white winning. We can look at the frequencies and proportions on the table below,

Table 04. Frequency and proportions of result responses for online games.

Result level	Black wins	Draw	White wins
Frequency	9107	950	10001
Proportion	0.45403330	0.04736265	0.49860405

This is a big departure from the consistent response results for over the board games. But, it may be explained by the behavioural changes in online play. In online rated chess games, players earn more rating points if they win compared to drawing the game. Thus, players in an online setting tend to play to win and decline offers of draw. Still, the first move advantage that inherently gives advantage to white exists, as we see that white is winning slightly more compared to black. But, only a small number of the games are drawn.

This imbalanced proportions in the responses in the test set may yield to poorer accuracy performances by the classifiers, particularly when classifying drawn games, and leading to poorer overall accuracy. We shall explore later whether balancing the data improves any of these aspects.

The histograms of the predictor attributes are as follows,

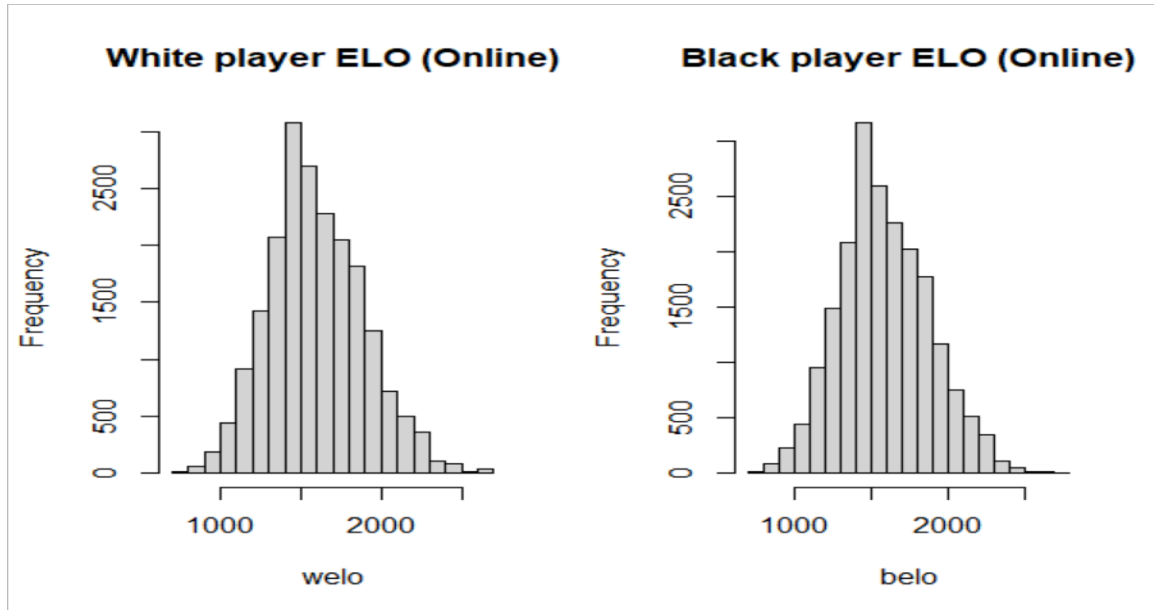


Figure 9. Histogram of white and black players ELO rating for Online chess games.

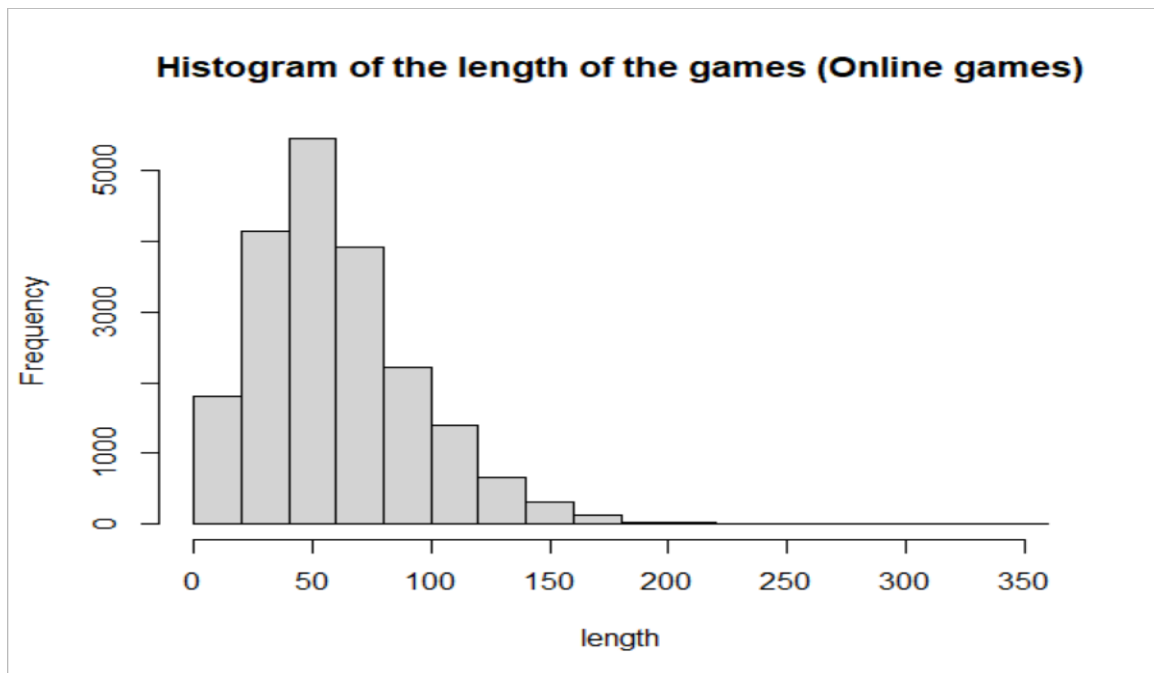


Figure 10. Histogram of number of turns or length of Online chess games.



We can carry out Hypothesis tests to check the significance of the model. Since the response variables are categorical, we use logistic regression.

The hypothesis is,

$H_0$ : The variables are not significant in the model.

$H_1$ : Atleast one of the are not significant in the model.

From logistic regression, setting the response black as the reference category.

The test statistic values,

	(Intercept)	welo	belo	length
draw	-37543899	319.5870	-226.2284	-238.02285
white	-1287253	544.4003	-529.8416	-44.65136

And corresponding P values,

	(Intercept)	welo	belo	length
draw	0	0	0	0
white	0	0	0	0

Let  $\alpha = 0.05$ . Here, P-values are  $< 0.05$ . Thus, all three variables are significant in the model.

Therefore, we shall use welo, belo and length of the game as predictor variables, with result being the response variable with three categories, black, white, and draw, when building the machine learning classifiers. From this point onwards, the standardized dataset with 1962388 observations is used for developing the

classifiers with 1942330 OTB games as training set and 20058 online games as testing set.

### 4.1.1 Linear Discriminant Analysis

Using the Linear Discriminant Analysis algorithm we train the model based on welo, belo, and length as the predictors and validate using testing dataset.

Let us observe the results when using Confusion Matrix as the method for validating the model.

#### Confusion Matrix and Statistics

	Reference		
Prediction	black	draw	white
black	3547	2969	2591
draw	308	224	418
white	1545	2973	5483

#### Overall Statistics

Accuracy : 0.4614  
 95% CI : (0.4544, 0.4683)  
 No Information Rate : 0.4234  
 P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.174

Mcnemar's Test P-Value : < 2.2e-16

#### Statistics by Class:

	Class: black	Class: draw	Class: white
Sensitivity	0.6569	0.03633	0.6457
Specificity	0.6207	0.94774	0.6094
Pos Pred Value	0.3895	0.23579	0.5482
Neg Pred Value	0.8308	0.68903	0.7008
Prevalence	0.2692	0.30741	0.4234
Detection Rate	0.1768	0.01117	0.2734
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.6388	0.49203	0.6275

Here, the accuracy is 46.14% with Kappa statistic 0.174 (No agreement).

Thus, the LDA classifier is slightly worse than a coin toss and it is performing no better compared to classifying simply by chance.

Additionally,

From Sensitivities by class, Black winning, draw, and white winning occurring is correctly classified for each response class is 65.69%, 3.633%, and 64.57% respectively.

From Specificities by class, Black winning, draw, and white winning not occurring is correctly classified for each response class is 62.07%, 94.774%, and 60.94% respectively.

From Positive Predictive Value by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total occurrences predicted for each response class is 38.95%, 23.579%, and 54.82% respectively.

From Negative Predictive Value by class, the percentage of Black winning, draw, and white winning not occurring is correctly classified with respect to total non-occurrences predicted for each response class is 83.08%, 68.903%, and 70.08% respectively.

From Prevalence by class, the percentage of Black winning, draw, and white winning actually occurring with respect to total observations in the sample for each response class is 26.92%, 30.741%, and 42.34% respectively.

From Detection rate by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total observations for each response class is 17.68%, 1.117%, and 27.34% respectively.

From Detection Prevalence by class, the percentage of Black winning, draw, and white winning occurring is predicted by the classifier with respect to total observations for each response class is 45.40%, 4.736% and 49.86% respectively.

From Balanced Accuracy by class, the average percentage of Black winning, draw, and white winning correctly classified by the classifier balanced for each response class is 63.88%, 49.203% and 62.75% respectively.

Let us observe the results when using K-fold Cross-Validation as the method for validating the model.

```
Linear Discriminant Analysis
1962388 samples
    3 predictor
    3 classes: 'black', 'draw', 'white'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1748097, 1748097, 1748097, 1748099,
1748096, 1748097, ...
Resampling results:

    Accuracy    Kappa
    0.5460607   0.3120672
```

Here, the accuracy is 54.60% with Kappa statistic 0.3121 (Minimal agreement).

Thus, the LDA classifier is slightly better than a coin toss and it is performing minimally better compared to classifying simply by chance.

## 4.1.2 Quadratic Discriminant Analysis

Using Quadratic Discriminant Analysis algorithm we train the model based on welo, belo, and length as the predictors and validate using testing dataset.

Let us observe the results when using Confusion Matrix as the method for validating the model.

Confusion Matrix and Statistics

	Reference		
Prediction	black	draw	white
black	3358	3452	2297
draw	180	434	336
white	1322	3525	5154

Overall Statistics

Accuracy : 0.446  
 95% CI : (0.4391, 0.4529)  
 No Information Rate : 0.3882  
 P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.184

Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

	Class: black	Class: draw	Class: white
Sensitivity	0.6909	0.05856	0.6619
Specificity	0.6217	0.95920	0.6050
Pos Pred Value	0.3687	0.45684	0.5153
Neg Pred Value	0.8628	0.63486	0.7382
Prevalence	0.2423	0.36948	0.3882
Detection Rate	0.1674	0.02164	0.2570
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.6563	0.50888	0.6334

Here, the accuracy is 44.60% with Kappa statistic 0.184 (No agreement).

Thus the QDA classifier is slightly worse than a coin toss and it is performing no better compared to classifying simply by chance.

Additionally,

From Sensitivities by class, Black winning, draw, and white winning occurring is correctly classified for each response class is 69.09%, 5.856%, and 66.19% respectively.

From Specificities by class, Black winning, draw, and white winning not occurring is correctly classified for each response class is 62.17%, 95.92%, and 60.50% respectively.

From Positive Predictive Value by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total occurrences predicted for each response class is 36.87%, 45.684%, and 51.53% respectively.

From Negative Predictive Value by class, the percentage of Black winning, draw, and white winning not occurring is correctly classified with respect to total non-occurrences predicted for each response class is 86.28%, 63.486%, and 73.82% respectively.

From Prevalence by class, the percentage of Black winning, draw, and white winning actually occurring with respect to total observations in the sample for each response class is 24.23%, 36.948%, and 38.82% respectively.

From Detection rate by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total observations for each response class is 16.74%, 2.164%, and 25.70% respectively.

From Detection Prevalence by class, the percentage of Black winning, draw, and white winning occurring is predicted by the classifier with respect to total observations for each response class is 45.40%, 4.736% and 49.86% respectively.

From Balanced Accuracy by class, the average percentage of Black winning, draw, and white winning correctly classified by the classifier balanced for each response class is 65.63%, 50.888% and 63.34% respectively.

Let us observe the results when using K-fold Cross-Validation as the method for validating the model.

Quadratic Discriminant Analysis

```
1962388 samples
      3 predictor
      3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 1748097, 1748098, 1748098, 1748096,  
1748096, 1748097, ...

Resampling results:

Accuracy	Kappa
0.5464195	0.3112956

Here the accuracy is 54.64% with Kappa statistic 0.3113 (Minimal agreement).

Thus, the QDA classifier is slightly better than a coin toss and it is performing minimally better compared to classifying simply by chance.

### 4.1.3 Multinomial Logistic Classification

Using Multinomial Logistic Regression algorithm we train the model based on welo, belo, and length as the predictors and validate using testing dataset. In this situation, where total response categories,  $k = 3$ , the outcome “Black wins” is defined as the reference group (as  $Y = 0$ ), and therefore the estimated coefficients describe a model for “Draws” (for  $Y = 1$ ), relative to “Black wins” and a model for “White wins” (for  $Y = 2$ ), relative to “Black wins”.

Since the parameter estimates are relative to the reference group, the standard interpretation of the multinomial logit is that for a unit change in the predictor variable, the logit of outcome relative to the reference group is expected to change by its respective parameter estimate (which is in log-odds units) given the variables in the model are held constant.

The results are,

Coefficients:

	(Intercept)	swelo	sbelo	slength
draw	0.3514169	1.180981	-0.8500516	-0.46962649
white	0.3118366	2.226885	-2.1642534	-0.08502437

Std. Errors:

	(Intercept)	swelo	sbelo	slength
draw	0.002085030	0.003771618	0.003968872	0.001993636
white	0.002118592	0.004242222	0.004267672	0.001955604

Residual Deviance: 3722625

AIC: 3722641



So, we may write the fitted logit model for black vs draw category for standardized values as,

$$g_1(x) = \ln \left[ \frac{\Pr(Y = 1|x)}{\Pr(Y = 0|x)} \right] = 0.3514169 + 1.180981 (\text{welo}) - 0.8500516 (\text{belo}) - 0.46962649 (\text{length})$$

And the fitted logit model for black vs white category for the standardized values is,

$$g_2(x) = \ln \left[ \frac{\Pr(Y = 1|x)}{\Pr(Y = 0|x)} \right] = 0.3514169 + 1.180981 (\text{welo}) - 0.8500516 (\text{belo}) - 0.46962649 (\text{length})$$

The Odds ratios are calculated as follows,

	(Intercept)	swelo	sbelo	slength
draw	1.421080	3.257567	0.4273929	0.6252358
white	1.365931	9.270946	0.1148356	0.9184899

Based on the results,

For 1 unit increase in the standardized value of welo, the odds of draw occurring is 3.257567 times likely compared to black winning.

For 1 unit increase in the standardized value of belo, the odds of draw winning is 0.4273929 times likely compared to black winning.

For 1 unit increase in the standardized value of length, the odds of draw winning is 0.6252358 times likely compared to black winning.

For 1 unit increase in the standardized value of welo, the odds of white winning is 9.270946 times likely compared to black winning.

For 1 unit increase in the standardized value of belo, the odds of white winning is 0.1148356 times likely compared to black winning.

For 1 unit increase in the standardized value of length, the odds of white winning is 0.9184899 times likely compared to black winning.

Let us observe the results when using Confusion Matrix as the method for validating the model.

#### Confusion Matrix and Statistics

		Reference		
Prediction	black	draw	white	
black	3634	2864	2609	
draw	315	205	430	
white	1587	2868	5546	

#### Overall Statistics

Accuracy : 0.4679  
 95% CI : (0.461, 0.4748)  
 No Information Rate : 0.428  
 P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.1779

Mcnemar's Test P-Value : < 2.2e-16

#### Statistics by Class:

	Class: black	Class: draw	Class: white
Sensitivity	0.6564	0.03453	0.6460
Specificity	0.6231	0.94724	0.6117
Pos Pred Value	0.3990	0.21579	0.5545
Neg Pred Value	0.8263	0.70002	0.6978
Prevalence	0.2760	0.29599	0.4280
Detection Rate	0.1812	0.01022	0.2765
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.6398	0.49089	0.6289

Here the accuracy is 46.79% with Kappa statistic 0.1779 (No agreement).

Thus the Multinomial logistic classifier is slightly worse than a coin toss and it is performing no better compared to classifying simply by chance.

Additionally,

From Sensitivities by class, Black winning, draw, and white winning occurring is correctly classified for each response class is 65.64%, 3.453%, and 64.60% respectively.

From Specificities by class, Black winning, draw, and white winning not occurring is correctly classified for each response class is 62.31%, 94.724%, and 61.17% respectively.

From Positive Predictive Value by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total occurrences predicted for each response class is 39.90%, 21.579%, and 55.45% respectively.

From Negative Predictive Value by class, the percentage of Black winning, draw, and white winning not occurring is correctly classified with respect to total non-occurrences predicted for each response class is 82.63%, 70.002%, and 69.78% respectively.

From Prevalence by class, the percentage of Black winning, draw, and white winning actually occurring with respect to total observations in the sample for each response class is 27.60%, 29.599%, and 42.80% respectively.

From Detection rate by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total observations for each response class is 18.12%, 1.022%, and 27.65% respectively.

From Detection Prevalence by class, the percentage of Black winning, draw, and white winning occurring is classified by the classifier with respect to total observations for each response class is 45.40%, 4.736% and 49.86% respectively.

From Balanced Accuracy by class, the average percentage of Black winning, draw, and white winning correctly classified by the classifier balanced for each response class is 63.98%, 49.089% and 62.89% respectively.

Let us observe the results when using K-fold Cross-Validation as the method for validating the model.

Penalized Multinomial Regression

```
1962388 samples
      3 predictor
      3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 1748097, 1748096, 1748098, 1748096, 1748097, 1748098, ...

Resampling results across tuning parameters:

decay	Accuracy	Kappa
0e+00	0.5498788	0.3181662
1e-04	0.5498808	0.3181694
1e-01	0.5498777	0.3181646

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was decay = 1e-04.

Here, since we are not using penalized multinomial regression, we only consider the decay at 0.

Here, the accuracy is 54.99% with Kappa statistic 0.3182 (Minimal agreement).

Thus the Multinomial logistic classifier is slightly better than a coin toss and it is performing minimally better compared to classifying simply by chance.

## 4.1.4 K Nearest Neighbour Classifier

Using K Nearest Neighbour algorithm we train the model based on welo, belo, and length as the predictors and validate using testing dataset. We train and compare the accuracies for three values of k. We use,  $k = 5$ ,  $k = 7$ ,  $k = 9$ .

**Setting K = 5**, let us observe the results when using Confusion Matrix as the method for validating the model.

### Confusion Matrix and Statistics

	Reference		
Prediction	black	draw	white
black	3587	2629	2891
draw	257	347	346
white	2501	2780	4720

### Overall Statistics

```

Accuracy : 0.4314
95% CI : (0.4246, 0.4383)
No Information Rate : 0.3967
P-Value [Acc > NIR] : < 2.2e-16

```

```
Kappa : 0.1185
```

```
McNemar's Test P-Value : < 2.2e-16
```

Statistics by Class:

	Class: black	Class: draw	Class: white
Sensitivity	0.5653	0.06028	0.5932
Specificity	0.5975	0.95784	0.5636
Pos Pred Value	0.3939	0.36526	0.4720
Neg Pred Value	0.7482	0.71692	0.6781
Prevalence	0.3163	0.28697	0.3967
Detection Rate	0.1788	0.01730	0.2353
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.5814	0.50906	0.5784

Here, the accuracy is 43.14% with Kappa statistic 0.1185 (No agreement).

Thus, the KNN classifier (K=5) is slightly worse than a coin toss and it is performing no better compared to classifying simply by chance.

Additionally,

From Sensitivities by class, Black winning, draw, and white winning occurring is correctly classified for each response class is 56.53%, 6.028%, and 59.32% respectively.

From Specificities by class, Black winning, draw, and white winning not occurring is correctly classified for each response class is 59.75%, 95.784%, and 56.36% respectively.

From Positive Predictive Value by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total occurrences predicted for each response class is 39.39%, 36.526%, and 47.20% respectively.

From Negative Predictive Value by class, the percentage of Black winning, draw, and white winning not occurring is correctly classified with respect to total non-occurrences predicted for each response class is 74.82%, 71.692%, and 67.81% respectively.

From Prevalence by class, the percentage of Black winning, draw, and white winning actually occurring with respect to total observations in the sample for each response class is 31.63%, 28.697%, and 39.67% respectively.

From Detection rate by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total observations for each response class is 17.88%, 1.730%, and 23.53% respectively.

From Detection Prevalence by class, the percentage of Black winning, draw, and white winning occurring is classified by the classifier with respect to total observations for each response class is 45.40%, 4.736% and 49.86% respectively.

From Balanced Accuracy by class, the average percentage of Black winning, draw, and white winning correctly classified by the classifier balanced for each response class is 58.14%, 50.906% and 57.84% respectively.

**Setting K = 7**, let us observe the results when using Confusion Matrix as the method for validating the model.

Confusion Matrix and Statistics

	Reference		
Prediction	black	draw	white
black	3695	2505	2907
draw	247	346	357
white	2482	2606	4913

#### Overall Statistics

Accuracy : 0.4464  
 95% CI : (0.4395, 0.4533)  
 No Information Rate : 0.4077  
 P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.1329

McNemar's Test P-Value : < 2.2e-16

#### Statistics by Class:

	Class: black	Class: draw	Class: white
Sensitivity	0.5752	0.06340	0.6008
Specificity	0.6031	0.95863	0.5718
Pos Pred Value	0.4057	0.36421	0.4913
Neg Pred Value	0.7508	0.73252	0.6754
Prevalence	0.3203	0.27206	0.4077
Detection Rate	0.1842	0.01725	0.2449
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.5891	0.51102	0.5863

Here, the accuracy is 44.64% with Kappa statistic 0.1329 (No agreement).

Thus, the KNN classifier (K=7) is slightly worse than a coin toss and it is performing no better compared to classifying simply by chance.

Additionally,

From Sensitivities by class, Black winning, draw, and white winning occurring is correctly classified for each response class is 57.52%, 6.340%, and 60.08% respectively.



From Specificities by class, Black winning, draw, and white winning not occurring is correctly classified for each response class is 60.31%, 95.863%, and 57.18% respectively.

From Positive Predictive Value by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total occurrences predicted for each response class is 40.57%, 36.421%, and 49.13% respectively.

From Negative Predictive Value by class, the percentage of Black winning, draw, and white winning not occurring is correctly classified with respect to total non-occurrences predicted for each response class is 75.08%, 73.252%, and 67.54% respectively.

From Prevalence by class, the percentage of Black winning, draw, and white winning actually occurring with respect to total observations in the sample for each response class is 32.03%, 27.206%, and 40.77% respectively.

From Detection rate by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total observations for each response class is 18.42%, 1.725%, and 24.49% respectively.

From Detection Prevalence by class, the percentage of Black winning, draw, and white winning occurring is classified by the classifier with respect to total observations for each response class is 45.40%, 4.736% and 49.86% respectively.

From Balanced Accuracy by class, the average percentage of Black winning, draw, and white winning correctly classified by the classifier balanced for each response class is 58.91%, 51.102% and 58.63% respectively.

**Finally, setting K = 9**, let us observe the results when using Confusion Matrix as the method for validating the model.

#### Confusion Matrix and Statistics

	Reference		
Prediction	black	draw	white
black	3754	2466	2887
draw	262	332	356
white	2408	2560	5033

#### Overall Statistics

Accuracy : 0.4546  
 95% CI : (0.4477, 0.4616)  
 No Information Rate : 0.4126  
 P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.1428

Mcnemar's Test P-Value : < 2.2e-16

#### Statistics by Class:

	Class: black	Class: draw	Class: white
Sensitivity	0.5844	0.06196	0.6081
Specificity	0.6074	0.95796	0.5783
Pos Pred Value	0.4122	0.34947	0.5032
Neg Pred Value	0.7562	0.73697	0.6775
Prevalence	0.3203	0.26713	0.4126
Detection Rate	0.1872	0.01655	0.2509
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.5959	0.50996	0.5932

Here the accuracy is 45.46% with Kappa statistic 0.1428 (No agreement).

Thus the KNN classifier ( $K=9$ ) is slightly worse than a coin toss and it is performing no better compared to classifying simply by chance.

Additionally,

From Sensitivities by class, Black winning, draw, and white winning occurring is correctly classified for each response class is 58.44%, 6.196%, and 60.81% respectively.

From Specificities by class, Black winning, draw, and white winning not occurring is correctly classified for each response class is 60.74%, 95.796%, and 57.83% respectively.

From Positive Predictive Value by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total occurrences predicted for each response class is 41.22%, 34.947%, and 50.32% respectively.

From Negative Predictive Value by class, the percentage of Black winning, draw, and white winning not occurring is correctly classified with respect to total non-occurrences predicted for each response class is 75.62%, 73.697%, and 67.75% respectively.

From Prevalence by class, the percentage of Black winning, draw, and white winning actually occurring with respect to total observations in the sample for each response class is 32.03%, 26.713%, and 41.26% respectively.

From Detection rate by class, the percentage of Black winning, draw, and white winning occurring is correctly classified with respect to total observations for each response class is 18.72%, 1.655%, and 25.09% respectively.

From Prevalence by class, the percentage of Black winning, draw, and white winning actually occurring with respect to total observations in the sample for each response class is 45.40%, 4.736% and 49.86% respectively.

From Balanced Accuracy by class, the average percentage of Black winning, draw, and white winning correctly classified by the classifier balanced for each response class is 59.59%, 50.996% and 59.32% respectively.

Let us observe the results when using K-fold Cross-Validation as the method for validating the model.

k-Nearest Neighbors

```
1962388 samples
    3 predictor
    3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 1748097, 1748097, 1748098, 1748098, 1748096, 1748097, ...

Resampling results across tuning parameters:

k	Accuracy	Kappa
5	0.6180742	0.4242890
7	0.6228041	0.4315363
9	0.6227871	0.4315548

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was  $k = 7$ .

Here, for  $k = 5$  the accuracy is 61.81% with Kappa statistic 0.4243 (Weak agreement).

For  $k = 7$  the accuracy is 62.28% with Kappa statistic 0.43154 (Weak agreement).

For  $k = 9$  the accuracy is 62.28% with Kappa statistic 0.4316 (Weak agreement).

Thus, the KNN classifier is better than a coin toss and it is performing weakly better compared to classifying simply by chance.

## 4.1.5 Phase I Summaries

We may summarize the accuracies and kappa values of each of the classifiers according to model validation method for ease of comparison.

The accuracies and Kappa statistic from Confusion Matrices are,

Table 05. Accuracy and Kappa statistic from Confusion Matrices

Method	Accuracy	Kappa Statistic
LDA	0.4613621	0.1740084
QDA	0.4460066	0.1840070
<b>Multinomial Logistic model</b>	<b>0.4678931</b>	<b>0.1779100</b>
KNN (K=5)	0.4314488	0.1185073
KNN (K=7)	0.4464054	0.1328893
KNN (K=9)	0.4546316	0.1427837

We can create a plot comparing the accuracies and Kappa statistic obtained from the Confusion Matrix method to illustrate the differences. The plots are provided below.

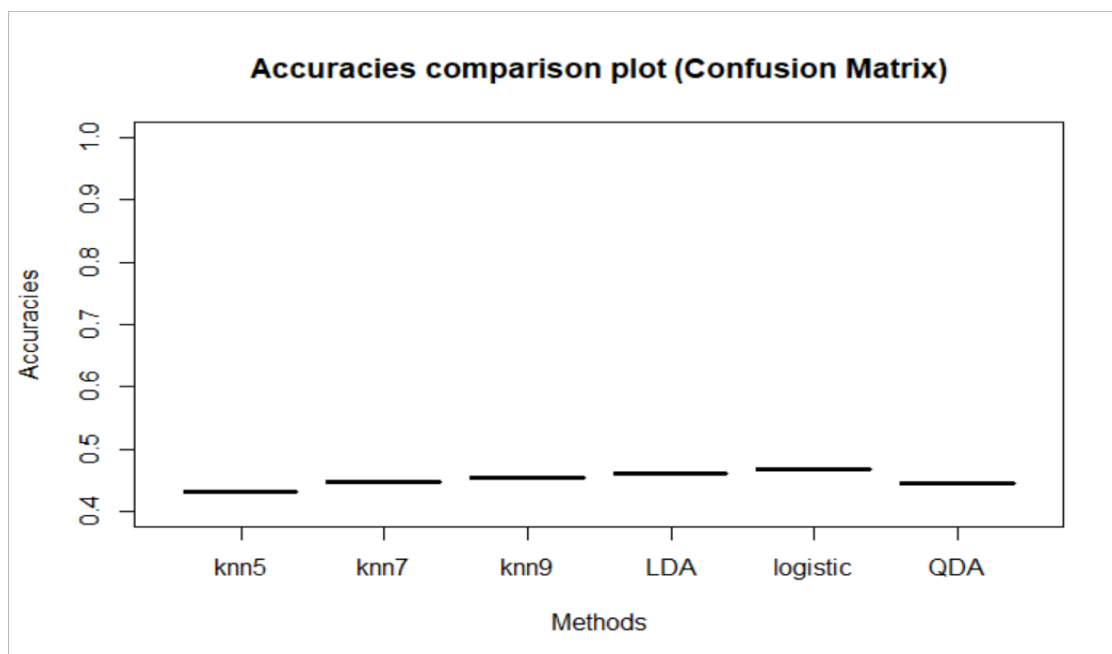


Figure 11. Plot comparing accuracies of the classifiers under Confusion Matrix.

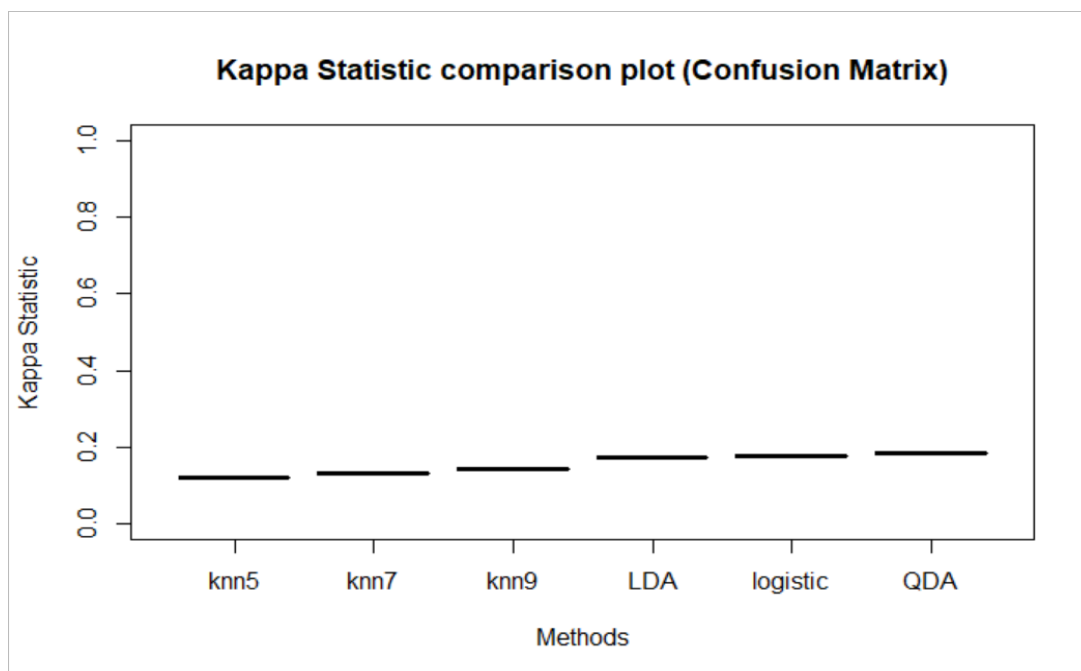


Figure 12. Plot comparing Kappa statistics of the classifiers under Confusion Matrix.

The accuracies and Kappa statistic from K Fold Cross-Validation are,

Table 06. Accuracy and Kappa statistic from K Fold Cross-Validation

Method	Accuracy	Kappa Statistic
LDA	0.5460607	0.3120672
QDA	0.5464195	0.3112956
Multinomial Logistic Model	0.5498788	0.3181662
KNN (K=5)	0.6180742	0.4242890
<b>KNN (K=7)</b>	<b>0.6228041</b>	<b>0.4315363</b>
KNN (K=9)	0.6227871	0.4315548

We can create a plot comparing the accuracies and Kappa statistic obtained from the K Fold Cross-Validation method to illustrate the differences. The plot is provided below.

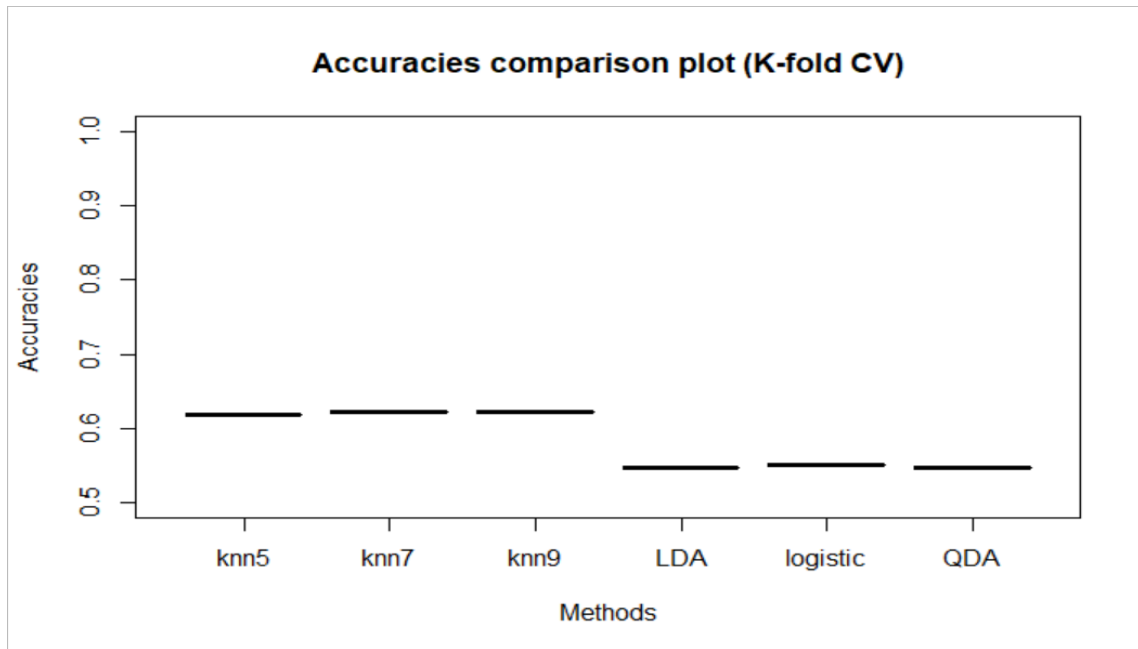


Figure 13. Plot comparing accuracies of the classifiers under K Fold Cross-Validation.

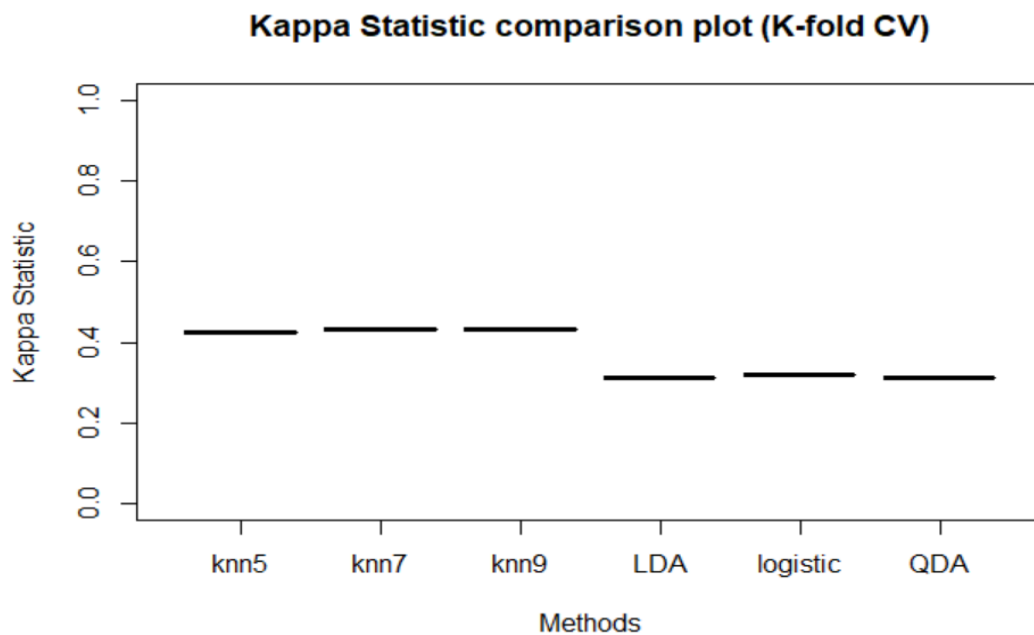


Figure 14. Plot comparing kappa statistics of the classifiers under K Fold Cross-Validation.

Therefore, according to confusion matrices, logistic regression classifier is performing the best in comparison, with the accuracy being 46.79% and Kappa statistic 0.1779 (No agreement).

Additionally, according to K-fold CV, KNN classifier with K=7 is performing the best compared to others, with the accuracy being 62.28% and Kappa statistic 0.43154 (Weak agreement).

However, none of the classifiers are performing on a utilitarian level. In the next phase of the analysis, we introduce a few selective variables and train the models for a smaller set of data with the hopes of improving the performances of the models.



We can summarize the confusion matrix statistics by class in a table for comparison.

Table 07. Summary table of the confusion Matrix statistics by each response class.

Statistic	Black Wins	Draw	White Wins
<b>Linear Discriminant Analysis</b>			
Sensitivity	0.6569	0.03633	0.6457
Specificity	0.6207	0.94774	0.6094
Pos Pred Value	0.3895	0.23579	0.5482
Neg Pred Value	0.8308	0.68903	0.7008
Prevalence	0.2692	0.30741	0.4234
Detection Rate	0.1768	0.01117	0.2734
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.6388	0.49203	0.6275
<b>Quadratic Discriminant Analysis</b>			
Sensitivity	0.6909	0.05856	0.6619
Specificity	0.6217	0.95920	0.6050
Pos Pred Value	0.3687	0.45684	0.5153
Neg Pred Value	0.8628	0.63486	0.7382
Prevalence	0.2423	0.36948	0.3882
Detection Rate	0.1674	0.02164	0.2570
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.6563	0.50888	0.6334
<b>Multinomial Logistic Regression</b>			
Sensitivity	0.6564	0.03453	0.6460
Specificity	0.6231	0.94724	0.6117
Pos Pred Value	0.3990	0.21579	0.5545
Neg Pred Value	0.8263	0.70002	0.6978
Prevalence	0.2760	0.29599	0.4280
Detection Rate	0.1812	0.01022	0.2765
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.6398	0.49089	0.6289
<b>K Nearest Neighbour (K=5)</b>			
Sensitivity	0.5653	0.06028	0.5932
Specificity	0.5975	0.95784	0.5636
Pos Pred Value	0.3939	0.36526	0.4720
Neg Pred Value	0.7482	0.71692	0.6781
Prevalence	0.3163	0.28697	0.3967
Detection Rate	0.1788	0.01730	0.2353
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.5814	0.50906	0.5784

<b>K Nearest Neighbour (K=7)</b>			
Sensitivity	0.5752	0.06340	0.6008
Specificity	0.6031	0.95863	0.5718
Pos Pred Value	0.4057	0.36421	0.4913
Neg Pred Value	0.7508	0.73252	0.6754
Prevalence	0.3203	0.27206	0.4077
Detection Rate	0.1842	0.01725	0.2449
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.5891	0.51102	0.5863
<b>K Nearest Neighbour (K=9)</b>			
Sensitivity	0.5844	0.06196	0.6081
Specificity	0.6074	0.95796	0.5783
Pos Pred Value	0.4122	0.34947	0.5032
Neg Pred Value	0.7562	0.73697	0.6775
Prevalence	0.3203	0.26713	0.4126
Detection Rate	0.1872	0.01655	0.2509
Detection Prevalence	0.4540	0.04736	0.4986
Balanced Accuracy	0.5959	0.50996	0.5932

As we can observe here, the Draw class generally has lower Sensitivity (True Positive rate) and Positive Prediction Value (Precision), but it has high Specificity and Negative Prediction Value. This implies that the classifiers are predicting the non-occurrence of draws accurately, but predicting the actual occurrences of draws poorly. Additionally we observe comparatively lower balanced accuracy for Draws, compared to Black or White Winning.

As discussed before, this may be due to imbalanced number of cases for drawn games in the testing dataset. We can check whether balancing the observations to have equal proportions leads to improvement in performances. We create a test set with equal proportions of games where Black wins, White wins and Draws. The

smaller test dataset has 2850 observations, with 950 games with each of the response categories.

We can validate this new testing dataset against the trained classifier with the poorest accuracy performance under confusion matrix in phase I and obtain the statistics. Then we can check whether balancing the data led to improved performance.

The poorest performing classifier in this phase I was K Nearest Neighbour (K=5).

Table 08. Comparison of accuracy performance for imbalanced and balanced dataset for KNN (K=5)

	<b>Old testing dataset</b>			<b>New testing dataset</b>		
	Accuracy: 0.4314 ; Kappa: 0.1185			Accuracy: 0.4056 ; Kappa: 0.1084		
<b>Statistics by class</b>	<b>Black</b>	<b>Draw</b>	<b>White</b>	<b>Black</b>	<b>Draw</b>	<b>White</b>
Sensitivity	0.5653	0.06028	0.5932	0.4271	0.3593	0.4253
Specificity	0.5975	0.95784	0.5636	0.7099	0.6781	0.7225
Pos Pred Value	0.3939	0.36526	0.4720	0.4042	0.3305	0.4821
Neg Pred Value	0.7482	0.71692	0.6781	0.7289	0.7053	0.6742
Prevalence	0.3163	0.28697	0.3967	0.3154	0.3067	0.3779
Detection Rate	0.1788	0.01730	0.2353	0.1347	0.1102	0.1607
Detection Prevalence	0.4540	0.04736	0.4986	0.3333	0.3333	0.3333
Balanced Accuracy	0.5814	0.50906	0.5784	0.5685	0.5187	0.5739

Here we see that, for the balanced dataset the overall accuracy and kappa value decreases. However, there is significant improvement in the sensitivity and detection rate statistic for the response category Draw. It implies the percentage of

correct predictions of draw occurrences improves. The balanced accuracy also improves slightly for draws.

However, the improvement are not quite large and we do not gain in terms of accuracy and kappa. Therefore, a balanced testing dataset did not lead to improved performance for the worst performing classifier. So, we may conclude that balancing may not lead to sufficient improvements in accuracy in our case.

However, the size of the balanced dataset is quite small. We can do further exploration of this phenomena by obtaining a larger balanced dataset. A larger balanced dataset may provide a cleared picture.

## 4.2 Phase II

The smaller pool data that we randomly sampled from the OTB games to obtain several new attributes, we can obtain the summaries and plot. This dataset contains 212 observations.

The summaries are,

result	welo	belo	length	w.inaccuracy	w.mistakes
black:42	Min. :2794	Min. :2018	Min. : 21.00	Min. :0.000	Min. :0.0000
draw :82	1st Qu.:2827	1st Qu.:2620	1st Qu.: 61.00	1st Qu.:1.000	1st Qu.:0.0000
white:88	Median :2838	Median :2698	Median : 78.00	Median :2.000	Median :0.0000
	Mean :2834	Mean :2651	Mean : 81.13	Mean :1.939	Mean :0.8019
	3rd Qu.:2849	3rd Qu.:2744	3rd Qu.: 98.00	3rd Qu.:3.000	3rd Qu.:1.0000
	Max. :2851	Max. :2838	Max. :168.00	Max. :7.000	Max. :7.0000
w.blunders	w.ac1	b.inaccuracy	b.mistakes	b.blunders	b.ac1
Min. :0.0000	Min. : 4.0	Min. :0.000	Min. :0.0000	Min. :0.000	Min. : 2.00
1st Qu.:0.0000	1st Qu.:11.0	1st Qu.:1.000	1st Qu.:0.0000	1st Qu.:0.000	1st Qu.:12.00
Median :0.0000	Median :18.0	Median :2.000	Median :0.0000	Median :0.000	Median :23.50
Mean :0.6604	Mean :21.6	Mean :2.108	Mean :0.7311	Mean :0.816	Mean :26.08
3rd Qu.:1.0000	3rd Qu.:30.0	3rd Qu.:3.000	3rd Qu.:1.0000	3rd Qu.:1.000	3rd Qu.:36.00
Max. :6.0000	Max. :75.0	Max. :9.000	Max. :5.0000	Max. :6.000	Max. :84.00

Figure 15. Summary table of the small pool of games attributes.

The bar plot summarizing the frequencies of the game results are,

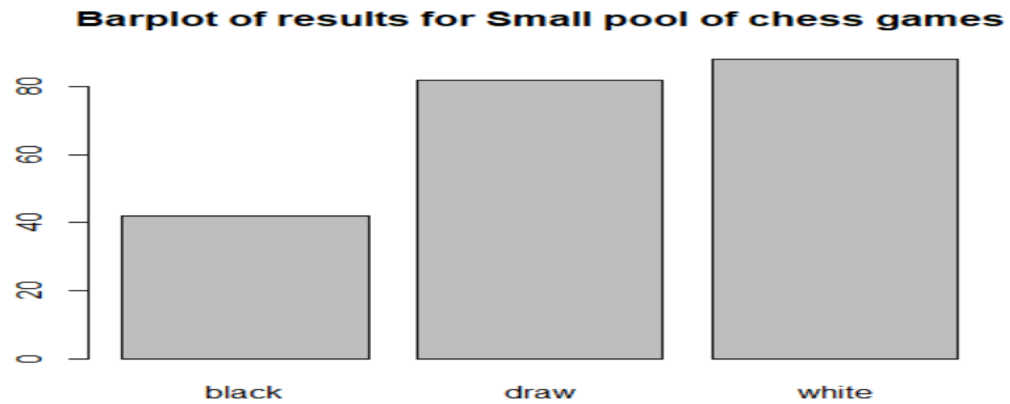


Figure 16. Bar plot of the results of the small scale chess games.

We can look at the frequencies and proportions on the table below,

Table 09. Frequency and proportions of result responses for small pool of games.

Result level	Black wins	Draw	White wins
Frequency	42	82	88
Proportion	0.1981132	0.3867925	0.4150943

The histograms of the attributes is provided below,

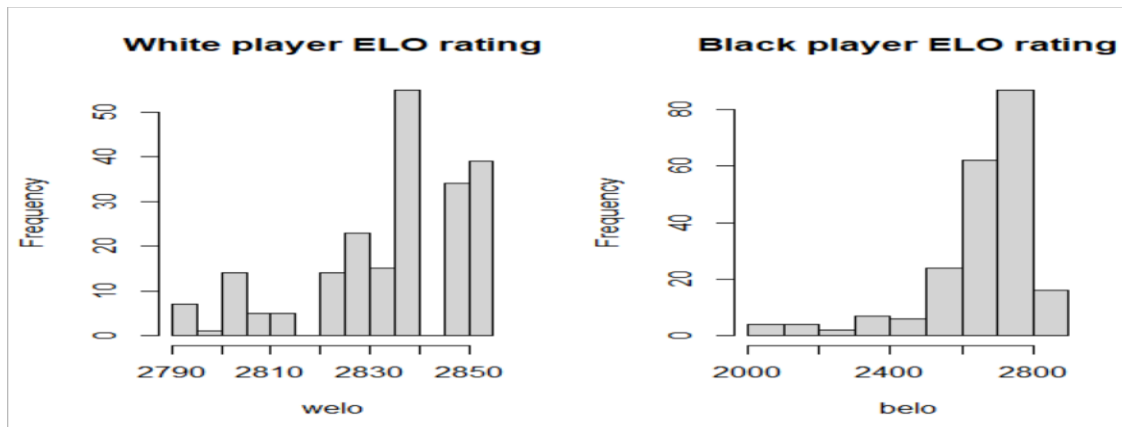


Figure 17. Histograms of white and black players ELO ratings in the small pool of games.

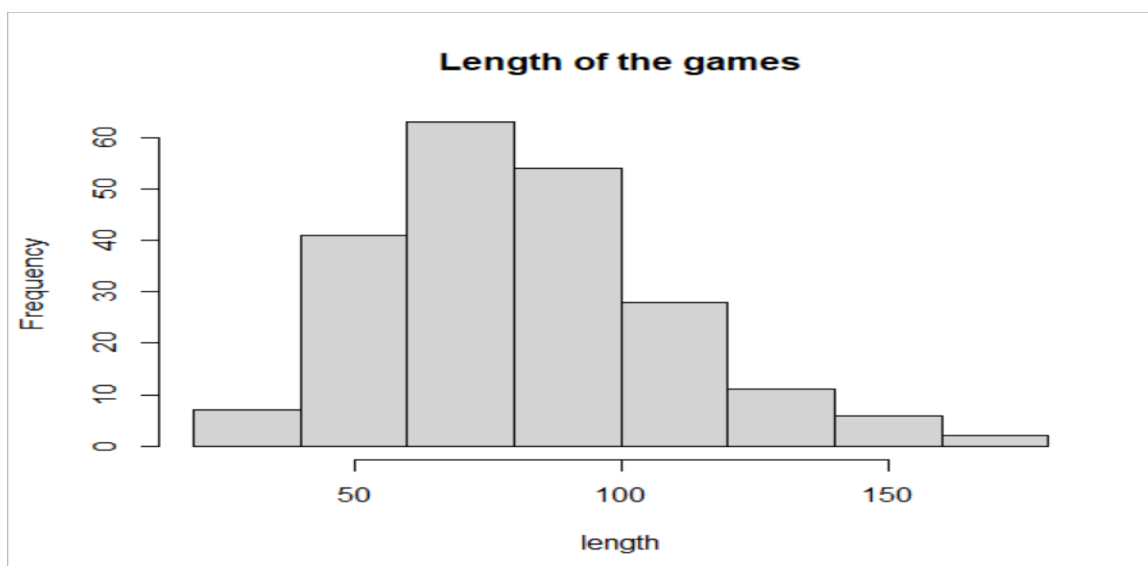


Figure 18. Histograms of the number of turns or length of the small pool of games.

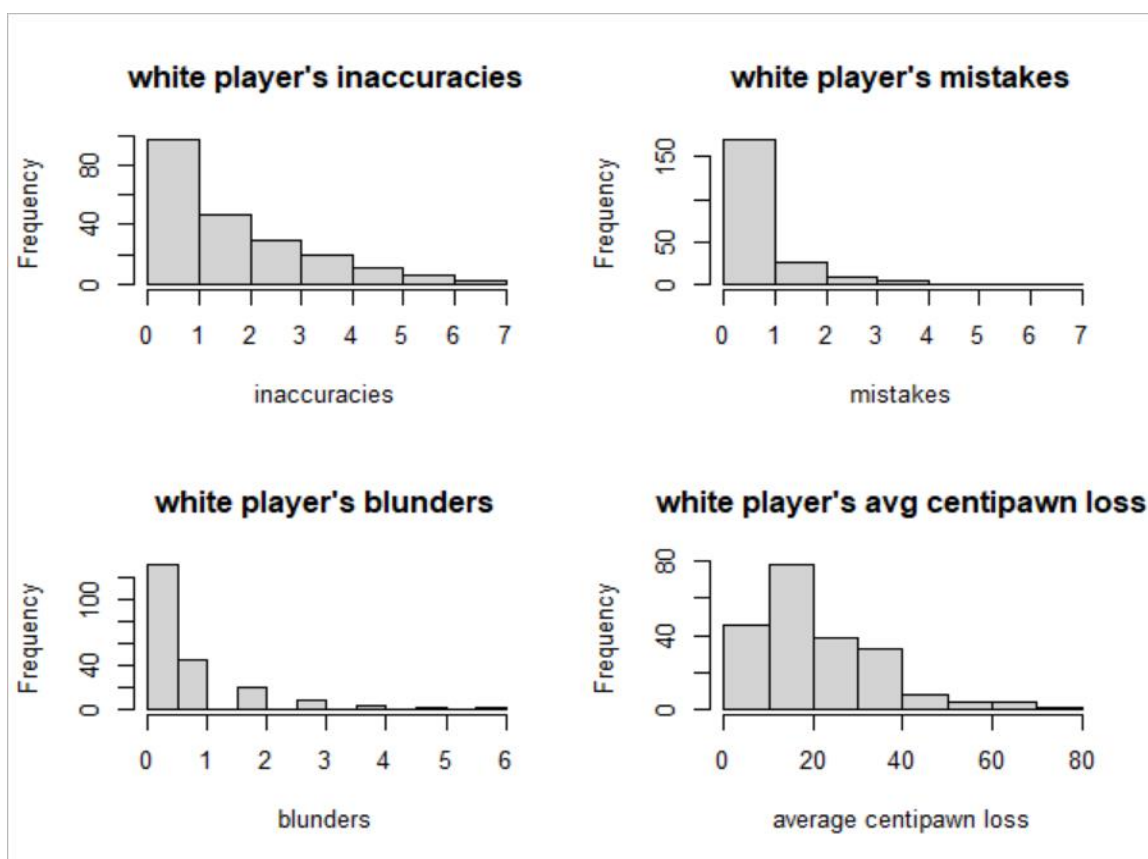


Figure 19. Histograms of the White player's inaccuracies, mistakes, blunders, and ACL.

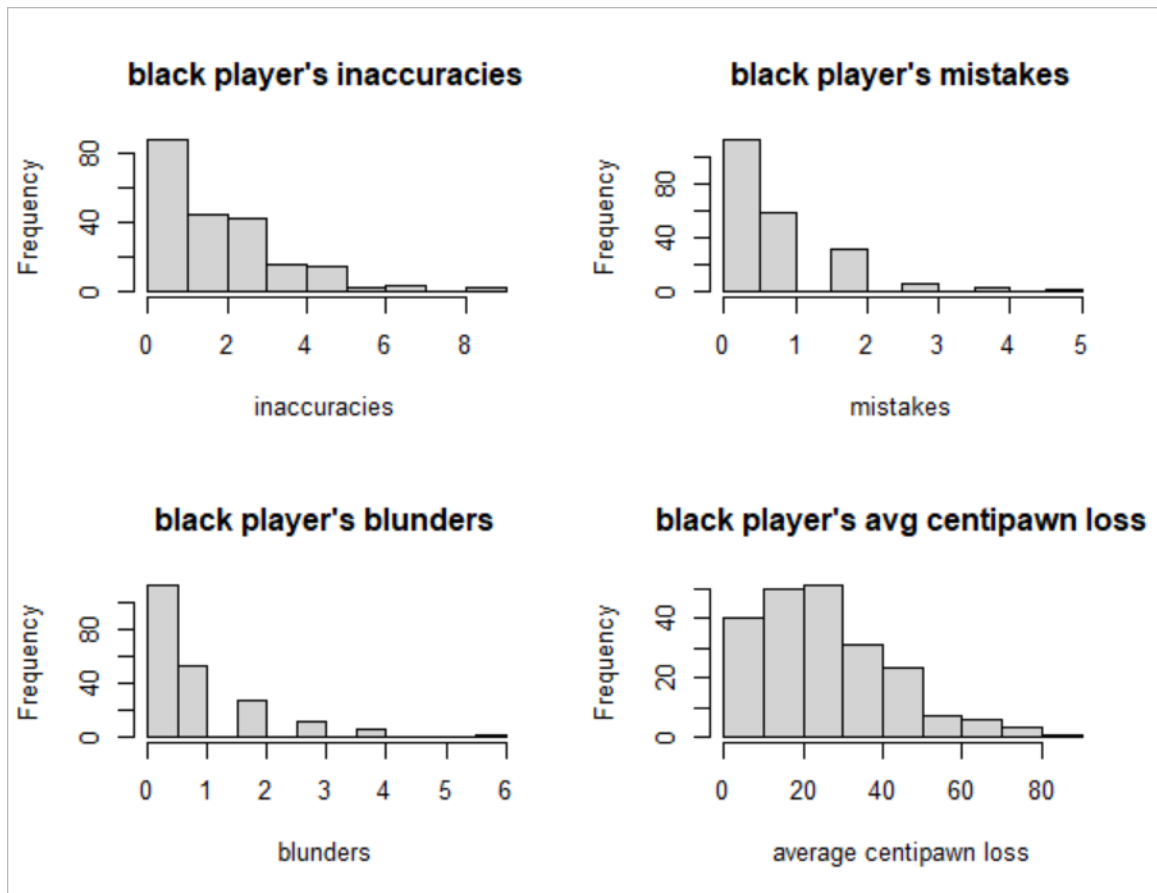


Figure 20. Histograms of the Black player's inaccuracies, mistakes, blunders, and ACL.

We split this small pool of games data into 2 splits, with 148 observations as the training dataset (70% of total) and 64 observations as the testing dataset.

We train the model based on *welo*, *belo*, and *length* as the predictors and validate using testing dataset. This is the baseline case of modelling using the 3 variables.

And next we use 11 predictors (*welo*, *belo*, *length*, *w.inaccuracy*, *w.mistakes*, *w.blunders*, *w.acl*, *b.inaccuracy*, *b.mistakes*, *b.blunders*, and *b.acl*) as predictors and

validate using testing dataset. This is the comparison case of modelling using the 11 variables.

## 4.2.1 Linear Discriminant Analysis

Using Linear Discriminant Analysis algorithm let us observe the results when using Confusion Matrix as the method for model with 3 predictors and 11 predictors. The confusion matrices are,

Table 10. LDA classifier Confusion Matrices for 3 predictors and 11 predictors.

<b>3 predictor Confusion Matrix</b>				<b>11 predictor Confusion Matrix</b>			
Reference				Reference			
Prediction	black	draw	white	Prediction	black	draw	white
black	7	1	1	black	8	1	0
draw	0	15	6	draw	1	19	1
white	0	19	15	white	0	3	31

For the sake of brevity, we can summarize the statistics in a table for easy comparison.

Table 11. Comparison table of Confusion Matrix statistics for 3 predictors vs. 11 predictors models.

	<b>3 predictors</b>				<b>11 predictors</b>		
	Accuracy: 0.5781 ; Kappa: 0.3224				Accuracy: 0.9062; Kappa: 0.8429		
<b>Statistics by class</b>	<b>Black</b>	<b>Draw</b>	<b>White</b>		<b>Black</b>	<b>Draw</b>	<b>White</b>
Sensitivity	1.0000	0.4286	0.6818		0.8889	0.8261	0.9688
Specificity	0.9649	0.7931	0.5476		0.9818	0.9512	0.9062
Pos Pred Value	0.7778	0.7143	0.4412		0.8889	0.9048	0.9118
Neg Pred Value	1.0000	0.5349	0.7667		0.9818	0.9070	0.9667
Prevalence	0.1094	0.5469	0.3438		0.1406	0.3594	0.5000



Detection Rate	0.1094	0.2344	0.2344		0.1250	0.2969	0.4844
Detection Prevalence	0.1406	0.3281	0.5312		0.1406	0.3281	0.5312
Balanced Accuracy	0.9825	0.6108	0.6147		0.9354	0.8887	0.9375

Here, we can see that the overall accuracy increases from 57.81% to 90.62% and the Kappa statistic increases from 0.3224 (weak agreement) to 0.8429 (strong agreement). We also observe improvements in the confusion matrix statistics, implying the model is generalizing better by each response class with 11 predictors compared to 3 predictors.

Now, Let us observe the results from K Fold Cross-Validation for model with 3 predictors (welo, belo, and length),

Linear Discriminant Analysis

```
212 samples
 3 predictor
 3 classes: 'black', 'draw', 'white'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 133, 134, 133, 133, 134, 133, ...
Resampling results:
```

```
Accuracy    Kappa
0.6819048   0.4999835
```

Here the accuracy is 68.19% with Kappa statistic 0.5 (Weak agreement).

Thus the LDA classifier is 68% accurate and it is performing slightly better compared to classifying simply by chance.

Now, Let us observe the results from K Fold Cross-Validation for model with 11 predictors (welo, belo, length, w.inaccuracy, w.mistakes, w.blunders, w.acl, b.inaccuracy, b.mistakes, b.blunders, and b.acl),

```
Linear Discriminant Analysis
```

```
212 samples
 11 predictor
   3 classes: 'black', 'draw', 'white'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 133, 134, 134, 133, 134, 133, ...
Resampling results:
```

```
Accuracy  Kappa
0.889881  0.8263807
```

Here, the accuracy is 88.99% with Kappa statistic 0.8264 (Strong agreement).

Thus, the LDA classifier is approximately 89% accurate and it is performing strongly compared to classifying simply by chance.

Therefore, we see an increase in overall accuracy and Kappa statistic increase under K Fold Cross-Validation method as well. The accuracy increases from 68.19% to 88.99% and the Kappa increases from 0.5 (weak agreement) to 0.8264 (strong agreement).

Now, Let us observe the results from Leave-One-Out Cross-Validation for model with 3 predictors (welo, belo, and length),

### Linear Discriminant Analysis

```
212 samples
  3 predictor
  3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Leave-One-Out Cross-Validation

Summary of sample sizes: 147, 147, 147, 147, 147, 147, ...

Resampling results:

Accuracy	Kappa
0.6486486	0.444292

Here the accuracy is 64.86% with Kappa statistic 0.4443 (weak agreement).

Thus, the LDA classifier is approximately 65% accurate and it is performing weakly compared to classifying simply by chance.

Now, Let us observe the results from Leave-One-Out Cross-Validation for model

with 11 predictors (welo, belo, length, w.inaccuracy, w.mistakes, w.blunders, w.acl, b.inaccuracy, b.mistakes, b.blunders, and b.acl)

### Linear Discriminant Analysis

```
212 samples
 11 predictor
  3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Leave-One-Out Cross-Validation

Summary of sample sizes: 147, 147, 147, 147, 147, 147, ...

Resampling results:

Accuracy	Kappa
0.8783784	0.8092784

Here, the accuracy is 87.84% with Kappa statistic 0.8093 (Strong agreement).

Thus, the LDA classifier is approximately 88% accurate and it is performing strongly compared to classifying simply by chance.

Therefore, we see an increase in overall accuracy and kappa statistic increase under Leave One Out Cross-Validation method as well. The accuracy increases from 68.19% to 88.99% and the Kappa increases from 0.4445 (weak agreement) to 0.8093 (strong agreement).

Thus, we see that the accuracy performance of the LDA classifier improves with addition of the engine evaluation variables, over all model validation methods.

## 4.2.2 Quadratic Discriminant Analysis

Using Quadratic Discriminant Analysis algorithm let us observe the results when using Confusion Matrix as the method for model with 3 predictors and 11 predictors. The confusion matrices are,

Table 12. QDA classifier Confusion Matrices for 3 predictors and 11 predictors.

<b>3 predictor Confusion Matrix</b>					<b>11 predictor Confusion Matrix</b>				
Reference					Reference				
Prediction	black	draw	white		Prediction	black	draw	white	
black	7	1	1		black	9	0	0	
draw	0	16	5		draw	3	17	1	
white	0	15	19		white	1	3	30	

To make it concise, we can summarize the statistics for both models in a table for easy comparison.

Table 13. Comparison table of Confusion Matrix statistics for 3 predictors vs. 11 predictors models.

	<b>3 predictors</b>				<b>11 predictors</b>		
	Accuracy: 0.6562; Kappa: 0.4393				Accuracy: 0.8750 ; Kappa: 0.7956		
<b>Statistics by class</b>	<b>Black</b>	<b>Draw</b>	<b>White</b>		<b>Black</b>	<b>Draw</b>	<b>White</b>
Sensitivity	1.0000	0.5000	0.7600		0.6923	0.8500	0.9677
Specificity	0.9649	0.8438	0.6154		1.0000	0.9091	0.8788
Pos Pred Value	0.7778	0.7619	0.5588		1.0000	0.8095	0.8824
Neg Pred Value	1.0000	0.6279	0.8000		0.9273	0.9302	0.9667
Prevalence	0.1094	0.5000	0.3906		0.2031	0.3125	0.4844
Detection Rate	0.1094	0.2500	0.2969		0.1406	0.2656	0.4688
Detection Prevalence	0.1406	0.3281	0.5312		0.1406	0.3281	0.5312
Balanced Accuracy	0.9825	0.6719	0.6877		0.8462	0.8795	0.9233

Here, we can see that the overall accuracy increases from 65.62% to 87.5% and the kappa statistic increases from 0.4393 (weak agreement) to 0.7956 (moderate agreement). We also observe improvements in the confusion matrix statistics, implying the model is generalizing better by each response class with 11 predictors compared to 3 predictors.

Now, Let us observe the results from K Fold Cross-Validation for model with 3 predictors (welo, belo, and length),

Quadratic Discriminant Analysis

```
212 samples
 3 predictor
 3 classes: 'black', 'draw', 'white'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold)
```

Summary of sample sizes: 132, 133, 133, 134, 134, 132, ...  
 Resampling results:

Accuracy	Kappa
0.6066071	0.3757971

Here, the accuracy is 60.66% with Kappa statistic 0.3758 (Minimal agreement).

Thus, the QDA classifier is 68% accurate and it is performing slightly better compared to classifying simply by chance.

Now, Let us observe the results from K Fold Cross-Validation for model with 11 predictors (welo, belo, length, w.inaccuracy, w.mistakes, w.blunders, w.acl, b.inaccuracy, b.mistakes, b.blunders, and b.acl),

Quadratic Discriminant Analysis

212 samples  
 11 predictor  
 3 classes: 'black', 'draw', 'white'

No pre-processing  
 Resampling: Cross-Validated (10 fold)  
 Summary of sample sizes: 134, 133, 133, 133, 133, 134, ...  
 Resampling results:

Accuracy	Kappa
0.9061905	0.8560987

Here, the accuracy is 90.62% with Kappa statistic 0.8560 (Strong agreement).

Thus, the QDA classifier is approximately 91% accurate and it is performing strongly compared to classifying simply by chance.

Therefore, we see an increase in overall accuracy and kappa statistic increase under K Fold Cross-Validation method as well. The accuracy increases from 60.66% to

90.62% and the kappa increases from 0.3758 (minimal agreement) to 0.856 (strong agreement).

Now, Let us observe the results from Leave-One-Out Cross-Validation for model with 3 predictors (welo, belo, and length),

```
Quadratic Discriminant Analysis
```

```
212 samples
  3 predictor
  3 classes: 'black', 'draw', 'white'
```

```
No pre-processing
```

```
Resampling: Leave-One-Out Cross-Validation
```

```
Summary of sample sizes: 147, 147, 147, 147, 147, 147, ...
```

```
Resampling results:
```

Accuracy	Kappa
0.6013514	0.3717082

Here, the accuracy is 60.14% with Kappa statistic 0.3717 (weak agreement).

Thus, the QDA classifier is approximately 60% accurate and it is performing weakly compared to classifying simply by chance.

Now, Let us observe the results from Leave-One-Out Cross-Validation for model with 11 predictors (welo, belo, length, w.inaccuracy, w.mistakes, w.blunders, w.acl, b.inaccuracy, b.mistakes, b.blunders, and b.acl)

```
Quadratic Discriminant Analysis
```

```
212 samples
 11 predictor
  3 classes: 'black', 'draw', 'white'
```

```
No pre-processing
```

```
Resampling: Leave-One-Out Cross-Validation
```

```
Summary of sample sizes: 147, 147, 147, 147, 147, 147, ...
```

Resampling results:

```
Accuracy   Kappa
0.9121622  0.8650298
```

Here, the accuracy is 91.22% with Kappa statistic 0.8650 (Strong agreement).

Thus, the QDA classifier is approximately 91% accurate and it is performing strongly compared to classifying simply by chance.

Therefore, we see an increase in overall accuracy and kappa statistic increase under Leave One Out Cross-Validation method as well. The accuracy increases from 60.14% to 91.22% and the kappa increases from 0.3717 (weak agreement) to 0.8650 (strong agreement).

Thus, we see that the accuracy performance of the QDA classifier improves with addition of the engine evaluation variables, over all model validation methods.

## 4.2.3 Multinomial Logistic Regression Classifier

Using Multinomial Logistic Regression Classifier let us observe the results when using Confusion Matrix as the method for model with 3 predictors and 11 predictors. The confusion matrices are,

Table 14. Multinomial Logistic classifier Confusion Matrices for 3 predictors and 11 predictors.

<b>3 predictor Confusion Matrix</b>					<b>11 predictor Confusion Matrix</b>				
Reference					Reference				
Prediction	black	draw	white		Prediction	black	draw	white	
black	7	1	1		black	8	1	0	
draw	0	16	5		draw	2	18	1	
white	0	15	19		white	0	4	30	



For the sake of brevity, we can summarize the statistics for both models in a table for easy comparison.

Table 15. Comparison table of Confusion Matrix statistics for 3 predictors vs. 11 predictors models.

	<b>3 predictors</b>				<b>11 predictors</b>		
	Accuracy: 0.5469; Kappa:0.2647				Accuracy: 0.8750 ; Kappa: 0.7956		
<b>Statistics by class</b>	<b>Black</b>	<b>Draw</b>	<b>White</b>		<b>Black</b>	<b>Draw</b>	<b>White</b>
Sensitivity	1.0000	0.3939	0.6250		0.8000	0.7826	0.9677
Specificity	0.9649	0.7419	0.5250		0.9815	0.9268	0.8788
Pos Pred Value	0.7778	0.6190	0.4412		0.8889	0.8571	0.8824
Neg Pred Value	1.0000	0.5349	0.7000		0.9636	0.8837	0.9667
Prevalence	0.1094	0.5156	0.3750		0.1562	0.3594	0.4844
Detection Rate	0.1094	0.2031	0.2344		0.1250	0.2812	0.4688
Detection Prevalence	0.1406	0.3281	0.5312		0.1406	0.3281	0.5312
Balanced Accuracy	0.9825	0.5679	0.5750		0.8907	0.8547	0.9233

Here, we can see that the overall accuracy increases from 54.69% to 87.51% and the Kappa statistic increases from 0.2647 (minimal agreement) to 0.7956 (moderate agreement). We also observe improvements in the confusion matrix statistics, implying the model is generalizing better by each response class with 11 predictors compared to 3 predictors.

Now, Let us observe the results from K Fold Cross-Validation for model with 3 predictors (welo, belo, and length),

Penalized Multinomial Regression

212 samples

```
3 predictor
3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 134, 134, 133, 134, 133, 132, ...

Resampling results across tuning parameters:

decay	Accuracy	Kappa
0e+00	0.6741071	0.4831433
1e-04	0.6741071	0.4831433
1e-01	0.6598214	0.4602092

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was decay = 1e-04.

Since we are using Multinomial Logistic Regression, we only consider the metrics for decay 0. Here the accuracy is 67.41% with Kappa statistic 0.4831 (Weak agreement).

Thus, the Multinomial Logistic Classifier is 68% accurate and it is performing weakly compared to classifying simply by chance.

Now, Let us observe the results from K Fold Cross-Validation for model with 11 predictors (welo, belo, length, w.inaccuracy, w.mistakes, w.blunders, w.acl, b.inaccuracy, b.mistakes, b.blunders, and b.acl),

Penalized Multinomial Regression

```
212 samples
11 predictor
3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 133, 133, 132, 133, 134, 134, ...

Resampling results across tuning parameters:

decay	Accuracy	Kappa
0e+00	0.8909524	0.8288350
1e-04	0.8985119	0.8424572
1e-01	0.8917857	0.8286922

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was `decay = 1e-04`.

Since we are using Multinomial Logistic Regression, we only consider the metrics for decay 0. Here the accuracy is 89.09% with Kappa statistic 0.8288 (Strong agreement).

Thus, the Multinomial Logistic Regression classifier is approximately 91% accurate and it is performing strongly compared to classifying simply by chance.

Therefore, we see an increase in overall accuracy and kappa statistic increase under K Fold Cross-Validation method as well. The accuracy increases from 68.19% to 89.09% and the kappa increases from 0.4831 (weak agreement) to 0.8288 (strong agreement).

Now, let us observe the results from Leave-One-Out Cross-Validation for model with 3 predictors (welo, belo, and length),

Penalized Multinomial Regression

```
212 samples
 3 predictor
 3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Leave-One-Out Cross-Validation

Summary of sample sizes: 147, 147, 147, 147, 147, 147, ...

Resampling results across tuning parameters:

decay	Accuracy	Kappa
0e+00	0.6689189	0.4792475
1e-04	0.6689189	0.4792475
1e-01	0.6689189	0.4792475

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was decay = 0.1.

Since we are using Multinomial Logistic Regression, we only consider the metrics for decay 0. Here the accuracy is 66.89% with Kappa statistic 0.4792 (weak agreement).

Thus, the Multinomial Logistic Classifier is approximately 67% accurate and it is performing weakly compared to classifying simply by chance.

Now, let us observe the results from Leave-One-Out Cross-Validation for model with 11 predictors (welo, belo, length, w.inaccuracy, w.mistakes, w.blunders, w.acl, b.inaccuracy, b.mistakes, b.blunders, and b.acl)

Penalized Multinomial Regression

```
212 samples
11 predictor
3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Leave-One-Out Cross-Validation

Summary of sample sizes: 147, 147, 147, 147, 147, 147, ...

Resampling results across tuning parameters:

decay	Accuracy	Kappa
0e+00	0.8851351	0.8217499
1e-04	0.8851351	0.8228044
1e-01	0.8918919	0.8320686

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was decay = 0.1.

Since we are using Multinomial Logistic Regression, we only consider the metrics for decay 0. Here, the accuracy is 88.51% with Kappa statistic 0.8217 (Strong agreement).

Thus, the Multinomial Logistic classifier is approximately 89% accurate and it is performing strongly compared to classifying simply by chance.

Therefore, we see an increase in overall accuracy and kappa statistic increase under Leave One Out Cross-Validation method as well. The accuracy increases from 66.89% to 88.51% and the kappa increases from 0.4792 (weak agreement) to 0.8217 (strong agreement).

Thus, we see that the accuracy performance of the Multinomial Logistic Regression classifier improves with addition of the engine evaluation variables, over all model validation methods.

## 4.2.4 K Nearest Neighbour

Using K Nearest Neighbour Classifier let us observe the results when using Confusion Matrix as the method for model with 3 predictors and 11 predictors. We shall develop this classifier using K=5, K=7, and K=9.

**For K = 5**, we obtain the confusion matrices,

Table 16. KNN (K=5) classifier Confusion Matrices for 3 predictors and 11 predictors.

<b>3 predictor Confusion Matrix</b>					<b>11 predictor Confusion Matrix</b>				
Reference					Reference				
Prediction	black	draw	white		Prediction	black	draw	white	
black	6	2	1		black	7	0	2	
draw	0	16	5		draw	1	17	3	
white	0	8	26		white	1	2	31	

We can summarize the statistics in a table for easy comparison.

Table 17. Comparison table of Confusion Matrix statistics for 3 predictors vs. 11 predictors models.

	<b>3 predictors</b>				<b>11 predictors</b>		
	Accuracy: 0.75 ; Kappa: 0.5748				Accuracy: 0.8594 ; Kappa: 0.7592		
<b>Statistics by class</b>	<b>Black</b>	<b>Draw</b>	<b>White</b>		<b>Black</b>	<b>Draw</b>	<b>White</b>
Sensitivity	1.0000	0.6154	0.8125		0.7778	0.8947	0.8611
Specificity	0.94828	0.8684	0.7500		0.9636	0.9111	0.8929
Pos Pred Value	0.66667	0.7619	0.7647		0.7778	0.8095	0.9118
Neg Pred Value	1.0000	0.7674	0.8000		0.9636	0.9535	0.8333
Prevalence	0.09375	0.4062	0.5000		0.1406	0.2969	0.5625
Detection Rate	0.09375	0.2500	0.4062		0.1094	0.2656	0.4844
Detection Prevalence	0.1406	0.3281	0.5312		0.1406	0.3281	0.5312
Balanced Accuracy	0.97414	0.7419	0.7812		0.8707	0.9029	0.8770

Here, we can see that the overall accuracy increases from 75% to 85.94% and the Kappa statistic increases from 0.5748 (weak agreement) to 0.7592 (moderate agreement). We also observe improvements in the confusion matrix statistics,

implying the model is generalizing better by each response class with 11 predictors compared to 3 predictors.

**For K = 7**, we obtain the confusion matrices,

Table 18. KNN (K=7) classifier Confusion Matrices for 3 predictors and 11 predictors.

<b>3 predictor Confusion Matrix</b>					<b>11 predictor Confusion Matrix</b>				
Reference					Reference				
Prediction	black	draw	white		Prediction	black	draw	white	
black	6	2	1		black	7	1	1	
draw	0	15	6		draw	2	16	3	
white	0	9	25		white	1	3	30	

We can summarize the statistics in a table for easy comparison.

Table 18. Comparison table of Confusion Matrix statistics for 3 predictors vs. 11 predictors models.

	<b>3 predictors</b>				<b>11 predictors</b>		
	Accuracy: 0.7188; Kappa: 0.5216				Accuracy: 0.8281; Kappa: 0.7103		
<b>Statistics by class</b>	<b>Black</b>	<b>Draw</b>	<b>White</b>		<b>Black</b>	<b>Draw</b>	<b>White</b>
Sensitivity	1.0000	0.5769	0.7812		0.7000	0.8000	0.8824
Specificity	0.94828	0.8421	0.7188		0.9630	0.8864	0.8667
Pos Pred Value	0.66667	0.7143	0.7353		0.7778	0.7619	0.8824
Neg Pred Value	1.0000	0.7442	0.7667		0.9455	0.9070	0.8667
Prevalence	0.09375	0.4062	0.5000		0.1562	0.3125	0.5312
Detection Rate	0.09375	0.2344	0.3906		0.1094	0.2500	0.4688
Detection Prevalence	0.1406	0.3281	0.5312		0.1406	0.3281	0.5312
Balanced Accuracy	0.97414	0.7095	0.7500		0.8315	0.8432	0.8745

Here, we can see that the overall accuracy increases from 71.88% to 82.81% and the Kappa statistic increases from 0.5216 (weak agreement) to 0.7103 (moderate agreement). We also observe improvements in the confusion matrix statistics, implying the model is generalizing better by each response class with 11 predictors compared to 3 predictors.

**For K = 9**, we obtain the confusion matrices,

Table 19. KNN (K=9) classifier Confusion Matrices for 3 predictors and 11 predictors.

<b>3 predictor Confusion Matrix</b>				<b>11 predictor Confusion Matrix</b>			
Reference				Reference			
Prediction	black	draw	white	Prediction	black	draw	white
black	6	1	2	black	7	0	2
draw	0	13	8	draw	1	17	3
white	0	11	23	white	0	3	31

We can summarize the statistics in a table for easy comparison.

Table 20. Comparison table of Confusion Matrix statistics for 3 predictors vs. 11 predictors models.

	<b>3 predictors</b>			<b>11 predictors</b>		
	Accuracy:0.6562; Kappa: 0.4121			Accuracy: 0.8594 ; Kappa: 0.758		
Statistics by class	<b>Black</b>	<b>Draw</b>	<b>White</b>	<b>Black</b>	<b>Draw</b>	<b>White</b>
Sensitivity	1.0000	0.5200	0.6970	0.8750	0.8500	0.8611
Specificity	0.94828	0.7949	0.6452	0.9643	0.9091	0.8929
Pos Pred Value	0.66667	0.6190	0.6765	0.7778	0.8095	0.9118
Neg Pred Value	1.0000	0.7209	0.6667	0.9818	0.9302	0.8333
Prevalence	0.09375	0.3906	0.5156	0.1250	0.3125	0.5625
Detection Rate	0.09375	0.2031	0.3594	0.1094	0.2656	0.4844
Detection Prevalence	0.1406	0.3281	0.5312	0.1406	0.3281	0.5312
Balanced Accuracy	0.97414	0.6574	0.6711	0.9196	0.8795	0.8770



Here, we can see that the overall accuracy increases from 65.62% to 85.94% and the Kappa statistic increases from 0.41.21 (weak agreement) to 0.758 (moderate agreement). We also observe improvements in the confusion matrix statistics, implying the model is generalizing better by each response class with 11 predictors compared to 3 predictors.

Now, Let us observe the results from K Fold Cross-Validation for model with 3 predictors (welo, belo, and length),

k-Nearest Neighbors

```
212 samples
  3 predictor
  3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 132, 132, 134, 133, 133, 133, ...

Resampling results across tuning parameters:

k	Accuracy	Kappa
5	0.6630952	0.4734862
7	0.6777976	0.4951164
9	0.6911905	0.5162781

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was k = 9.

Here, for k = 5 the accuracy is 66.30% with Kappa statistic 0.4735 (Weak agreement).

For k = 7 the accuracy is 67.78% with Kappa statistic 0.4951 (Weak agreement).

For  $k = 9$  the accuracy is 69.12% with Kappa statistic 0.5163 (Weak agreement).

Thus the KNN classifiers are approximately 66-69% accurate and it is performing weakly compared to classifying simply by chance, With KNN ( $K=9$ ) performing the best.

Now, Let us observe the results from K Fold Cross-Validation for model with 11 predictors (welo, belo, length, w.inaccuracy, w.mistakes, w.blunders, w.acl, b.inaccuracy, b.mistakes, b.blunders, and b.acl),

k-Nearest Neighbors

```
212 samples
 11 predictor
   3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 133, 133, 133, 134, 134, 133, ...

Resampling results across tuning parameters:

k	Accuracy	Kappa
5	0.8379762	0.7456190
7	0.8251190	0.7251675
9	0.8322024	0.7365115

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was  $k = 5$ .

Here, for  $k = 5$  the accuracy is 83.80% with Kappa statistic 0.7456 (Moderate agreement).

For  $k = 7$  the accuracy is 82.51% with Kappa statistic 0.7252 (Moderate agreement).

For  $k = 9$  the accuracy is 83.22% with Kappa statistic 0.7365 (Moderate agreement).

Thus, the KNN classifiers are approximately 83-84% accurate and it is performing moderately compared to classifying simply by chance, With KNN (K=5) performing the best.

Therefore, we see an increase in overall accuracy and kappa statistic increase under K Fold Cross-Validation method as well. The accuracy range increases from 66-69% to 83-84%% and the kappa increases from weak agreement to moderate agreement.

Now, Let us observe the results from Leave-One-Out Cross-Validation for model with 3 predictors (welo, belo, and length),

k-Nearest Neighbors

```
212 samples
  3 predictor
  3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Leave-One-Out Cross-Validation

Summary of sample sizes: 147, 147, 147, 147, 147, 147, ...

Resampling results across tuning parameters:

k	Accuracy	Kappa
5	0.6554054	0.4609726
7	0.6689189	0.4800315
9	0.6891892	0.5126002

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was k = 9.

Here, for k = 5 the accuracy is 65.54% with Kappa statistic 0.4670 (Weak agreement).

For k = 7 the accuracy is 66.89% with Kappa statistic 0.4800 (Weak agreement).

For  $k = 9$  the accuracy is 68.92% with Kappa statistic 0.5126 (Weak agreement).

Thus the KNN classifiers are approximately 66-69% accurate and it is performing weakly compared to classifying simply by chance, With KNN ( $K=9$ ) performing the best.

Now, Let us observe the results from Leave-One-Out Cross-Validation for model with 11 predictors (welo, belo, length, w.inaccuracy, w.mistakes, w.blunders, w.acl, b.inaccuracy, b.mistakes, b.blunders, and b.acl)

k-Nearest Neighbors

```
212 samples
 11 predictor
   3 classes: 'black', 'draw', 'white'
```

No pre-processing

Resampling: Leave-One-Out Cross-Validation

Summary of sample sizes: 147, 147, 147, 147, 147, 147, ...

Resampling results across tuning parameters:

k	Accuracy	Kappa
5	0.8378378	0.7481030
7	0.8310811	0.7372159
9	0.8581081	0.7790417

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was  $k = 9$ .

Here, for  $k = 5$  the accuracy is 83.78% with Kappa statistic 0.7481 (Moderate agreement).

For  $k = 7$  the accuracy is 83.11% with Kappa statistic 0.7372 (Moderate agreement).

For  $k = 9$  the accuracy is 85.81% with Kappa statistic 0.7790 (Moderate agreement).

Thus the KNN classifiers are approximately 83-86% accurate and it is performing moderately compared to classifying simply by chance, With KNN (K=9) performing the best.

Therefore, we see an increase in overall accuracy and Kappa statistic increase under K Fold Cross-Validation method as well. The accuracy range increases from 66-69% to 83-86% and the Kappa increases from weak agreement to moderate agreement.

Thus, we see that the accuracy performance of the KNN classifier improves over all the selected values for K (K=5, K=7, and K=9) with addition of the engine evaluation variables, over all model validation methods.

## 4.2.5 Phase II Summaries

We may summarize the accuracies and Kappa values of each of the classifiers according to model validation method for ease of comparison between 3 predictor and 11 predictor models.

The accuracies and Kappa statistic from Confusion Matrices are,

Table 21. Accuracy and Kappa statistic of Confusion Matrices for 3 predictor vs 11 predictor models.

Method	3 Predictor Models		11 Predictor Models	
	Accuracy	Kappa	Accuracy	Kappa
LDA	0.5781	0.3224	<b>0.9063</b>	<b>0.8429</b>
QDA	0.6563	0.4393	0.8750	0.7956
Multinomial Logistic Model	0.5469	0.2647	0.8750	0.7926
KNN (K=5)	<b>0.7500</b>	<b>0.5748</b>	0.8594	0.7592
KNN (K=7)	0.7188	0.5216	0.8281	0.7103
KNN (K=9)	0.6563	0.4121	0.8594	0.7580

As discussed before, in each section we see the accuracy performance for all classifiers under Confusion Matrix improve significantly from 3 predictor model to 11 predictor model. When training the models with 3 predictors, K Nearest Neighbour (K=5) is the best model. However for the 11 predictor model Linear Discriminant Analysis classifier has the highest accuracy.

The accuracies and Kappa statistic from K fold Cross-Validation process are,

Table 22. Accuracy and Kappa statistic of K Fold CV for 3 predictor vs 11 predictor models.

	3 Predictor Models		11 Predictor Models	
Method	Accuracy	Kappa	Accuracy	Kappa
LDA	0.6819	0.5000	0.8914	0.8281
QDA	0.6066	0.3758	<b>0.9062</b>	<b>0.8561</b>
Multinomial Logistic Model	0.6741	0.4831	0.8910	0.8288
KNN (K=5)	0.6631	0.4735	0.8380	0.7456
KNN (K=7)	0.6778	0.4951	0.8251	0.7252
KNN (K=9)	<b>0.6912</b>	<b>0.5163</b>	0.8322	0.7365

As discussed beforehand, we see an increase in accuracy performance for all classifiers under K Fold Cross-Validation from 3 predictor model to 11 predictor model. When training the models with 3 predictors, K Nearest Neighbour (K=9) is the best model. However for the 11 predictor model Quadratic Discriminant Analysis classifier has the highest accuracy.

The accuracies and Kappa statistic from Leave One Out Cross-Validation process are,

Table 23. Accuracy and Kappa statistic of LOOCV for 3 predictor vs 11 predictor models.

	3 Predictor Models		11 Predictor Models	
Method	Accuracy	Kappa	Accuracy	Kappa
LDA	0.6486	0.4443	0.8784	0.8093
QDA	0.6014	0.3717	<b>0.9122</b>	<b>0.8650</b>
Multinomial Logistic Model	0.6689	0.4792	0.8851	0.82175
KNN (K=5)	0.6554	0.4697	0.8378	0.7481
KNN (K=7)	0.6689	0.4800	0.8311	0.7372
KNN (K=9)	<b>0.6892</b>	<b>0.5126</b>	0.8581	0.7790

Here, we see observe the accuracy performance for all classifiers under K Fold Cross-Validation improve significantly from 3 predictor model to 11 predictor model. When training the models with 3 predictors, K Nearest Neighbour (K=9) is the best model. However, for the 11 predictor model Quadratic Discriminant Analysis classifier has the highest accuracy.

We can carry out Hypothesis tests to check the significance of the 11 predictor model. From the test, we can obtain the variables that are significant and then train the models using only the significant variables and check whether it leads to any change in accuracy performances.

Since the response variables are categorical, we use logistic regression. The hypothesis is,

$H_0$ : *The variables are not significant in the model.*

$H_1$ : *Atleast one of the are not significant in the model.*

From logistic regression, setting the response black as the reference category.

The test statistic values,

	(Intercept)	welo	belo	length	w.inaccuracy	w.mistakes
draw	1.68951	0.2793630	0.8477952	0.1828026	0.04207232	0.7884874
white	1.67492	0.2904203	0.8321247	0.2032248	0.03729245	0.7955861
	w.blunders	w.acl	b.inaccuracy	b.mistakes	b.blunders	b.acl
draw	0.2347784	-12.51509	-0.3368477	-0.7253813	-0.3397923	4.688825
white	0.2289476	-12.90265	-0.3356292	-0.7032143	-0.3446260	4.816948

And corresponding P values,

	(Intercept)	welo	belo	length	w.inaccuracy	w.mistakes
draw	0.09112175	0.7799663	0.3965520	0.8549529	0.9664410	0.4304117
white	0.09395000	0.7714947	0.4053386	0.8389593	0.9702518	0.4262726
	w.blunders	w.acl	b.inaccuracy	b.mistakes	b.blunders	b.acl
draw	0.8143808	0	0.7362317	0.4682182	0.7340130	2.747789e-06
white	0.8189097	0	0.7371505	0.4819222	0.7303755	1.457705e-06

Let  $\alpha = 0.05$ . Here, P-values are  $< 0.05$ . Thus, all w.acl and b.acl variables are significant in the model.

It is not a surprise that these two variable, white player's average centipawn loss and black player's average centipawn loss is significant. In chess games, a lower average centipawn loss would represent minimizing the player's advantage loss. It would also lead to fewer inaccuracies, mistakes and blunders by the players, which in turn would lead to winning more games.

Therefore, we shall use w.acl and b.acl as predictor variables, with the result being the response variable with three categories, black, white, and draw, when building the machine learning classifiers based on only the significant variables.



Now, we can summarize the obtained accuracy and kappa values of the significant model under Confusion Matrix, K Fold Cross-Validation, and Leave One Out Cross-Validation in one table as follows,

Table 24. Accuracy and Kappa statistic of the significant model under different validation methods.

Method	Confusion Matrix		K Fold CV		LOOCV	
	Accuracy	Kappa	Accuracy	Kappa	Accuracy	Kappa
LDA	0.8906	0.8195	0.8306	0.7254	0.8445	0.7539
QDA	0.8906	0.8194	0.9202	0.8756	0.9256	0.8845
MLR model	<b>0.9062</b>	<b>0.8451</b>	0.9046	0.8528	0.9121	0.8636
KNN (K=5)	0.8125	0.6822	<b>0.9255</b>	<b>0.8835</b>	<b>0.9324</b>	<b>0.8946</b>
KNN (K=7)	0.8281	0.7071	0.8863	0.8209	0.9054	0.8515
KNN (K=9)	0.8281	0.7102	0.8601	0.7815	0.8918	0.8301

As we can observe here, the classifiers are producing similar or even higher accuracy and Kappa statistic when trained using only two variables, namely, the White player's average centipawn loss, and Black player's average centipawn loss.

We can compare the improvements in performance by comparing the accuracies and Kappa statistics from 3 predictor models, 11 predictor models, and significant predictor models graphically. The comparison plots are provided below. Please note that the numbers 1, 2, 3, 4, 5, and 6 in the X-axis of the plots represent LDA, QDA, Multinomial Logistic Regression, KNN(K=5), KNN(K=7), and KNN(K=9) respectively.

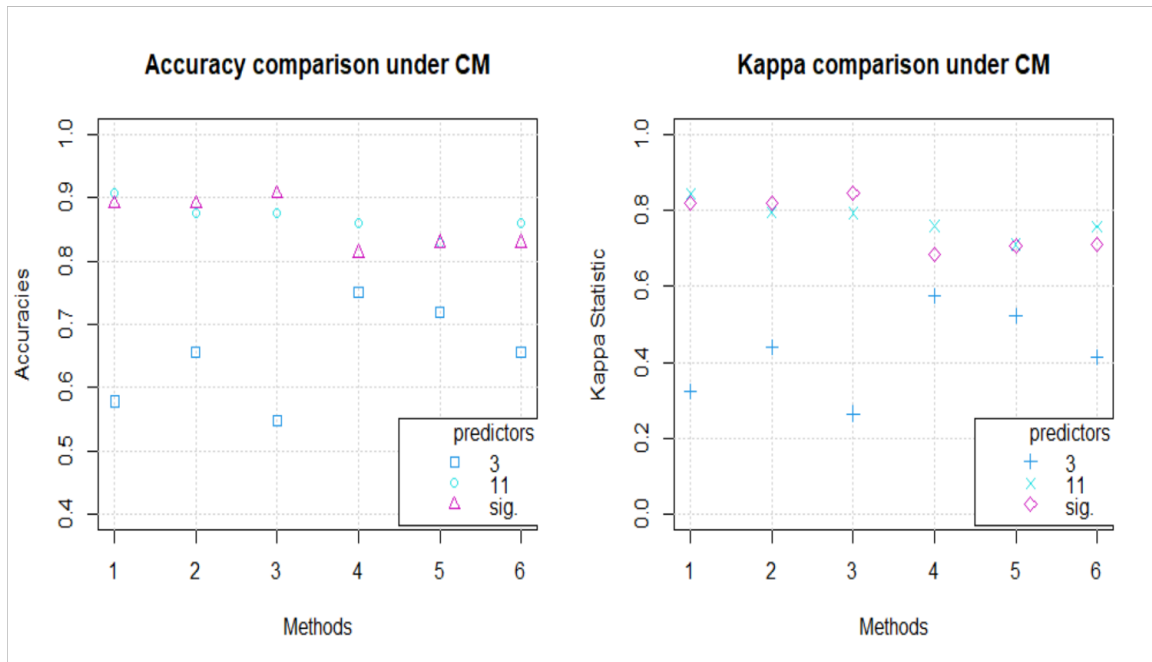


Figure 22. Performance comparison of 3, 11 and significant predictor models under CM

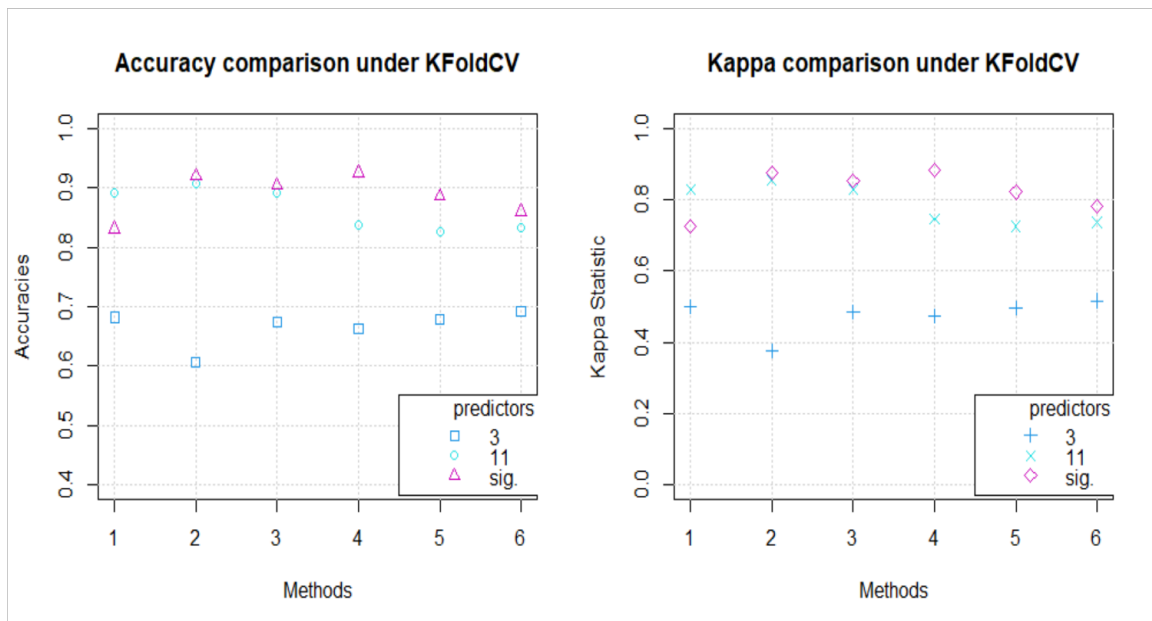


Figure 23. Performance comparison of 3, 11 and significant predictor models under KFold CV.

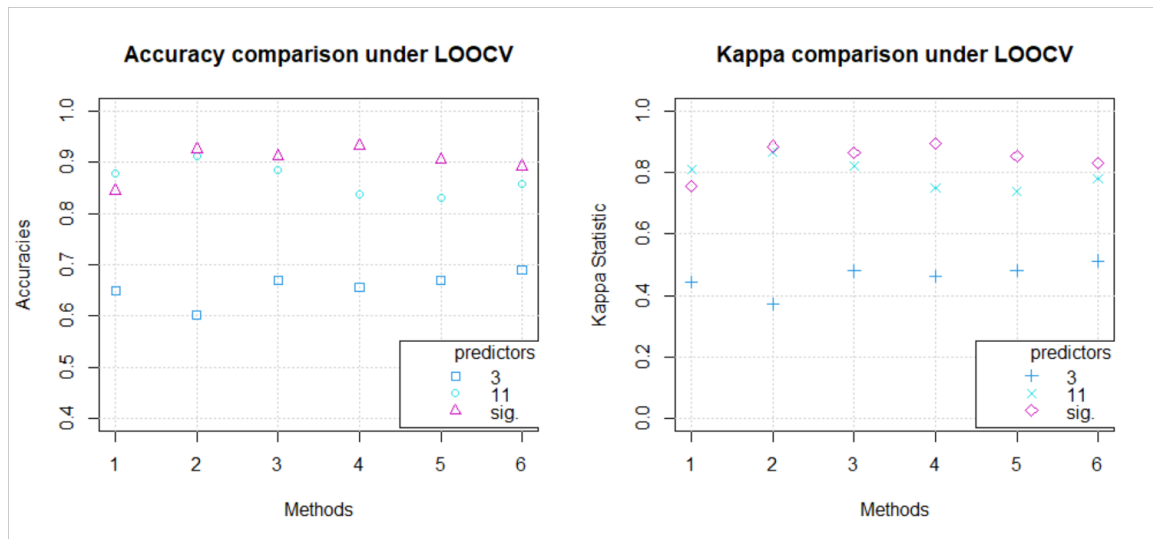


Figure 24. Performance comparison of 3, 11 and significant predictor models under LOOCV.

Therefore, we see that adding the new variables from chess engine evaluations (the white and black players' inaccuracies, mistakes, blunders, and average centipawn loss) may increase the accuracy performance of the models significantly. It can be argued that adding more predictor variables may lead to overfitting, but the increase in accuracy performance cannot be ignored.

Furthermore, we see that it is possible to obtain very high accuracy by using significant variables to train the model. In our case, even though models trained with 11 predictors have higher accuracy compared to 3 predictor models, we can also obtain the same level of accuracy by using only two of the significant predictors. Furthermore, even if we found the significant variables by modelling them under Multinomial Logistic Regression, the variables increase accuracy performance for all the classifiers.

It is important to note that, the data we use for this phase is a minute part of the larger database (only 212 observations). A future possible step should be to obtain these chess engine evaluations for the larger set of chess games from the large database of 1.94 million games to develop classifiers.

## 4.3 Phase III

In this phase of the analysis, we obtain the list of correct classifications and misclassifications by the most accurate models and investigate to see if any discernible patterns are observable. This may provide us with an idea of what generally leads classifiers to misclassify results and what may be considered as anomalies in chess games.

In phase I, we have 20058 observations in the test dataset. According to the K Fold Cross-Validation, we see that K Nearest Neighbour with  $K = 7$  provided the highest accuracy. Obtaining the predictions from the trained K Nearest Neighbour ( $K=7$ ) classifier model using 3 predictors (welo, belo, and length) and comparing the results with the actual results, we have 8954 correct classifications and 11104 misclassifications.

In all the plots, correct classifications are coloured blue, while misclassifications are coloured purple.

The scatterplot matrix of the variables is provided below.

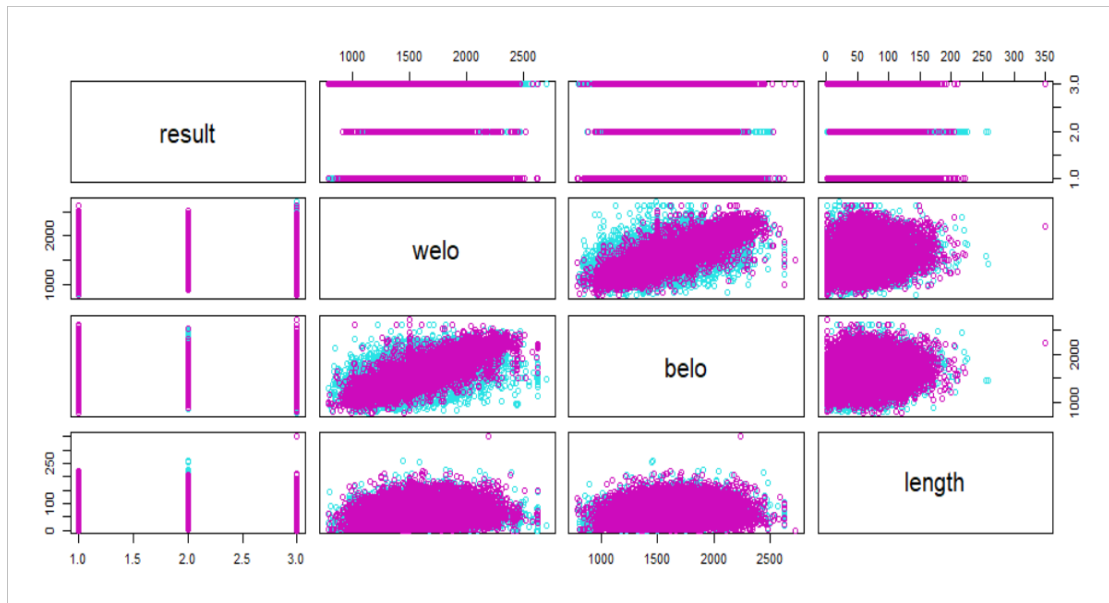


Figure 25. Scatterplot matrix of the classifications from the KNN(K=7) predictions for phase I.

We can see that the correct predictions and misclassifications overlap and have the same spread, and there is no clear pattern. We can investigate the scatterplot for each predictor variables to see if there are any trends or patterns visible.

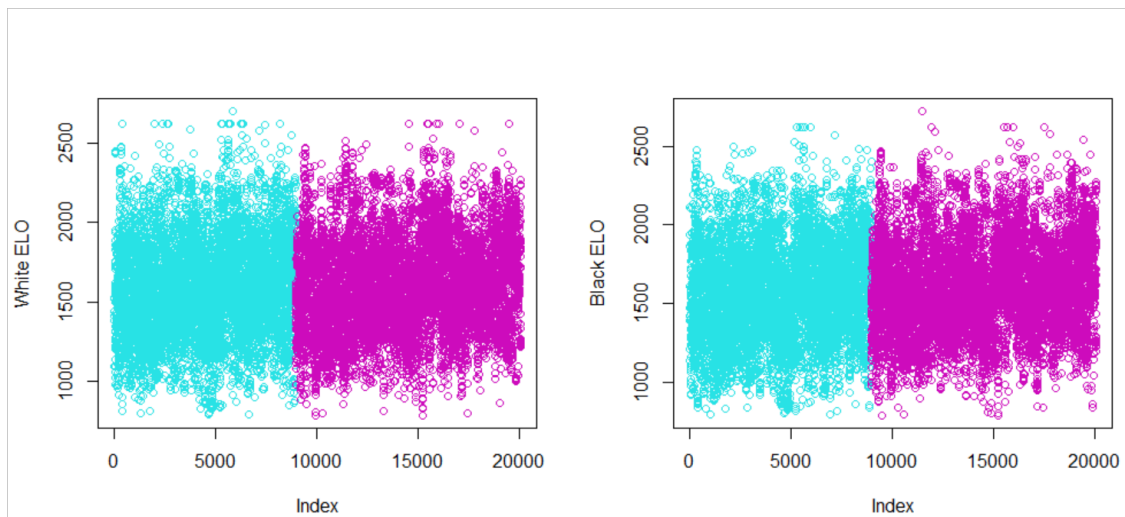


Figure 26. Scatterplots of white ELO and Black ELO from the KNN(K=7) predictions for phase I.

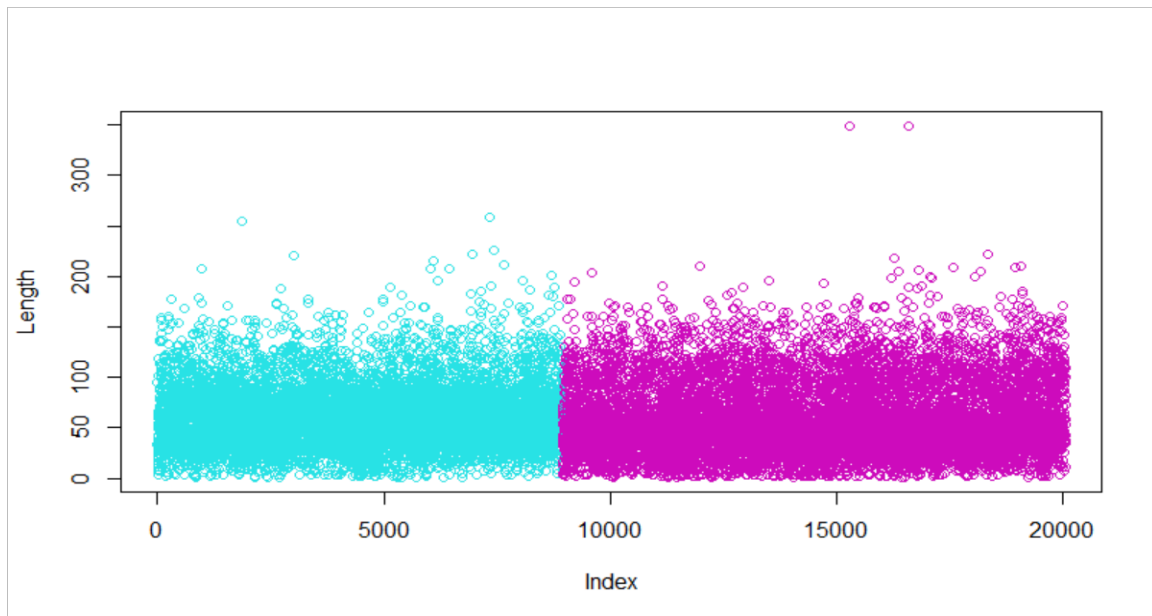


Figure 27. Scatterplots of number of turns (length) from the KNN(K=7) predictions for phase I.

We can see that the overall spread is similar for both classifications and misclassifications for all the variables. The points are scattered evenly. And there is no discernible difference.

Next, we may investigate for patterns in the smaller pool of games in phase 2. The smaller dataset may yield a cleaner scatterplot for us to investigate. Additionally, we can see if there is a difference in the scatterplots for predictions of classifiers with 3 predictors and classifiers with 11 predictors.

We know from the results in phase II, K Nearest Neighbour model with K=9 was the most accurate algorithm. Obtaining the predictions from the trained KNN (K=9) classifier model using 3 predictors (welo, belo, and length) and comparing the results with the actual results, we have 138 correct classifications and 74 misclassifications. Now we can obtain scatterplots to look for patterns. In all the

plots, correct classifications are coloured blue, while misclassifications are coloured purple.

The scatterplot matrix of the variables is provided below.

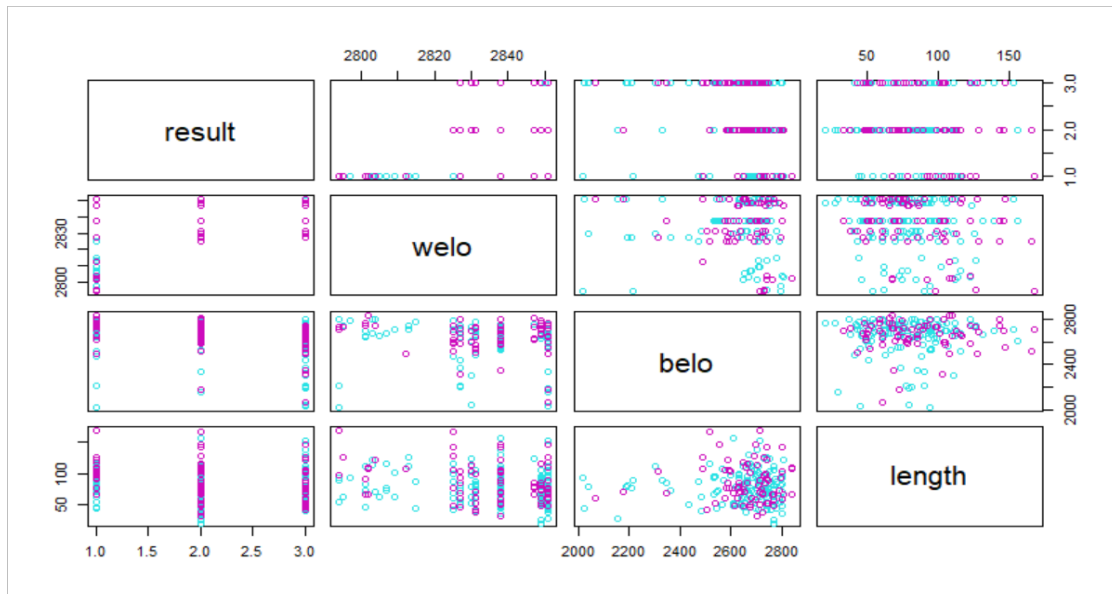


Figure 28. Scatterplot matrix of the classifications from the KNN(K=9) predictions for phase II.

We can see that the correct predictions and misclassifications are mostly adjacent, with a few misclassifications deviating from the clusters. However, there is no clear pattern.

We can investigate the scatterplot for each predictor variables to see if there are any trends or patterns visible there.

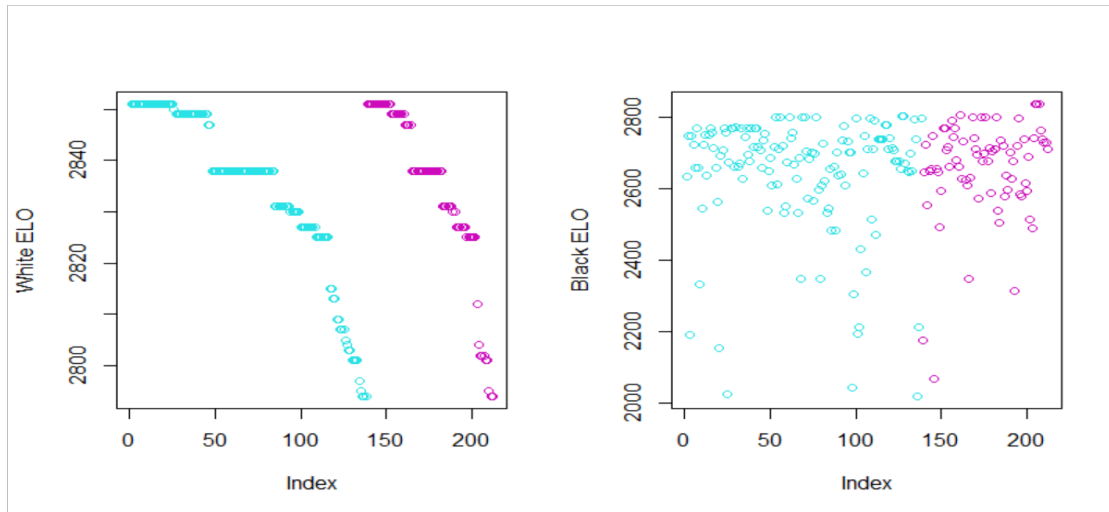


Figure 29. Scatterplots of white ELO and Black ELO from the KNN(K=9) predictions for phase II.

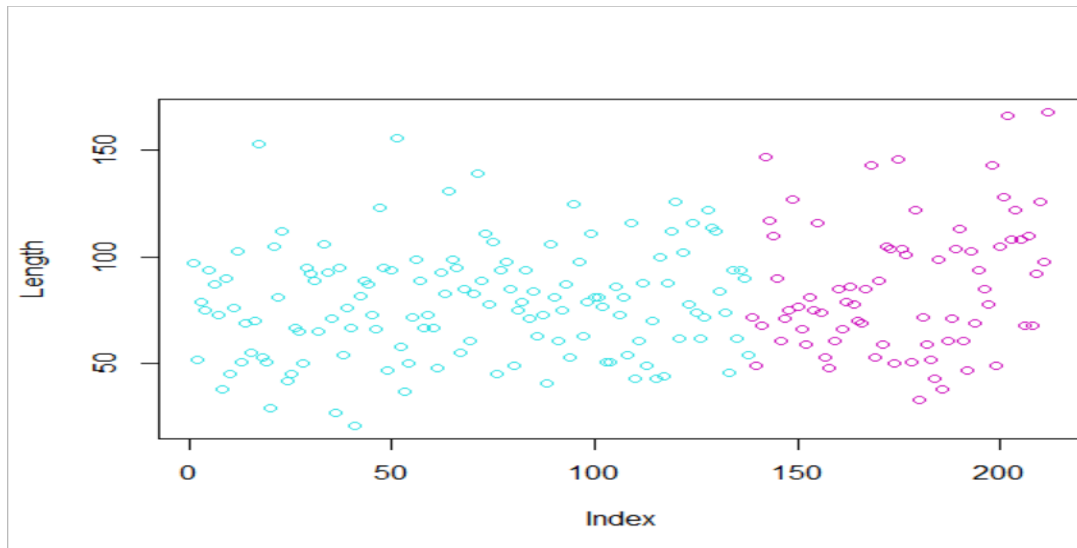


Figure 30. Scatterplots of number of turns (length) from the KNN(K=9) predictions for phase II.

We can see that the overall spread is similar for both classifications and misclassifications for all the variables. The points are scattered evenly. And there are no discernible pattern differences noticeable.

Similarly, we can carry out the same process for 11 predictor models. We know from the results in phase II, Quadratic Discriminant Model was the most accurate



algorithm. Obtaining the predictions from the trained QDA classifier model using 11 predictors (welo, belo, length, w.inaccuracy, w.mistakes, w.blunders, w.acl, b.inaccuracy, b.mistakes, b.blunders, and b.acl) and comparing the results with the actual results, we have 184 correct classifications and 28 misclassifications.

Then, we can plot a scatter plot matrix for all the variables. In all the plots, correct classifications are coloured blue, while misclassifications are coloured purple.

The scatterplot matrix of the variables is provided below.

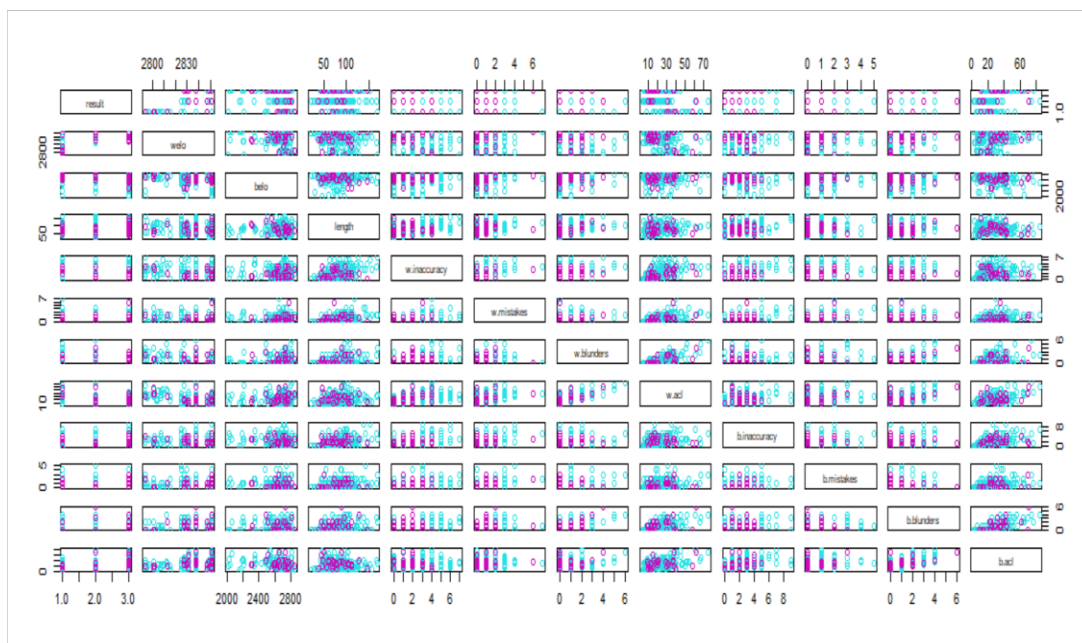


Figure 31. Scatterplot matrix of the classifications from the QDA classifier predictions for phase II.

We can observe that most of the misclassifications are still clustered with the correct classifications.

We can investigate the scatterplot for each predictor variables to see if there are any trends or patterns visible.

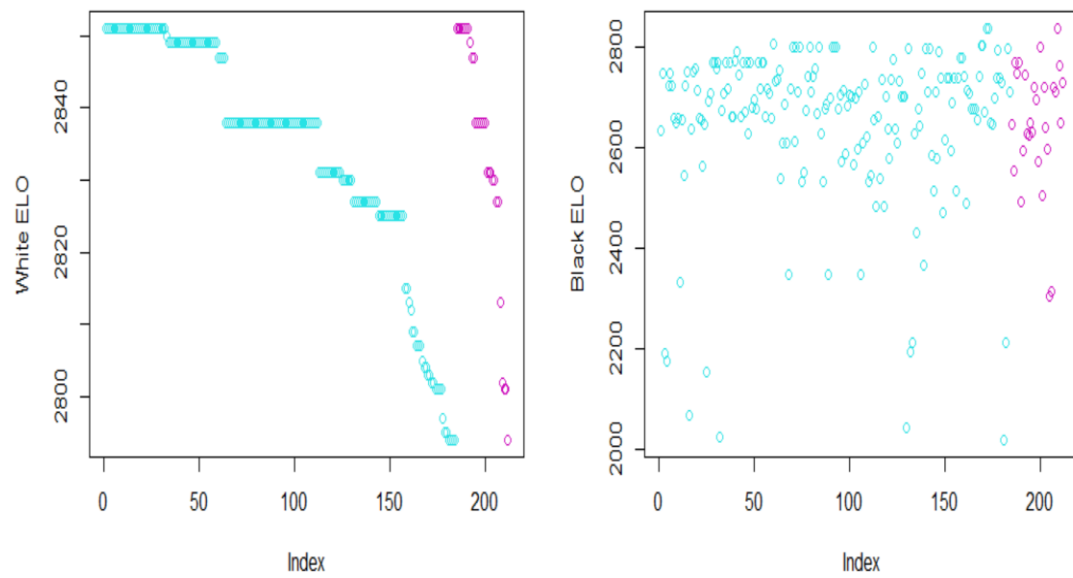


Figure 32. Scatterplots of white ELO and Black ELO from the QDA classifier predictions for phase II.

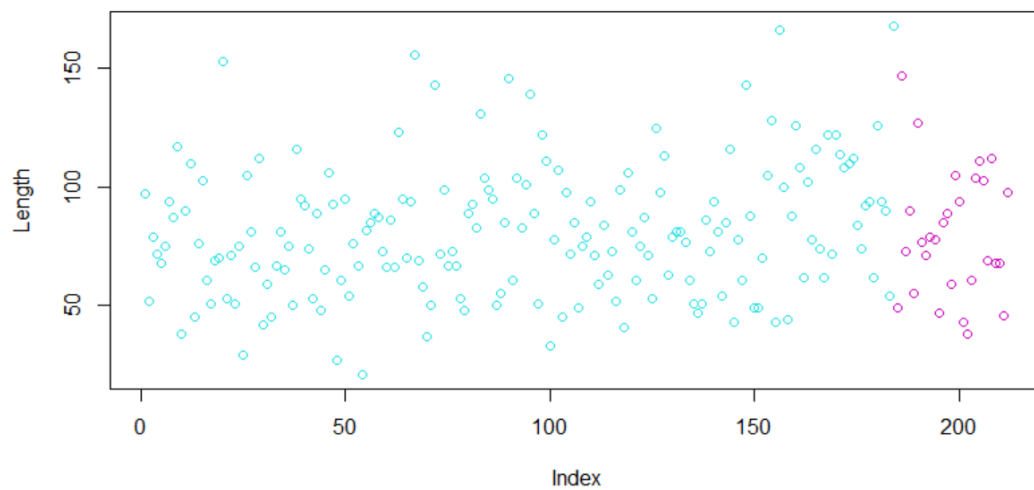


Figure 33. Scatterplots of number of turns (length) from the QDA classifier predictions for phase II.

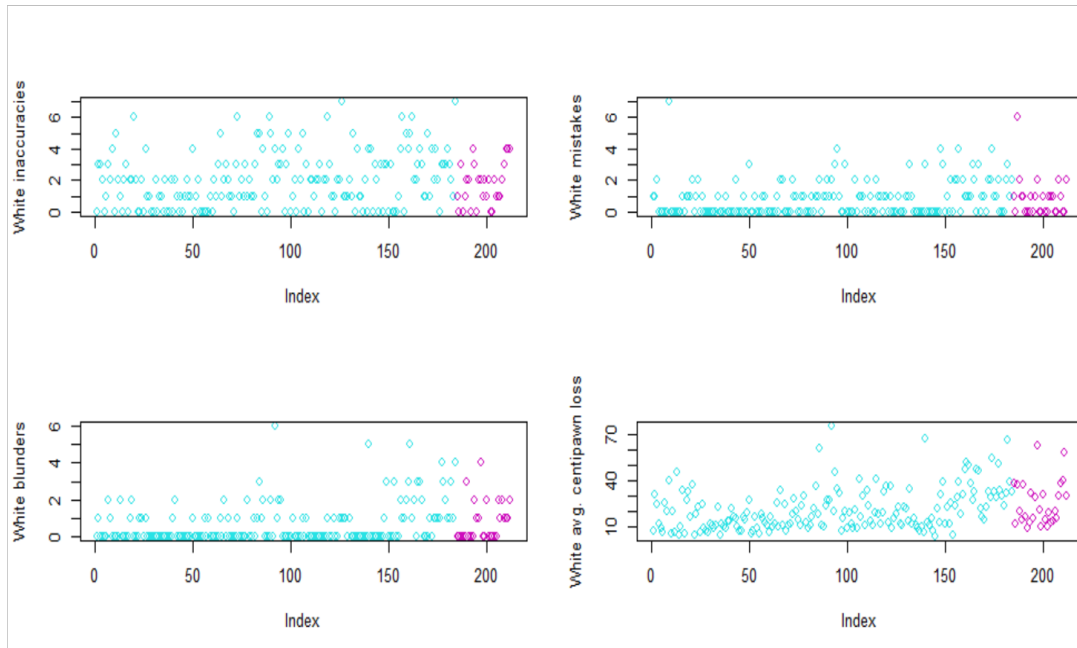


Figure 34. Scatterplots of inaccuracies, mistakes, blunders, and acl of White from QDA classifier.

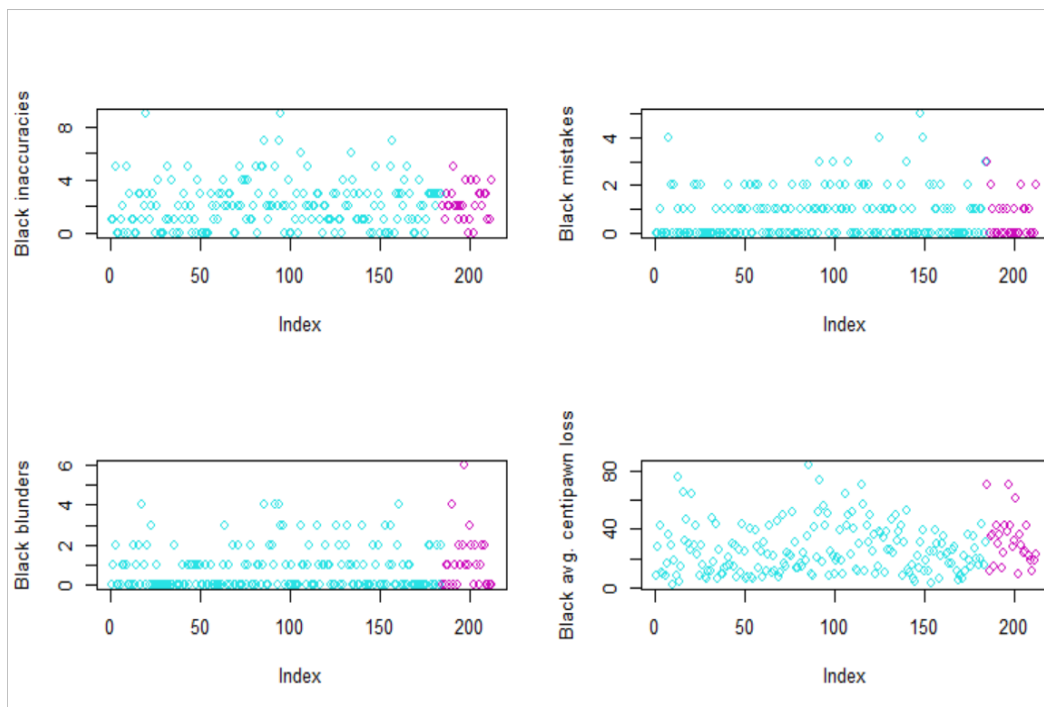


Figure 35. Scatterplots of inaccuracies, mistakes, blunders, and acl of Black from QDA classifier.

We can see that the overall spread is similar for both classifications and misclassifications for all the variables. The points are scattered evenly. And there are no discernible pattern differences that are notable between the correct classifications and misclassifications.

We see that there are no specific patterns that may be considered as anomalous. But this result is not surprising. It is quite difficult to assess which instances are beyond possibility in chess. For example, a very highly skilled player may lose to a very low skilled player due to a massive blunder, an instance that has a very low probability to happen but happens sometimes nonetheless. We require to approach this particular problem from a different and more incisive perspective, involving a further active investigation into misclassified games.

# Chapter 5: Conclusions

## 5.1 Conclusions

From phase I of the analysis, using confusion matrices, the accuracy of the classifiers ranged approximately from 44-47% with kappa statistic indicating No agreement overall. Multinomial Logistic Regression classifier produced the highest accuracy. According to the K Fold Cross-Validation method, however, the accuracy ranged approximately from 54-63% with minimal to weak agreement overall. K Nearest Neighbour with  $K = 7$  was the best model.

Hence, we see that the models are not performing at a utilizable level of accuracy. For all classifiers, it is no better than deciding the class with a coin toss. Initially, we may consider the chess games results in the test dataset having a very small proportion of draws compared to white or black winning (a case of imbalanced data) is leading the classifier to perform poorly, particularly when classifying draws. But balancing the dataset did not yield a significant increase in the accuracy performance of the classifiers, though it did improve sensitivity, positive prediction value, and balanced accuracy when predicting draws. It is important to note that the balanced testing set is quite small. Obtaining a larger balanced dataset may be better.

Next in phase II of the analysis, we replicated the modelling process with 3 initial predictors and then 11 predictors. 8 of the new predictors were obtained using Chess engine evaluations. We saw that adding these variables in the process improved the performance of all the classifiers significantly across all validation methods.

From Confusion Matrices, the accuracy increases from 57-75% to 85-90% along the Kappa values increasing to indicate a strong agreement overall for 3 predictor and 11 predictor models respectively.

From the K Fold Cross-Validation method, the accuracy increases from 60-70% to 83-90% along the Kappa values increasing to indicate a strong agreement overall for 3 predictor and 11 predictor models respectively.

From the Leave One Out Cross-Validation method, the accuracy increases from 60-69% to 83-91% along the Kappa values increasing to indicate a strong agreement overall for 3 predictor and 11 predictor models respectively.

Furthermore, Modelling using only significant variables (White player's and Black Player's Average Centipawn Loss) produced accuracies ranging from 81-93% with very strong agreement according to the Kappa coefficient. In chess games, a lower average centipawn loss would represent minimizing the player's advantage loss. It would also lead to fewer inaccuracies, mistakes and blunders by the players, which in turn would lead to winning more games.

Therefore, we may conclude that adding the variables obtained chess engine evaluations in phase I will likely improve the accuracies for all of the classifiers as well.

Finally, from phase III, we could not find any patterns or significant differences between the predictors for both correct classifications and misclassifications.

Thus, we cannot elaborate on what may be considered as an anomaly in chess games this way. We should approach this particular problem from a different and more incisive perspective.

Thus, we cannot simply judge for anomalous games using only one method. Machine learning classification is only one tool, and it may be a useful tool to spot instances that is indicative of anomalies. However, we shall need an arsenal of tools in conjunction to confirm that a particular instance is truly deviating from normalcy, or if cheating is occurring.

With the additional investigation of the misclassified games with techniques, such as, analyzing the moves played in the game, checking if certain moves strictly follow chess engine moves, if certain moves deviate from normal human moves, observing more games of the certain player whose games are frequently misclassified etc., may help to make conclusive decisions regarding if a player is cheating or not.

## 5.2 Future work

The exploration of the topic is far from exhaustive. The problem that we are trying to solve is a difficult one, and the idea discussed in this paper is only a fraction of the answer.

Originally, the objective was to develop the classifiers with hopes to utilize them for anomaly detection. But, since such data of anomalous chess games or chess games where cheating occurred is not available, we cannot say for certain whether the classifiers are operating in that capacity. Thus, for future work, we may need to collect primary data with instances of cheating so we can develop the idea further. Furthermore, if we can obtain such data, we can analyze it for patterns or trends that may explain what cheating looks like in terms of data. Understanding these phenomena can then allow us to train more precise classifiers or even simulate similar data for future testing.

Additionally, the proportions of games resulting in draws seem to be significantly lower compared to other response categories for online chess games. This is due to online games having more incentive to win rather than accepting draws. This imbalance leads to the classifiers performing poorly, particularly when classifying drawn games in this experiment. However, we expect if the testing data is balanced proportionally and large, the classifier accuracy will likely improve.

Furthermore, as observed in a small scale experiment, additional predictor variables obtained from chess engine evaluations such as frequency of inaccuracies, mistakes,



and blunders made by the players and average centipawn loss, can improve the accuracy and performance of the classifiers drastically. Therefore, another possible future endeavour can be to obtain the engine evaluations for all the 1.9 million games (or as many as possible) and check whether it performs with a similar improvement in performance.

We also saw that using significant variables for training models can help the classifiers to perform more accurately. However, we only considered and obtained 8 variables which are chess engine evaluation statistics of the moves played by the players. There is more information that can help in more accurate predictions, such as time taken for each move, centipawn gain or loss for each move, etc. Future work may be to obtain and investigate more variables that may be used as predictor variables.

We also applied Linear Discriminant Analysis and Quadratic Discriminant Analysis without testing for normality and covariance assumptions. A future possible analysis maybe to investigate which of these method is more applicable for the data at hand.

Finally, we have only used 4 classification techniques to solve this problem.

However, there are several other classification algorithms we could have used, such as Naïve Bayes, Decision Tree, Random Forest, Support Vector Machine, etc. Hence, future work may lie in exploring the efficiency and performances of these models in solving the problem.

# References

1. lichess.org open database. (2021). Retrieved 30 Mar 2021, from <https://database.lichess.org/>
2. (CHESScom), C. (2020). About Online Chess Cheating. Retrieved 30 Mar 2021, from <https://www.chess.com/article/view/online-chess-cheating>.
3. (PROChessLeague), P. (2020). Saint Louis Arch Bishops 2020 PRO Chess League Champions; Armenia Eagles Disqualified. Retrieved 30 Mar 2021, from <https://www.chess.com/news/view/saint-louis-arch-bishops-2020-pro-chess-champions>.
4. Zimek, A., & Schubert, E. (2017). Outlier Detection. Encyclopedia Of Database Systems, 1-5. doi: 10.1007/978-1-4899-7993-3\_80719-1.
5. Hodge, V., & Austin, J. (2004). A Survey of Outlier Detection Methodologies. Artificial Intelligence Review, 22(2), 85-126. Retrieved from <https://link.springer.com/article/10.1007/s10462-004-4304-y>.
6. Omar, Salima & Ngadi, Md & Jebur, Hamid & Benqdara, Salima. (2013). Machine Learning Techniques for Anomaly Detection: An Overview. International Journal of Computer Applications. 79. 10.5120/13715-1478.
7. Revathi, A., & Kumar, D. (2016). An efficient system for anomaly detection using deep learning classifier. Signal, Image And Video Processing, 11(2), 291-299. doi: 10.1007/s11760-016-0935-0.

8. Lehana, P., Kulshrestha, S., Thakur, N., & Asthana, P. (2021). International Journal of Information Engineering and Electronic Business(IJIEEB). International Journal Of Information Engineering And Electronic Business(IJIEEB), 10(4), 25. Retrieved from <http://www.mecspress.org/ijieeb/ijieeb-v10-n4/v10n4-4.html>.
9. Elo Rating System - Chess Terms. (2021). Retrieved 30 Mar 2021, from <https://www.chess.com/terms/elo-rating-chess>.
10. Chess Results Analysis Using Elo Measure with Machine Learning | Journal of Information & Knowledge Management. (2021). Journal Of Information & Knowledge Management. Retrieved from <https://www.worldscientific.com/doi/abs/10.1142/S0219649220500069>.
11. Barnes, D., & Hernandez-Castro, J. (2015). On the limits of engine analysis for cheating detection in chess. Computers & Security, 48, 58-73. doi: 10.1016/j.cose.2014.10.002.
12. Documentation for the chess dataset — chess\_web\_page 0 documentation. (2021). Retrieved 30 Mar 2021, from <https://chess-research-project.readthedocs.io/en/latest/>.
13. Chess Game Dataset (Lichess). (2021). Retrieved 30 Mar 2021, from <https://www.kaggle.com/datasnaek/chess>.
14. Johnson, R., & Wichern, D. (2019). Applied multivariate statistical analysis. Uttar Pardesh, India: Pearson India Education Services.

15. Green, S.B., Salkind, N. J., & Akey, T. M. (2008). Using SPSS for Windows and Macintosh: Analyzing and understanding data. New Jersey: Prentice Hall.
16. BÖKEOĞLU ÇOKLUK, Ö, & BÜYÜKÖZTÜRK, Ş. (2008). Discriminant function analysis: Concept and application. *Eğitim araştırmaları dergisi*, (33), 73-92.
17. Lachenbruch, P. A. (1975). Discriminant analysis. NY: Hafner
18. Klecka, William R. (1980). Discriminant analysis. Quantitative Applications in the Social Sciences Series, No. 19. Thousand Oaks, CA: Sage Publications.
19. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning : with Applications in R. New York :Springer.
20. Fix, Evelyn; Hodges, Joseph L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties (PDF) (Report). USAF School of Aviation Medicine, Randolph Field, Texas.
21. Altman, Naomi S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression" (PDF). *The American Statistician*. 46 (3): 175–185. doi:10.1080/00031305.1992.10475879. hdl:1813/31637.
22. Leif E. Peterson (2009) K-nearest neighbor. *Scholarpedia*, 4(2):1883.
23. Géron, A. (2020). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. Beijing: O'Reilly.
24. Hastie, T., Friedman, J., & Tibshirani, R. (2017). The Elements of statistical learning: Data mining, inference, and prediction. New York: Springer.

25. Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression. Hoboken, NJ: Wiley.
26. Marston, L. (2010). Introductory statistics for health and nursing using SPSS. Los Angeles: SAGE.
27. McHugh, Mary L. (2012). Interrater reliability: the kappa statistic. *Biochemia Medica*, 22(3). Retrieved from <https://pubmed.ncbi.nlm.nih.gov/23092060/>.
28. Kuhn, M. (2008). Building Predictive Models in R Using the caret Package. *Journal Of Statistical Software*, 28(5). doi: 10.18637/jss.v028.i05.
29. Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Undefined. Retrieved from <https://www.semanticscholar.org/paper/A-Study-of-Cross-Validation-and-Bootstrap-for-and-Kohavi/8c70a0a39a686bf80b76cb1b77f9eef156f6432d>.
30. Streeter, W.F. (May 1946). "Is the First Move an Advantage?". *Chess Review*. p. 16.
31. Interesting tidbits about the World Blitz Championship. (2009). Retrieved 30 Mar 2021, from <https://chessdailynews.com/interesting-tidbits-about-the-world-blitz-championship/>.
32. Resampling Methods—The solution to small datasets. (2020). Retrieved 30 Mar 2021, from <https://medium.datadriveninvestor.com/resampling-methods-the-solution-to-small-datasets-5b9e5c390eb5>

# Appendix

## Python codes

```

from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

os.chdir('E:\DATA PROJECTS\CHESSDB')
df = pd.read_csv('all_with_filtered_anotations_since1998.txt', skiprows = [0,1,2,3,4],
                 header=None)
dfs = pd.read_csv('all_with_filtered_anotations_since1998.txt',
                  names = ['1.t',
'2.date','3.result','4.welo','5.belo','6.len','7.date_c','8.resu_c','9.welo_c','10.belo_c',
'11.edate_c','12.setup','13.fen','14.resu2_c','15.oyrange','16.bad_len','17.game','1',
2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20','21',
'22','23','24','25','26','27','28','29','30','31','32','33','34','35','36','37','38','3',
9','40','41','42','43','44','45','46','47','48','49','50','51','52','53','54','55','56','',
57','58','59','60','61','62','63','64','65','66','67','68','69','70','71','72','73','74',
'75','76','77','78','79','80','81','82','83','84','85','86','87','88','89','90','91','92',
'93','94','95','96','97','98','99','100','101','102','103','104','105','106','107','108',
'109','110','111','112','113','114','115','116','117','118','119','120','121','122','123',
'124','125','126','127','128','129','130','131','132','133','134','135','136','137','13',
8','139','140','141','142','143','144','145','146','147','148','149','150','151','152','1',
53','154','155','156','157','158','159','160','161','162','163','164','165','166','167','',
168','169','170','171','172','173','174','175','176','177','178','179','180','181','182',
'183','184','185','186','187','188','189','190','191','192','193','194','195','196','197',
'198','199','200','201','202','203','204','205','206','207','208','209','210','211','212',
'213','214','215','216','217','218','219','220','221','222','223','224','225','226','22',
7','228','229','230','231','232','233','234','235','236','237','238','239','240','241','2',
42','243','244','245','246','247','248','249','250','251','252','253','254','255','256','',
257','258','259','260','261','262','263','264','265','266','267','268','269','270','271',
'272','273','274','275','276','277','278','279','280','281','282','283','284','285','286',
'287','288','289','290','291','292','293','294','295','296','297','298','299','300','301',
'302','303','304','305','306','307','308','309','310','311','312','313','314','315','31',
6','317','318','319','320','321','322','323','324','325','326','327','328','329','330','3',
31','332','333','334','335','336','337','338','339','340','341','342','343','344','345','',
346','347','348','349','350','351','352','353','354','355','356','357','358','359','360',
'361','362','363','364','365','366','367','368','369','370','371','372','373','374','375',
'376','377','378','379','380','381','382','383','384','385','386','387','388','389','390',
'391','392','393','394','395','396','397','398','399','400','401','402','403','404','40',
5','406','407','408','409','410','411','412','413','414','415','416','417','418','419','4',
20','421','422','423','424','425','426','427','428','429','430','431','432','433','434','',
435','436','437','438','439','440','441','442','443','444','445','446','447','448','449',
'450','451','452','453','454','455','456','457','458','459','460','461','462','463','464',
'465','466','467','468','469','470','471','472','473','474','475','476','477','478','479',
'480','481','482','483','484','485','486','487','488','489','490','491','492','493','49',
4','495','496','497','498','499','500'],
                 skiprows = [0,1,2,3,4], sep="\s+")
df1 = pd.read_csv('all_with_filtered_anotations_since1998.txt',
                  names = ['1.t',
'2.date','3.result','4.welo','5.belo','6.len','7.date_c','8.resu_c','9.welo_c','10.belo_c',
'11.edate_c','12.setup','13.fen','14.resu2_c','15.oyrange','16.bad_len','17.game','1',
2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20','21',
'22','23','24','25','26','27','28','29','30','31','32','33','34','35','36','37','38','3',
9','40','41','42','43','44','45','46','47','48','49','50','51','52','53','54','55','56','',
57','58','59','60','61','62','63','64','65','66','67','68','69','70','71','72','73','74',
'75','76','77','78','79','80','81','82','83','84','85','86','87','88','89','90','91','92'

```

```
, '93', '94', '95', '96', '97', '98', '99', '100', '101', '102', '103', '104', '105', '106', '107', '108',
'109', '110', '111', '112', '113', '114', '115', '116', '117', '118', '119', '120', '121', '122', '123',
'124', '125', '126', '127', '128', '129', '130', '131', '132', '133', '134', '135', '136', '137', '138',
'139', '140', '141', '142', '143', '144', '145', '146', '147', '148', '149', '150', '151', '152', '153',
'154', '155', '156', '157', '158', '159', '160', '161', '162', '163', '164', '165', '166', '167', '168',
'169', '170', '171', '172', '173', '174', '175', '176', '177', '178', '179', '180', '181', '182', '183',
'184', '185', '186', '187', '188', '189', '190', '191', '192', '193', '194', '195', '196', '197', '198',
'199', '200', '201', '202', '203', '204', '205', '206', '207', '208', '209', '210', '211', '212', '213',
'214', '215', '216', '217', '218', '219', '220', '221', '222', '223', '224', '225', '226', '227', '228',
'229', '230', '231', '232', '233', '234', '235', '236', '237', '238', '239', '240', '241', '242', '243',
'244', '245', '246', '247', '248', '249', '250', '251', '252', '253', '254', '255', '256', '257', '258',
'259', '260', '261', '262', '263', '264', '265', '266', '267', '268', '269', '270', '271', '272', '273',
'274', '275', '276', '277', '278', '279', '280', '281', '282', '283', '284', '285', '286', '287', '288',
'289', '290', '291', '292', '293', '294', '295', '296', '297', '298', '299', '300', '301', '302', '303',
'304', '305', '306', '307', '308', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318',
'319', '320', '321', '322', '323', '324', '325', '326', '327', '328', '329', '330', '331', '332', '333',
'334', '335', '336', '337', '338', '339', '340', '341', '342', '343', '344', '345', '346', '347', '348',
'349', '350', '351', '352', '353', '354', '355', '356', '357', '358', '359', '360', '361', '362', '363',
'364', '365', '366', '367', '368', '369', '370', '371', '372', '373', '374', '375', '376', '377', '378',
'379', '380', '381', '382', '383', '384', '385', '386', '387', '388', '389', '390', '391', '392', '393',
'394', '395', '396', '397', '398', '399', '400', '401', '402', '403', '404', '405', '406', '407', '408',
'409', '410', '411', '412', '413', '414', '415', '416', '417', '418', '419', '420', '421', '422', '423',
'424', '425', '426', '427', '428', '429', '430', '431', '432', '433', '434', '435', '436', '437', '438',
'439', '440', '441', '442', '443', '444', '445', '446', '447', '448', '449', '450', '451', '452', '453',
'454', '455', '456', '457', '458', '459', '460', '461', '462', '463', '464', '465', '466', '467', '468',
'469', '470', '471', '472', '473', '474', '475', '476', '477', '478', '479', '480', '481', '482', '483',
'484', '485', '486', '487', '488', '489', '490', '491', '492', '493', '494', '495', '496', '497', '498',
'499', '500'],
    skiprows = [0,1,2,3,4], nrows=1000000, sep="\s+")
df2 = pd.read_csv('all_with_filtered_anotations_since1998.txt',
    names = ['l.t',
'2.date', '3.result', '4.welo', '5.belo', '6.len', '7.date_c', '8.resu_c', '9.welo_c', '10.belo_c',
'11.edate_c', '12.setup', '13.fen', '14.resu2_c', '15.oynrange', '16.bad_len', '17.game', '1',
'2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21',
'22', '23', '24', '25', '26', '27', '28', '29', '30', '31', '32', '33', '34', '35', '36', '37', '38', '39',
'40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '57',
'58', '59', '60', '61', '62', '63', '64', '65', '66', '67', '68', '69', '70', '71', '72', '73', '74', '75',
'76', '77', '78', '79', '80', '81', '82', '83', '84', '85', '86', '87', '88', '89', '90', '91', '92',
'93', '94', '95', '96', '97', '98', '99', '100', '101', '102', '103', '104', '105', '106', '107', '108',
'109', '110', '111', '112', '113', '114', '115', '116', '117', '118', '119', '120', '121', '122', '123',
'124', '125', '126', '127', '128', '129', '130', '131', '132', '133', '134', '135', '136', '137', '138',
'139', '140', '141', '142', '143', '144', '145', '146', '147', '148', '149', '150', '151', '152', '153',
'154', '155', '156', '157', '158', '159', '160', '161', '162', '163', '164', '165', '166', '167', '168',
'169', '170', '171', '172', '173', '174', '175', '176', '177', '178', '179', '180', '181', '182', '183',
'184', '185', '186', '187', '188', '189', '190', '191', '192', '193', '194', '195', '196', '197', '198',
'199', '200', '201', '202', '203', '204', '205', '206', '207', '208', '209', '210', '211', '212', '213',
'214', '215', '216', '217', '218', '219', '220', '221', '222', '223', '224', '225', '226', '227', '228',
'229', '230', '231', '232', '233', '234', '235', '236', '237', '238', '239', '240', '241', '242', '243',
'244', '245', '246', '247', '248', '249', '250', '251', '252', '253', '254', '255', '256', '257', '258',
'259', '260', '261', '262', '263', '264', '265', '266', '267', '268', '269', '270', '271', '272', '273',
'274', '275', '276', '277', '278', '279', '280', '281', '282', '283', '284', '285', '286', '287', '288',
'289', '290', '291', '292', '293', '294', '295', '296', '297', '298', '299', '300', '301', '302', '303',
'304', '305', '306', '307', '308', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318',
'319', '320', '321', '322', '323', '324', '325', '326', '327', '328', '329', '330', '331', '332', '333',
'334', '335', '336', '337', '338', '339', '340', '341', '342', '343', '344', '345', '346', '347', '348',
'349', '350', '351', '352', '353', '354', '355', '356', '357', '358', '359', '360', '361', '362', '363',
'364', '365', '366', '367', '368', '369', '370', '371', '372', '373', '374', '375', '376', '377', '378',
'379', '380', '381', '382', '383', '384', '385', '386', '387', '388', '389', '390', '391', '392', '393',
'394', '395', '396', '397', '398', '399', '400', '401', '402', '403', '404', '405', '406', '407', '408',
'409', '410', '411', '412', '413', '414', '415', '416', '417', '418', '419', '420', '421', '422', '423',
'424', '425', '426', '427', '428', '429', '430', '431', '432', '433', '434', '435', '436', '437', '438',
'439', '440', '441', '442', '443', '444', '445', '446', '447', '448', '449', '450', '451', '452', '453',
'454', '455', '456', '457', '458', '459', '460', '461', '462', '463', '464', '465', '466', '467', '468',
'469', '470', '471', '472', '473', '474', '475', '476', '477', '478', '479', '480', '481', '482', '483',
'484', '485', '486', '487', '488', '489', '490', '491', '492', '493', '494', '495', '496', '497', '498',
'499', '500'],
    skiprows = 1000005, nrows=1000000, sep="\s+")
```

```

df3= pd.read_csv('all_with_filtered_anotations_since1998.txt',
                 names = ['1.t',
'2.date','3.result','4.welo','5.belo','6.len','7.date_c','8.resu_c','9.welo_c','10.belo_c',
'11.edate_c','12.setup','13.fen','14.resu2_c','15.oynrange','16.bad len','17.game','1',
2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20','21',
'22','23','24','25','26','27','28','29','30','31','32','33','34','35','36','37','38','3',
9','40','41','42','43','44','45','46','47','48','49','50','51','52','53','54','55','56','5',
57','58','59','60','61','62','63','64','65','66','67','68','69','70','71','72','73','74',
'75','76','77','78','79','80','81','82','83','84','85','86','87','88','89','90','91','92',
'93','94','95','96','97','98','99','100','101','102','103','104','105','106','107','108',
'109','110','111','112','113','114','115','116','117','118','119','120','121','122','123',
'124','125','126','127','128','129','130','131','132','133','134','135','136','137','13',
8','139','140','141','142','143','144','145','146','147','148','149','150','151','152','1',
53','154','155','156','157','158','159','160','161','162','163','164','165','166','167',
168','169','170','171','172','173','174','175','176','177','178','179','180','181','182',
'183','184','185','186','187','188','189','190','191','192','193','194','195','196','197',
'198','199','200','201','202','203','204','205','206','207','208','209','210','211','212',
'213','214','215','216','217','218','219','220','221','222','223','224','225','226','22',
7','228','229','230','231','232','233','234','235','236','237','238','239','240','241','2',
42','243','244','245','246','247','248','249','250','251','252','253','254','255','256',
257','258','259','260','261','262','263','264','265','266','267','268','269','270','271',
'272','273','274','275','276','277','278','279','280','281','282','283','284','285','286',
'287','288','289','290','291','292','293','294','295','296','297','298','299','300','301',
'302','303','304','305','306','307','308','309','310','311','312','313','314','315','31',
6','317','318','319','320','321','322','323','324','325','326','327','328','329','330','3',
31','332','333','334','335','336','337','338','339','340','341','342','343','344','345',
346','347','348','349','350','351','352','353','354','355','356','357','358','359','360',
'361','362','363','364','365','366','367','368','369','370','371','372','373','374','375',
'376','377','378','379','380','381','382','383','384','385','386','387','388','389','390',
'391','392','393','394','395','396','397','398','399','400','401','402','403','404','40',
5','406','407','408','409','410','411','412','413','414','415','416','417','418','419','4',
20','421','422','423','424','425','426','427','428','429','430','431','432','433','434',
435','436','437','438','439','440','441','442','443','444','445','446','447','448','449',
'450','451','452','453','454','455','456','457','458','459','460','461','462','463','464',
'465','466','467','468','469','470','471','472','473','474','475','476','477','478','479',
'480','481','482','483','484','485','486','487','488','489','490','491','492','493','49',
4','495','496','497','498','499','500'],
                 skiprows = 2000005, nrows=999990, sep="\s+")
df3= pd.read_csv('all_with_filtered_anotations_since1998.txt',
                 names = ['1.t',
'2.date','3.result','4.welo','5.belo','6.len','7.date_c','8.resu_c','9.welo_c','10.belo_c',
'11.edate_c','12.setup','13.fen','14.resu2_c','15.oynrange','16.bad len','17.game','1',
2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20','21',
'22','23','24','25','26','27','28','29','30','31','32','33','34','35','36','37','38','3',
9','40','41','42','43','44','45','46','47','48','49','50','51','52','53','54','55','56','5',
57','58','59','60','61','62','63','64','65','66','67','68','69','70','71','72','73','74',
'75','76','77','78','79','80','81','82','83','84','85','86','87','88','89','90','91','92',
'93','94','95','96','97','98','99','100','101','102','103','104','105','106','107','108',
'109','110','111','112','113','114','115','116','117','118','119','120','121','122','123',
'124','125','126','127','128','129','130','131','132','133','134','135','136','137','13',
8','139','140','141','142','143','144','145','146','147','148','149','150','151','152','1',
53','154','155','156','157','158','159','160','161','162','163','164','165','166','167',
168','169','170','171','172','173','174','175','176','177','178','179','180','181','182',
'183','184','185','186','187','188','189','190','191','192','193','194','195','196','197',
'198','199','200','201','202','203','204','205','206','207','208','209','210','211','212',
'213','214','215','216','217','218','219','220','221','222','223','224','225','226','22',
7','228','229','230','231','232','233','234','235','236','237','238','239','240','241','2',
42','243','244','245','246','247','248','249','250','251','252','253','254','255','256',
257','258','259','260','261','262','263','264','265','266','267','268','269','270','271',
'272','273','274','275','276','277','278','279','280','281','282','283','284','285','286',
'287','288','289','290','291','292','293','294','295','296','297','298','299','300','301',
'302','303','304','305','306','307','308','309','310','311','312','313','314','315','31',
6','317','318','319','320','321','322','323','324','325','326','327','328','329','330','3',
31','332','333','334','335','336','337','338','339','340','341','342','343','344','345',
346','347','348','349','350','351','352','353','354','355','356','357','358','359','360',
'361','362','363','364','365','366','367','368','369','370','371','372','373','374','375',
'376','377','378','379','380','381','382','383','384','385','386','387','388','389','390',
'391','392','393','394','395','396','397','398','399','400','401','402','403','404','40',
5','406','407','408','409','410','411','412','413','414','415','416','417','418','419','4',
20','421','422','423','424','425','426','427','428','429','430','431','432','433','434',
435','436','437','438','439','440','441','442','443','444','445','446','447','448','449',
'450','451','452','453','454','455','456','457','458','459','460','461','462','463','464',
'465','466','467','468','469','470','471','472','473','474','475','476','477','478','479',
'480','481','482','483','484','485','486','487','488','489','490','491','492','493','49',
4','495','496','497','498','499','500'],
                 skiprows = 2000005, nrows=999990, sep="\s+")

```



```

', '391', '392', '393', '394', '395', '396', '397', '398', '399', '400', '401', '402', '403', '404', '40
5', '406', '407', '408', '409', '410', '411', '412', '413', '414', '415', '416', '417', '418', '419', '4
20', '421', '422', '423', '424', '425', '426', '427', '428', '429', '430', '431', '432', '433', '434', '
435', '436', '437', '438', '439', '440', '441', '442', '443', '444', '445', '446', '447', '448', '449',
'450', '451', '452', '453', '454', '455', '456', '457', '458', '459', '460', '461', '462', '463', '464',
'465', '466', '467', '468', '469', '470', '471', '472', '473', '474', '475', '476', '477', '478', '479
', '480', '481', '482', '483', '484', '485', '486', '487', '488', '489', '490', '491', '492', '493', '49
4', '495', '496', '497', '498', '499', '500'],
    skiprows = 2000005, nrows=999990, sep="\s+")
df4= pd.read_csv('all with filtered_annotaions_since1998.txt',
    names = ['1.t',
'2.date', '3.result', '4.welo', '5.belo', '6.len', '7.date_c', '8.resu_c', '9.welo_c', '10.belo_c
', '11.edate_c', '12.setup', '13.fen', '14.resu2_c', '15.oyrange', '16.bad_len', '17.game', '1', '
2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21
', '22', '23', '24', '25', '26', '27', '28', '29', '30', '31', '32', '33', '34', '35', '36', '37', '38', '3
9', '40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '
57', '58', '59', '60', '61', '62', '63', '64', '65', '66', '67', '68', '69', '70', '71', '72', '73', '74',
'75', '76', '77', '78', '79', '80', '81', '82', '83', '84', '85', '86', '87', '88', '89', '90', '91', '92',
'93', '94', '95', '96', '97', '98', '99', '100', '101', '102', '103', '104', '105', '106', '107', '108',
'109', '110', '111', '112', '113', '114', '115', '116', '117', '118', '119', '120', '121', '122', '123
', '124', '125', '126', '127', '128', '129', '130', '131', '132', '133', '134', '135', '136', '137', '13
8', '139', '140', '141', '142', '143', '144', '145', '146', '147', '148', '149', '150', '151', '152', '1
53', '154', '155', '156', '157', '158', '159', '160', '161', '162', '163', '164', '165', '166', '167', '
168', '169', '170', '171', '172', '173', '174', '175', '176', '177', '178', '179', '180', '181', '182',
'183', '184', '185', '186', '187', '188', '189', '190', '191', '192', '193', '194', '195', '196', '197',
'198', '199', '200', '201', '202', '203', '204', '205', '206', '207', '208', '209', '210', '211', '212
', '213', '214', '215', '216', '217', '218', '219', '220', '221', '222', '223', '224', '225', '226', '22
7', '228', '229', '230', '231', '232', '233', '234', '235', '236', '237', '238', '239', '240', '241', '2
42', '243', '244', '245', '246', '247', '248', '249', '250', '251', '252', '253', '254', '255', '256', '
257', '258', '259', '260', '261', '262', '263', '264', '265', '266', '267', '268', '269', '270', '271',
'272', '273', '274', '275', '276', '277', '278', '279', '280', '281', '282', '283', '284', '285', '286',
'287', '288', '289', '290', '291', '292', '293', '294', '295', '296', '297', '298', '299', '300', '301
', '302', '303', '304', '305', '306', '307', '308', '309', '310', '311', '312', '313', '314', '315', '31
6', '317', '318', '319', '320', '321', '322', '323', '324', '325', '326', '327', '328', '329', '330', '3
31', '332', '333', '334', '335', '336', '337', '338', '339', '340', '341', '342', '343', '344', '345', '
346', '347', '348', '349', '350', '351', '352', '353', '354', '355', '356', '357', '358', '359', '360',
'361', '362', '363', '364', '365', '366', '367', '368', '369', '370', '371', '372', '373', '374', '375',
'376', '377', '378', '379', '380', '381', '382', '383', '384', '385', '386', '387', '388', '389', '390
', '391', '392', '393', '394', '395', '396', '397', '398', '399', '400', '401', '402', '403', '404', '40
5', '406', '407', '408', '409', '410', '411', '412', '413', '414', '415', '416', '417', '418', '419', '4
20', '421', '422', '423', '424', '425', '426', '427', '428', '429', '430', '431', '432', '433', '434', '
435', '436', '437', '438', '439', '440', '441', '442', '443', '444', '445', '446', '447', '448', '449',
'450', '451', '452', '453', '454', '455', '456', '457', '458', '459', '460', '461', '462', '463', '464',
'465', '466', '467', '468', '469', '470', '471', '472', '473', '474', '475', '476', '477', '478', '479
', '480', '481', '482', '483', '484', '485', '486', '487', '488', '489', '490', '491', '492', '493', '49
4', '495', '496', '497', '498', '499', '500'],
    skiprows = 2999995, nrows=1000000, sep="\s+")
df1.to_csv(r'E:\\DATA PROJECTS\\CHESSDB\\test1.csv')
df2.to_csv(r'E:\\DATA PROJECTS\\CHESSDB\\test2.csv')
df3.to_csv(r'E:\\DATA PROJECTS\\CHESSDB\\test3.csv')
df4.to_csv(r'E:\\DATA PROJECTS\\CHESSDB\\test4.csv')
df = pd.read_csv('test1.csv', sep="")
data = df[""].split(',', expand=True)

```

## R codes

```

####Database Preprocessing codes####
predata <- read.csv(file="E:/DATA PROJECTS/CHESSDB/test1.csv", header=TRUE)
data <-
data.frame(predata$X1,predata$X1.t,predata$X2.date,predata$X3.result,predata$X4.welo,predata$X5.belo,predata$X6.len,predata$X7.date_c,predata$X8.resu_c,predata$X9.welo_c,predata$X10.belo_c,predata$X11.edate_c,predata$X12.setup,predata$X13.fen,predata$X14.resu2_c,predata$X15.oyrange,predata$X16.bad_len)
write.csv(data, file="E:/DATA PROJECTS/CHESSDB/test1.csv")

```

```

predata <- read.csv(file="E:/DATA PROJECTS/CHESSDB/test2.csv", header=TRUE)
data <-
data.frame(predata$X,predata$X1.t,predata$X2.date,predata$X3.result,predata$X4.welo,preda
ta$X5.belo,predata$X6.len,predata$X7.date_c,predata$X8.resu_c,predata$X9.welo_c,predata$X
10.belo_c,predata$X11.edate_c,predata$X12.setup,predata$X13.fen,predata$X14.resu2_c,preda
ta$X15.oyrange,predata$X16.bad_len)
write.csv(data, file="E:/DATA PROJECTS/CHESSDB/test2.csv")
predata <- read.csv(file="E:/DATA PROJECTS/CHESSDB/test3.csv", header=TRUE)
data <-
data.frame(predata$X,predata$X1.t,predata$X2.date,predata$X3.result,predata$X4.welo,preda
ta$X5.belo,predata$X6.len,predata$X7.date_c,predata$X8.resu_c,predata$X9.welo_c,predata$X
10.belo_c,predata$X11.edate_c,predata$X12.setup,predata$X13.fen,predata$X14.resu2_c,preda
ta$X15.oyrange,predata$X16.bad_len)
write.csv(data, file="E:/DATA PROJECTS/CHESSDB/test3.csv")
predata <- read.csv(file="E:/DATA PROJECTS/CHESSDB/test4.csv", header=TRUE)
data <-
data.frame(predata$X,predata$X1.t,predata$X2.date,predata$X3.result,predata$X4.welo,preda
ta$X5.belo,predata$X6.len,predata$X7.date_c,predata$X8.resu_c,predata$X9.welo_c,predata$X
10.belo_c,predata$X11.edate_c,predata$X12.setup,predata$X13.fen,predata$X14.resu2_c,preda
ta$X15.oyrange,predata$X16.bad_len)
write.csv(data, file="E:/DATA PROJECTS/CHESSDB/test4.csv")
#combining data blocks
data1 <- read.csv(file="E:/DATA PROJECTS/CHESSDB/test1.csv", header=TRUE)
data2 <- read.csv(file="E:/DATA PROJECTS/CHESSDB/test2.csv", header=TRUE)
data3 <- read.csv(file="E:/DATA PROJECTS/CHESSDB/test3.csv", header=TRUE)
data4 <- read.csv(file="E:/DATA PROJECTS/CHESSDB/test4.csv", header=TRUE)
data <- rbind(data1,data2,data3,data4)
colnames(data)<-
c("X","p.X","t","date","result","welo","belo","len","date_c","resu_c","welo_c","belo_c","
edate_c","setup","fen","resu2_c","oyrange","bad_len")
write.csv(data, file="E:/DATA PROJECTS/CHESSDB/dataclean1.csv")
#Inputting data
df <- read.csv(file="E:/DATA PROJECTS/CHESSDB/dataclean1.csv", header=TRUE)
# sample
ind = sample(nrow(df), 100, replace = FALSE)
s = df[ind,]
write.csv(s, file = "E:/s.csv")
# turning all undesirable data points into NA then omitting them
df$welo_c <- replace(df$welo_c, df$welo_c == "welo_true", NA )
df$belo_c <- replace(df$belo_c, df$belo_c == "belo_true", NA )
df$resu_c <- replace(df$resu_c, df$resu_c == "result_true", NA )
df$resu2_c <- replace(df$resu2_c, df$resu2_c == "result2_true", NA )
df$setup <- replace(df$setup, df$setup == "setup_true", NA )
df$bad_len <- replace(df$bad_len, df$bad_len == "blen_true", NA )
df$result <- replace(df$result, df$result == "*", NA )
df <- na.omit(df)
#recoding results
df$result <- replace(df$result, df$result == "1-0", "white")
df$result <- replace(df$result, df$result == "0-1", "black")
df$result <- replace(df$result, df$result == "1/2-1/2", "draw")
#Setting up the factors
df$result<- as.factor(df$result)
df$date_c<- as.factor(df$date_c)
df$resu_c<- as.factor(df$resu_c)
df$welo_c<- as.factor(df$welo_c)
df$belo_c<- as.factor(df$belo_c)
df$edate_c<- as.factor(df$edate_c)
df$setup<- as.factor(df$setup)
df$fen<- as.factor(df$fen)
df$resu2_c<- as.factor(df$resu2_c)
df$oyrange<- as.factor(df$oyrange)
df$bad_len<- as.factor(df$bad_len)
#Cheking the dataframe
str(df)
#Final data
data <- data.frame(df$result,df$welo,df$belo,df$len)

```

```

#Writing the final dataset into a clean CSV
write.csv(data, file = "E:/DATA PROJECTS/CHESSDB/chessdb.csv")

#### Getting the PGN
smallldb <- read.csv(file="E:/DATA PROJECTS/CHESSDB/smallldb.csv", header=TRUE)
gamenote<-smallldb[,9:211]
gamenote[is.na(gamenote)] <- "NA."
sum(is.na(gamenote))
spf<- function(dataframe){
  spl.df <- matrix(NA, nrow(dataframe), ncol(dataframe))
  for (j in 1: ncol(dataframe)){
    spl.df[,j]<- as.vector(sapply(dataframe[,j], function(x) unlist(strsplit(x, split =
".", fixed = T))[2]))
  }
  return(spl.df)
}

PGN <- as.data.frame(spf(gamenote))
PGN <- cbind(smallldb$X1.t,PGN)
write.csv(PGN, file="E:/DATA PROJECTS/CHESSDB/PGN.csv")

##### CODES FOR THE ANALYSIS BEGINS HERE #####
set.seed(123)
#libraries
library(MASS)
library(PASWR)
library(EnvStats)
library(VGAM)
library(nnet)
library(ROSE)
library(caret)
library(pROC)
library(class)
setwd("M:/My_Private_Files/Thesis/Data")

#Reading the OTB data
df <- read.csv(file="chessdb.csv", header=TRUE)
colnames(df)<- c("t", "result", "welo", "belo", "length")
df$result<- as.factor(df$result)

#checking data
str(df)

#basic statistics
summary(df)
table(df$result)
prop.table(table(df$result))

#standardised
swelo<- (df$welo - mean(df$welo))/sd(df$welo)
sbelo<- (df$belo - mean(df$belo))/sd(df$belo)
slength<- (df$length - mean(df$length))/sd(df$length)
sdf <- data.frame(df$result, swelo, sbelo, slength)

#### Online games ####
games <- read.csv("games.csv")
df.on<-data.frame(games$winner,games$white_rating,games$black_rating,games$turns)
colnames(df.on)<-c("result", "welo", "belo", "length")
df.on$result <- as.factor(df.on$result)

#checking data
str(df.on)
dim(df.on)

#basic statistics
summary(df.on)

```

```

table(df.on$result)
prop.table(table(df.on$result))

#TESTING FOR SIGNIFICANCE IN THE MODEL
df.all<-rbind(df[,2:5],df.on)
head(df.all)
logitmod.t <- multinom(result~welo+belo+length,data = df.all)
logit.sum.tt <- summary(logitmod.t)
logit.z.tt <- logit.sum.tt$coefficients/logit.sum.tt$standard.errors
logit.z.tt
# 2-tailed z test
p.tt <- (1 - pnorm(abs(logit.z.tt), 0, 1)) * 2
p.tt

#standardised
swelo<- (df.on$welo - mean(df.on$welo))/sd(df.on$welo)
sbelo<- (df.on$belo - mean(df.on$belo))/sd(df.on$belo)
slength<- (df.on$length - mean(df.on$length))/sd(df.on$length)
sdf.on <- data.frame(df.on$result, swelo, sbelo, slength)
str(sdf.on)

#Dividing into train and test
train<-sdf
test<-sdf.on
colnames(test) <- c("df.result", "swelo", "sbelo", "slength")

fsdf<-rbind(train,test)
fstrain<-fsdf[1:1942330,]
fstest<-fsdf[-(1:1942330),]
fsdt = 1:1942330

fstrain.x = cbind(sdf$swelo, sdf$sbelo, sdf$slength)
fstest.x = cbind(sdf.on$swelo, sdf.on$sbelo, sdf.on$slength)

#####
##### PHASE I #####
#####

# Define training control
train.control <- trainControl(method = "cv", number = 10)

##### LDA model #####
kfold.lda <- train(df.result ~ swelo+sbelo+slength,
                  data=fsdf, subset=fsdt, method ="lda",
                  trControl = train.control)
print(kfold.lda)

##LDA confusion matrix##
lda.fit=lda(df.result ~ swelo+sbelo+slength, data=fsdf, subset=fsdt)
lda.pred= predict(lda.fit, fstest)
lda.class =lda.pred$class
lda.cm = confusionMatrix(fstest$df.result,as.factor(lda.class))
lda.cm

##### QDA model #####
kfold.qda <- train(df.result ~ swelo+sbelo+slength,
                  data=fsdf, subset=fsdt, method ="qda",
                  trControl = train.control)
print(kfold.qda)

#QDA Confusion matrix#
qda.fit=qda(df.result ~ swelo+sbelo+slength, data=fsdf, subset=fsdt)
qda.class =predict(qda.fit, fstest)$class
qda.cm = confusionMatrix(fstest$df.result,as.factor(qda.class))
qda.cm

##### Logistic model #####

```

```

kfold.logistic <- train(df.result ~ swelo+sbelo+slength,
                        data=fsdf, subset=fsdt, method ="multinom",
                        trControl = train.control)
print(kfold.logistic)

#Multinomial Logistic Regression of phase I#
logitmod <- multinom(df.result~swelo+sbelo+slength,data=fsdf, subset=fsdt)
logitmod.sum <- summary(logitmod)
logitmod.z <- logitmod.sum$coefficients/logitmod.sum$standard.errors
logitmod.z
# 2-tailed z test
logitmod.p <- (1 - pnorm(abs(logitmod.z), 0, 1)) * 2
logitmod.p
## extract the coefficients from the model and exponentiate
logitmod.OR=exp(coef(logitmod))
logitmod.OR

#Multinomial Logistic regression model confusion matrix#####

logitmod.pred <-predict(logitmod, newdata = fstest.x, type="class")
logitmod.cm <- confusionMatrix(fstest$df.result,as.factor(logitmod.pred))
logitmod.cm

##### KNN model (Kfold and Confusion matrix) #####
kfold.knn <- train(df.result ~ swelo+sbelo+slength,
                  data=fsdf, subset=fsdt, method ="knn",
                  trControl = train.control)

print(kfold.knn)

#KNN 5
knn5.pred=knn(fstrain.x, fstest.x, fstrain$df.result, k=5)
knn5.cm = confusionMatrix(fstest$df.result,as.factor(knn5.pred))
knn5.cm
#KNN 7
knn7.pred=knn(fstrain.x, fstest.x, fstrain$df.result, k=7)
knn7.cm = confusionMatrix(fstest$df.result,as.factor(knn7.pred))
knn7.cm
#KNN 9
knn9.pred=knn(fstrain.x, fstest.x, fstrain$df.result, k=9)
knn9.cm = confusionMatrix(fstest$df.result,as.factor(knn9.pred))
knn9.cm

# accuracies
lda.accuracy<-lda.cm$overall
qda.accuracy<-qda.cm$overall
logitmod.accuracy<-logitmod.cm$overall
knn5.accuracy<-knn5.cm$overall
knn7.accuracy<-knn7.cm$overall
knn9.accuracy<-knn9.cm$overall

### Performance measure for balanced data ###
table(df.on$result)
q1<-df.on[which(df.on$result=="draw"),]
q2<-df.on[which(df.on$result=="white"),]
q3<-df.on[which(df.on$result=="black"),]
q4<-q2[1:950,]
q5<-q3[1:950,]
df.on.bal<-rbind(q1,q4,q5)
dim(df.on.bal)
table(df.on.bal$result)
plot(df.on.bal)
hist(df.on.bal$welo)
hist(df.on.bal$belo)
hist(df.on.bal$length)

#standardise
bwelo<- (df.on.bal$welo - mean(df.on.bal$welo))/sd(df.on.bal$welo)

```

```

bbelo<- (df.on.bal$belo - mean(df.on.bal$belo))/sd(df.on.bal$belo)
blength<- (df.on.bal$length - mean(df.on.bal$length))/sd(df.on.bal$length)
df.on.bal.s <- data.frame(df.on.bal$result, bwelo, bbelo, blength)
names(df.on.bal.s)<-c("df.result", "swelo", "sbelo", "slength")

test.pred1= predict(lda.fit,newdata=df.on.bal.s[,2:4],type='class')
test.cm1 = confusionMatrix(df.on.bal.s$df.result,as.factor(test.pred1$class))

test.pred2= predict(qda.fit,newdata=df.on.bal.s[,2:4],type='class')
test.cm2 = confusionMatrix(df.on.bal.s$df.result,as.factor(test.pred2$class))

test.pred3= predict(logitmod,newdata=df.on.bal.s[,2:4],type='class')
test.cm3 = confusionMatrix(df.on.bal.s$df.result,as.factor(test.pred3))

test.pred4= knn(fstrain.x, df.on.bal.s[,2:4], fstrain$df.result, k=5)
test.cm4 = confusionMatrix(df.on.bal.s$df.result,as.factor(test.pred4))

test.pred5= knn(fstrain.x, df.on.bal.s[,2:4], fstrain$df.result, k=7)
test.cm5 = confusionMatrix(df.on.bal.s$df.result,as.factor(test.pred5))

test.pred6= knn(fstrain.x, df.on.bal.s[,2:4], fstrain$df.result, k=9)
test.cm6 = confusionMatrix(df.on.bal.s$df.result,as.factor(test.pred6))

test.cm1 #LDA
lda.cm
test.cm2 #QDA
qda.cm
test.cm3 #MLR
logitmod.cm
test.cm4 #KNN5
knn5.cm
test.cm5 #KNN7
knn7.cm
test.cm6 #KNN9
knn9.cm

#####
##### PHASE II #####
#####

## reading the data ##
phase2 <- read.csv("phase2.csv")

#### For summary statistics and histograms ####
phase2.df<-data.frame(phase2[,c(3:14)])
colnames(phase2.df)<-c("result", "welo", "belo", "length",
                    "w.inaccuracy", "w.mistakes", "w.blunders", "w.acl",
                    "b.inaccuracy", "b.mistakes", "b.blunders", "b.acl")
phase2.df$result[phase2.df$result == 36526] <- "white"
phase2.df$result[phase2.df$result == "0-1"] <- "black"
phase2.df$result[phase2.df$result == "1/2-1/2"] <- "draw"
phase2.df<-na.omit(phase2.df)
phase2.df$result <- as.factor(phase2.df$result)

## Preprocessing the data for standardisation##
df.p2<-data.frame(phase2[,c(3:14)])
colnames(df.p2)<-c("result", "welo", "belo", "length",
                    "w.inaccuracy", "w.mistakes", "w.blunders", "w.acl",
                    "b.inaccuracy", "b.mistakes", "b.blunders", "b.acl")
df.p2$result[df.p2$result == 36526] <- "white"
df.p2$result[df.p2$result == "0-1"] <- "black"
df.p2$result[df.p2$result == "1/2-1/2"] <- "draw"
df.p2<-na.omit(df.p2)
df.p2$result <- as.factor(df.p2$result)

#standardised
df.p2$welo<- (df.p2$welo - mean(df.p2$welo))/sd(df.p2$welo)

```



```

kfold.logistic.p2.half <- train(result ~ welo+belo+length,
                                data=df.p2, subset=dt.p2, method ="multinom",
                                trControl = train.control)
kfold.knn.p2.half <- train(result ~ welo+belo+length,
                            data=df.p2, subset=dt.p2, method ="knn",
                            trControl = train.control)

# Summarize the results
print(kfold.lda.p2.half)
print(kfold.qda.p2.half)
print(kfold.logistic.p2.half)
print(kfold.knn.p2.half)

### LOOCV #####
# Define training control
train.control.loocv <- trainControl(method = "LOOCV", classProbs = TRUE)
# Train the half models
loocv.lda.p2.half <- train(result ~ welo+belo+length,
                            data=df.p2, subset=dt.p2, method ="lda",
                            trControl = train.control.loocv)
loocv.qda.p2.half <- train(result ~ welo+belo+length,
                            data=df.p2, subset=dt.p2, method ="qda",
                            trControl = train.control.loocv)
loocv.logistic.p2.half <- train(result ~ welo+belo+length,
                                data=df.p2, subset=dt.p2, method ="multinom",
                                trControl = train.control.loocv)
loocv.knn.p2.half <- train(result ~ welo+belo+length,
                            data=df.p2, subset=dt.p2, method ="knn",
                            trControl = train.control.loocv)

# Summarize the results
print(loocv.lda.p2.half)
print(loocv.qda.p2.half)
print(loocv.logistic.p2.half)
print(loocv.knn.p2.half)

#### Train the full models (11 predictors) #####

##### Confusion Matrices #####
lda.full.p2.fit=lda(result ~ welo+belo+length
                    +w.inaccuracy+w.mistakes+w.blunders+w.acl
                    +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                    data=df.p2, subset=dt.p2)
lda.full.p2.class= predict(lda.full.p2.fit, testing)$class
lda.full.p2.cm = confusionMatrix(testing$result,as.factor(lda.full.p2.class))
lda.full.p2.cm

qda.full.p2.fit=qda(result ~ welo+belo+length
                    +w.inaccuracy+w.mistakes+w.blunders+w.acl
                    +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                    data=df.p2, subset=dt.p2)
qda.full.p2.class =predict(qda.full.p2.fit, testing)$class
qda.full.p2.cm = confusionMatrix(testing$result,as.factor(qda.full.p2.class))
qda.full.p2.cm

logit.full.p2.fit <- multinom(result ~ welo+belo+length
                              +w.inaccuracy+w.mistakes+w.blunders+w.acl
                              +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                              data=df.p2, subset=dt.p2)
logit.full.p2.pred <-predict(logit.full.p2.fit, newdata = testing.x, type="class")
logit.full.p2.cm <- confusionMatrix(testing$result,as.factor(logit.full.p2.pred))
logit.full.p2.cm

knn5.full.p2.pred=knn(training.x, testing.x, training$result, k=5)
knn5.full.p2.cm = confusionMatrix(testing$result,as.factor(knn5.full.p2.pred))
knn5.full.p2.cm

knn7.full.p2.pred=knn(training.x, testing.x, training$result, k=7)
knn7.full.p2.cm = confusionMatrix(testing$result,as.factor(knn7.full.p2.pred))

```



```

knn7.full.p2.cm

knn9.full.p2.pred=knn(training.x, testing.x, training$result, k=9)
knn9.full.p2.cm = confusionMatrix(testing$result,as.factor(knn9.full.p2.pred))
knn9.full.p2.cm

### K FOLD #####
kfold.lda.p2.full <- train(result ~ welo+belo+length
                           +w.inaccuracy+w.mistakes+w.blunders+w.acl
                           +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                           data=df.p2, subset=dt.p2, method ="lda",
                           trControl = train.control)
kfold.qda.p2.full <- train(result ~ welo+belo+length
                           +w.inaccuracy+w.mistakes+w.blunders+w.acl
                           +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                           data=df.p2, subset=dt.p2, method ="qda",
                           trControl = train.control)
kfold.logistic.p2.full <- train(result ~ welo+belo+length
                                +w.inaccuracy+w.mistakes+w.blunders+w.acl
                                +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                                data=df.p2, subset=dt.p2, method ="multinom",
                                trControl = train.control)
kfold.knn.p2.full <- train(result ~ welo+belo+length
                           +w.inaccuracy+w.mistakes+w.blunders+w.acl
                           +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                           data=df.p2, subset=dt.p2, method ="knn",
                           trControl = train.control)

# Summarize the results
print(kfold.lda.p2.full)
print(kfold.qda.p2.full)
print(kfold.logistic.p2.full)
print(kfold.knn.p2.full)

##### LOOCV #####
loocv.lda.p2.full <- train(result ~ welo+belo+length
                           +w.inaccuracy+w.mistakes+w.blunders+w.acl
                           +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                           data=df.p2, subset=dt.p2, method ="lda",
                           trControl = train.control.loocv)
loocv.qda.p2.full <- train(result ~ welo+belo+length
                           +w.inaccuracy+w.mistakes+w.blunders+w.acl
                           +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                           data=df.p2, subset=dt.p2, method ="qda",
                           trControl = train.control.loocv)
loocv.logistic.p2.full <- train(result ~ welo+belo+length
                                +w.inaccuracy+w.mistakes+w.blunders+w.acl
                                +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                                data=df.p2, subset=dt.p2, method ="multinom",
                                trControl = train.control.loocv)
loocv.knn.p2.full <- train(result ~ welo+belo+length
                           +w.inaccuracy+w.mistakes+w.blunders+w.acl
                           +b.inaccuracy+b.mistakes+b.blunders+b.acl,
                           data=df.p2, subset=dt.p2, method ="knn",
                           trControl = train.control.loocv)

# Summarize the results
print(loocv.lda.p2.full)
print(loocv.qda.p2.full)
print(loocv.logistic.p2.full)
print(loocv.knn.p2.full)

#Checking significant models in phase II
logit.test <- multinom(result ~ welo+belo+length
                       +w.inaccuracy+w.mistakes+w.blunders+w.acl
                       +b.inaccuracy+b.mistakes+b.blunders+b.acl, data = df.p2)
logit.sum.t <- summary(logit.test)
logit.z.t <- logit.sum.t$coefficients/logit.sum.t$standard.errors
logit.z.t

```

```

# 2-tailed z test
p.t <- (1 - pnorm(abs(logit.z.t), 0, 1)) * 2
p.t

# Train the significant models under CM #####
lda.sig.p2.fit=lda(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2)
lda.sig.p2.class= predict(lda.sig.p2.fit, testing[,c(8,12)])$class
lda.sig.p2.cm = confusionMatrix(testing$result,as.factor(lda.sig.p2.class))
lda.sig.p2.cm

qda.sig.p2.fit=qda(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2)
qda.sig.p2.class =predict(qda.sig.p2.fit, testing[,c(8,12)])$class
qda.sig.p2.cm = confusionMatrix(testing$result,as.factor(qda.sig.p2.class))
qda.sig.p2.cm

logit.sig.p2.fit <- multinom(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2)
logit.sig.p2.pred <-predict(logit.sig.p2.fit, newdata = testing.x[,c(7,11)],
  type="class")
logit.sig.p2.cm <- confusionMatrix(testing$result,as.factor(logit.sig.p2.pred))
logit.sig.p2.cm

knn5.sig.p2.pred=knn(training.x[,c(7,11)], testing.x[,c(7,11)], training$result, k=5)
knn5.sig.p2.cm = confusionMatrix(testing$result,as.factor(knn5.sig.p2.pred))
knn5.sig.p2.cm

knn7.sig.p2.pred=knn(training.x[,c(7,11)], testing.x[,c(7,11)], training$result, k=7)
knn7.sig.p2.cm = confusionMatrix(testing$result,as.factor(knn7.sig.p2.pred))
knn7.sig.p2.cm

knn9.sig.p2.pred=knn(training.x[,c(7,11)], testing.x[,c(7,11)], training$result, k=9)
knn9.sig.p2.cm = confusionMatrix(testing$result,as.factor(knn9.sig.p2.pred))
knn9.sig.p2.cm

# Train the significant models under KFOLD #####
kfold.lda.p2.sig <- train(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2, method ="lda",
  trControl = train.control)
kfold.qda.p2.sig <- train(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2, method ="qda",
  trControl = train.control)
kfold.logistic.p2.sig <- train(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2, method ="multinom",
  trControl = train.control)
kfold.knn.p2.sig <- train(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2, method ="knn",
  trControl = train.control)

# Train the significant models under LOOCV
loocv.lda.p2.sig <- train(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2, method ="lda",
  trControl = train.control.loocv)
loocv.qda.p2.sig <- train(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2, method ="qda",
  trControl = train.control.loocv)
loocv.logistic.p2.sig <- train(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2, method ="multinom",
  trControl = train.control.loocv)
loocv.knn.p2.sig <- train(result ~ w.acl+b.acl,
  data=df.p2, subset=dt.p2, method ="knn",
  trControl = train.control.loocv)

#####
#### PHASE III ####
#####

```

```

#QDA model was the best model with 11 predictors

#generating predictions
predictions.p3.11 <- qda.full.p2.class

#KNN (k=9) model was the best model with 3 predictors

#generating predictions
predictions.p3.3 <-knn9.half.p2.pred

p2.train = phase2.df[dt.p2,]
p2.test = phase2.df[-dt.p2,]

####Finding the missclassifications ####
### 11 predictors
hits11<-phase2.df[predictions.p3.11 == p2.test$result,]
hits11
misses11<-phase2.df[predictions.p3.11 != p2.test$result,]
misses11

### 3 predictors
hits3<-phase2.df[predictions.p3.3 == p2.test$result,1:4]
hits3
misses3<-phase2.df[predictions.p3.3 != p2.test$result,1:4]
misses3

# plotting the hits and misses
plot(misses11)
plot(hits11)

plot(misses3)
plot(hits3)

### FOR FULL MODEL FROM PHASE I
# USING LOGITMOD/KNN7 AS IT HAD THE BEST ACCURACY FOR CM/kfold

predictions.p1<-knn7.pred
hits.p1<-df.on[predictions.p1 == fstest$df.result,1:4]
misses.p1<-df.on[predictions.p1 != fstest$df.result,1:4]

#####
my.col=c(261,142)
hits3$status<-"hit"
misses3$status<-"miss"

classif.3<-rbind(hits3,misses3)
classif.3$status<-as.factor(classif.3$status)
str(classif.3)

plot(classif.3[1:4], col=my.col[classif.3$status]) #purple are misses

par(mfrow=c(1,2))
plot(classif.3$welo, col=my.col[classif.3$status],ylab="White ELO")
plot(classif.3$belo, col=my.col[classif.3$status],ylab="Black ELO")
par(mfrow=c(1,1))
plot(classif.3$length, col=my.col[classif.3$status],ylab="Length")

hits11$status<-"hit"
misses11$status<-"miss"

classif.11<-rbind(hits11,misses11)
classif.11$status<-as.factor(classif.11$status)
str(classif.11)

plot(classif.11[1:12], col=my.col[classif.11$status])
#purple are misses

```

```

par(mfrow=c(1,2))
plot(classif.11$welo, col=my.col[classif.11$status],ylab="White ELO")
plot(classif.11$belo, col=my.col[classif.11$status],ylab="Black ELO")
par(mfrow=c(1,1))
plot(classif.11$length, col=my.col[classif.11$status],ylab="Length")
par(mfrow=c(2,2))
plot(classif.11$w.inaccuracy, col=my.col[classif.11$status],ylab="White inaccuracies")
plot(classif.11$w.mistakes, col=my.col[classif.11$status],ylab="White mistakes")
plot(classif.11$w.blunders, col=my.col[classif.11$status],ylab="White blunders")
plot(classif.11$w.acl, col=my.col[classif.11$status],ylab="White avg. centipawn loss")
par(mfrow=c(2,2))
plot(classif.11$b.inaccuracy, col=my.col[classif.11$status],ylab="Black inaccuracies")
plot(classif.11$b.mistakes, col=my.col[classif.11$status],ylab="Black mistakes")
plot(classif.11$b.blunders, col=my.col[classif.11$status],ylab="Black blunders")
plot(classif.11$b.acl, col=my.col[classif.11$status],ylab="Black avg. centipawn loss")
par(mfrow=c(1,1))

##### Results section #####

barplot(OTB.table)
hist(df$welo, main = "Histogram of White player ELO rating (OTB games)",xlab="welo")
hist(df$belo,main = "Histogram of Black player ELO rating (OTB games)",xlab="belo")
hist(df$length,main = "Histogram of the length of the games (OTB games)",xlab="length")

barplot(online.table)
hist(df,on$welo, main = "Histogram of White player ELO rating (Online
games)",xlab="welo")
hist(df,on$belo,main = "Histogram of Black player ELO rating (Online games)",xlab="belo")
hist(df,on$length,main = "Histogram of the length of the games (Online
games)",xlab="length")

##### CONFUSION MATRIX #####
lda.half.p2.cm
lda.full.p2.cm

qda.half.p2.cm
qda.full.p2.cm

logit.half.p2.cm
logit.full.p2.cm

knn5.half.p2.cm
knn5.full.p2.cm

knn7.half.p2.cm
knn7.full.p2.cm

knn9.half.p2.cm
knn9.full.p2.cm

##### KFOLD #####
kfold.lda.p2.half
kfold.lda.p2.full

kfold.qda.p2.half
kfold.qda.p2.full

kfold.logistic.p2.half
kfold.logistic.p2.full

kfold.knn.p2.half
kfold.knn.p2.full

##### LOOCV #####
loocv.lda.p2.half
loocv.lda.p2.full

```

```

loocv.qda.p2.half
loocv.qda.p2.full

loocv.logistic.p2.half
loocv.logistic.p2.full

loocv.knn.p2.half
loocv.knn.p2.full

# for the significant models --- RESULTS
print(lda.sig.p2.cm)
print(qda.sig.p2.cm)
print(logit.sig.p2.cm)
print(knn5.sig.p2.cm)
print(knn7.sig.p2.cm)
print(knn9.sig.p2.cm)

print(kfold.lda.p2.sig)
print(kfold.qda.p2.sig)
print(kfold.logistic.p2.sig)
print(kfold.knn.p2.sig)

print(loocv.lda.p2.sig)
print(loocv.qda.p2.sig)
print(loocv.logistic.p2.sig)
print(loocv.knn.p2.sig)

#####
##### RESULT summaries and plots #####
#####

####PHASE I #####
##OTB
#OTB.sum<-summary(df)
#OTB.dim<-dim(df)
#OTB.table<-table(df$result)
#OTB.prop<-prop.table(table(df$result))
##online
#online.sum<-summary(df.on)
#online.dim<-dim(df.on)
#online.table<-table(df.on$result)
#online.prop<-prop.table(table(df.on$result))

#OTB
OTB.sum
OTB.dim
OTB.table
OTB.prop
#online
online.sum
online.dim
online.table
online.prop

barplot(OTB.table)
hist(df$welo, main = "Histogram of White player ELO rating (OTB games)",xlab="welo")
hist(df$belo,main = "Histogram of Black player ELO rating (OTB games)",xlab="belo")
hist(df$length,main = "Histogram of the length of the games (OTB games)",xlab="length")

barplot(online.table)
hist(df.on$welo, main = "Histogram of White player ELO rating (Online
games)",xlab="welo")
hist(df.on$belo,main = "Histogram of Black player ELO rating (Online games)",xlab="belo")
hist(df.on$length,main = "Histogram of the length of the games (Online
games)",xlab="length")

```

```

#### LDA ####
# confusion matrix #
lda.cm
#kfold#
kfold.lda

#### QDA ####
# confusion matrix #
qda.cm
#kfold#
kfold.qda

#Logistic#
# confusion matrix #
logitmod.sum
logitmod.cm
#kfold#
kfold.logistic

#KNN#
#confusion matrix #
knn5.cm
knn7.cm
knn9.cm
#kfold#
kfold.knn

### accuracy plots #####
accuracies.<-c(lda.accuracy[1],
              qda.accuracy[1],
              logitmod.accuracy[1],
              knn5.accuracy[1],
              knn7.accuracy[1],
              knn9.accuracy[1])
kappas.<-c(lda.accuracy[2],
           qda.accuracy[2],
           logitmod.accuracy[2],
           knn5.accuracy[2],
           knn7.accuracy[2],
           knn9.accuracy[2])
method<-c("LDA","QDA","logistic", "knn5","knn7","knn9")
acctab.<-data.frame(method,accuracies.,kappas.)
plot.acc.p.<-plot(as.factor(acctab.$method),acctab.$accuracies.,
                 ylim=c(0.4,1),
                 main="Accuracies comparison plot (Confusion Matrix)",xlab="Methods",
                 ylab="Accuracies")
plot.acc.k.<-plot(as.factor(acctab.$method),acctab.$kappas.,
                 ylim=c(0,1),
                 main="Kappa Statistic comparison plot (Confusion
Matrix)",xlab="Methods", ylab="Kappa Statistic")

accuracies0<-c(kfold.lda$results$Accuracy,
               kfold.qda$results$Accuracy,
               kfold.logistic$results$Accuracy[c(1)],
               kfold.knn$results$Accuracy)
kappas0<-c(kfold.lda$results$Kappa,
           kfold.qda$results$Kappa,
           kfold.logistic$results$Kappa[c(1)],
           kfold.knn$results$Kappa)
method<-c("LDA","QDA","logistic", "knn5","knn7","knn9")
acctab0<-data.frame(method,accuracies0,kappas0)
plot.acc.pl<-plot(as.factor(acctab0$method),acctab0$accuracies0,
                  ylim=c(0.5,1),
                  main="Accuracies comparison plot (K-fold CV)",xlab="Methods",
                  ylab="Accuracies")
plot.acc.kl<-plot(as.factor(acctab0$method),acctab0$kappas0,
                  ylim=c(0,1),

```

```

                                main="Kappa Statistic comparison plot (K-fold CV)",xlab="Methods",
ylab="Kappa Statistic")

####PHASE II #####

#df.p2.sum<-summary(phase2.df)
#df.p2.dim<-dim(phase2.df)
#df.p2.table<-table(phase2.df$result)
#df.p2.prop<-prop.table(table(phase2.df$result))

df.p2.sum
df.p2.dim
df.p2.table
df.p2.prop

barplot(table(df.p2$result))
hist(df.p2$welo, main = "Histogram of White player ELO rating (Small scale)",xlab="welo")
hist(df.p2$belo,main = "Histogram of Black player ELO rating (Small scale)",xlab="belo")
hist(df.p2$length,main = "Histogram of the length of the games (Small
scale)",xlab="length")
hist(df.p2$w.inaccuracy,main = "Histogram of the white player's inaccuracies (Small
scale)",xlab="inaccuracies")
hist(df.p2$w.mistakes,main = "Histogram of the white player's mistakes (Small
scale)",xlab="mistakes")
hist(df.p2$w.blunders,main = "Histogram of the white player's blunders (Small
scale)",xlab="blunders")
hist(df.p2$w.acl,main = "Histogram of the white player's average centipawn loss (Small
scale)",xlab="average centipawn loss")
hist(df.p2$b.inaccuracy,main = "Histogram of the black player's inaccuracies (Small
scale)",xlab="inaccuracies")
hist(df.p2$b.mistakes,main = "Histogram of the black player's mistakes (Small
scale)",xlab="mistakes")
hist(df.p2$b.blunders,main = "Histogram of the black player's blunders (Small
scale)",xlab="blunders")
hist(df.p2$b.acl,main = "Histogram of the black player's average centipawn loss (Small
scale)",xlab="average centipawn loss")

#### Confusion matrix ####
lda.half.p2.cm
lda.full.p2.cm

qda.half.p2.cm
qda.full.p2.cm

logit.half.p2.cm
logit.full.p2.cm

knn5.half.p2.cm
knn5.full.p2.cm

knn7.half.p2.cm
knn7.full.p2.cm

knn9.half.p2.cm
knn9.full.p2.cm

##### KFOLD #####
kfold.lda.p2.half
kfold.lda.p2.full

kfold.qda.p2.half
kfold.qda.p2.full

kfold.logistic.p2.half
kfold.logistic.p2.full

```

```

kfold.knn.p2.half
kfold.knn.p2.full

##### LOOCV #####
loocv.lda.p2.half
loocv.lda.p2.full

loocv.qda.p2.half
loocv.qda.p2.full

loocv.logistic.p2.half
loocv.logistic.p2.full

loocv.knn.p2.half
loocv.knn.p2.full

### accuracy plots #####

### CONFUSION MATRIX ###
#half#
accuracies7<-c(lda.half.p2.cm$overall[1],
               qda.half.p2.cm$overall[1],
               logit.half.p2.cm$overall[1],
               knn5.half.p2.cm$overall[1],
               knn7.half.p2.cm$overall[1],
               knn9.half.p2.cm$overall[1])
kappas7<-c(lda.half.p2.cm$overall[2],
            qda.half.p2.cm$overall[2],
            logit.half.p2.cm$overall[2],
            knn5.half.p2.cm$overall[2],
            knn7.half.p2.cm$overall[2],
            knn9.half.p2.cm$overall[2])
method<-c("LDA","QDA","logistic", "knn5","knn7","knn9")
acctab7<-data.frame(method,accuracies7,kappas7)
plot.acc.p2.cm.half<-plot(as.factor(acctab7$method),acctab7$accuracies7,
                          ylim=c(0.5,1),
                          main="3 predictors Accuracies comparison plot
(CM)",xlab="Methods", ylab="Accuracies")
plot.k.p2.cm.half<-plot(as.factor(acctab7$method),acctab7$kappas7,
                          ylim=c(0,1),
                          main="3 predictors Kappa Comparison plot (CM)",xlab="Methods",
                          ylab="Kappa Statistic")

## Full ##
accuracies8<-c(lda.full.p2.cm$overall[1],
               qda.full.p2.cm$overall[1],
               logit.full.p2.cm$overall[1],
               knn5.full.p2.cm$overall[1],
               knn7.full.p2.cm$overall[1],
               knn9.full.p2.cm$overall[1])
kappas8<-c(lda.full.p2.cm$overall[2],
            qda.full.p2.cm$overall[2],
            logit.full.p2.cm$overall[2],
            knn5.full.p2.cm$overall[2],
            knn7.full.p2.cm$overall[2],
            knn9.full.p2.cm$overall[2])
method<-c("LDA","QDA","logistic", "knn5","knn7","knn9")
acctab8<-data.frame(method,accuracies8,kappas8)
plot.acc.p2.cm.full<-plot(as.factor(acctab8$method),acctab8$accuracies8,
                          ylim=c(0.5,1),
                          main="11 predictors Accuracies comparison plot
(CM)",xlab="Methods", ylab="Accuracies")
plot.k.p2.cm.full<-plot(as.factor(acctab8$method),acctab8$kappas8,
                          ylim=c(0,1),
                          main="11 predictors Kappa Comparison plot (CM)",xlab="Methods",
                          ylab="Kappa Statistic")

```



```

###      K FOLD ####
#HALF
accuracies1<-c(kfold.lda.p2.half$results$Accuracy,
               kfold.qda.p2.half$results$Accuracy,
               kfold.logistic.p2.half$results$Accuracy[c(1)],
               kfold.knn.p2.half$results$Accuracy)
kappas1<-c(kfold.lda.p2.half$results$Kappa,
           kfold.qda.p2.half$results$Kappa,
           kfold.logistic.p2.half$results$Kappa[c(1)],
           kfold.knn.p2.half$results$Kappa)
method<-c("LDA","QDA","logistic", "knn5","knn7","knn9")
acctab1<-data.frame(method,accuracies1,kappas1)
plot.acc.p2.kfold.half<-plot(as.factor(acctab1$method),acctab1$accuracies1,
                             ylim=c(0.5,1),
                             main="3 predictors Accuracies comparison plot (K-fold
CV)",xlab="Methods", ylab="Accuracies")
plot.k.p2.kfold.half<-plot(as.factor(acctab1$method),acctab1$kappas1,
                             ylim=c(0,1),
                             main="3 predictors Kappa comparison plot (K-fold
CV)",xlab="Methods", ylab="Kappa Statistic")

# FULL
accuracies2<-c(kfold.lda.p2.full$results$Accuracy,
               kfold.qda.p2.full$results$Accuracy,
               kfold.logistic.p2.full$results$Accuracy[c(1)],
               kfold.knn.p2.full$results$Accuracy)
kappas2<-c(kfold.lda.p2.full$results$Kappa,
           kfold.qda.p2.full$results$Kappa,
           kfold.logistic.p2.full$results$Kappa[c(1)],
           kfold.knn.p2.full$results$Kappa)
method<-c("LDA","QDA","logistic", "knn5","knn7","knn9")
acctab2<-data.frame(method,accuracies2,kappas2)
plot.acc.p2.kfold.full<-plot(as.factor(acctab2$method),acctab2$accuracies2,
                             ylim=c(0.5,1),
                             main="11 predictors Accuracies comparison plot (K-fold
CV)",xlab="Methods", ylab="Accuracies")
plot.k.p2.kfold.full<-plot(as.factor(acctab2$method),acctab2$kappas2,
                             ylim=c(0,1),
                             main="11 predictors Kappa comparison plot (K-fold
CV)",xlab="Methods", ylab="Kappa Statistic")

###      LOOCV ####
#HALF
accuracies4<-c(loocv.lda.p2.half$results$Accuracy,
               loocv.qda.p2.half$results$Accuracy,
               loocv.logistic.p2.half$results$Accuracy[c(1)],
               loocv.knn.p2.half$results$Accuracy)
kappas4<-c(loocv.lda.p2.half$results$Kappa,
           loocv.qda.p2.half$results$Kappa,
           loocv.logistic.p2.half$results$Kappa[c(1)],
           loocv.knn.p2.half$results$Kappa)
method<-c("LDA","QDA","logistic", "knn5","knn7","knn9")
acctab4<-data.frame(method,accuracies4,kappas4)
plot.acc.p2.loocv.half<-plot(as.factor(acctab4$method),acctab4$accuracies4,
                             ylim=c(0.5,1),
                             main="3 predictors Accuracies comparison plot
(LOOCV)",xlab="Methods", ylab="Accuracies")
plot.k.p2.loocv.half<-plot(as.factor(acctab4$method),acctab4$kappas4,
                             ylim=c(0,1),
                             main="3 predictors Kappa comparison plot
(LOOCV)",xlab="Methods", ylab="Kappa Statistic")

# FULL
accuracies5<-c(loocv.lda.p2.full$results$Accuracy,
               loocv.qda.p2.full$results$Accuracy,
               loocv.logistic.p2.full$results$Accuracy[c(1)],
               loocv.knn.p2.full$results$Accuracy)

```

```

kappas5<-c(loocv.lda.p2.full$results$Kappa,
           loocv.qda.p2.full$results$Kappa,
           loocv.logistic.p2.full$results$Kappa[c(1)],
           loocv.knn.p2.full$results$Kappa)
method<-c("LDA","QDA","logistic", "knn5","knn7","knn9")
acctab5<-data.frame(method,accuracies5,kappas5)
plot.acc.p2.loocv.full<-plot(as.factor(acctab5$method),acctab5$accuracies5,
                             ylim=c(0.5,1),
                             main="11 predictors Accuracies comparison plot
(LOOCV)",xlab="Methods", ylab="Accuracies")
plot.k.p2.loocv.full<-plot(as.factor(acctab5$method),acctab5$kappas5,
                             ylim=c(0,1),
                             main="11 predictors Kappa Comparison plot
(LOOCV)",xlab="Methods", ylab="Kappa Statistic")

##### Accuracy CM comparison 3 v 11 #####
plot(x,y,type="n", ylim=c(0.5,1),
      main="Comparison of models under Confusion Matrix",
      xlab="Methods", ylab="Accuracies")
grid()
points(x,accuracies7, pch=3, col=612)
points(x,accuracies8, pch=4, col=261)
legend(4.5,0.6,c("3 predictors","11 predictors"),pch=3:5,col=c(612,261))

##### Accuracy Kfold comparison 3 v 11####
x<-c(1,2,3,4,5,6)
y<-seq(1,6,1)
plot(x,y,type="n", ylim=c(0.5,1),
      main="Comparison of models under K-fold",
      xlab="Methods", ylab="Accuracies")
grid()
points(x,accuracies1, pch=0, col=612)
points(x,accuracies2, pch=1, col=261)
legend(4.5,0.6,c("3 predictors","11 predictors"),pch=0:2,col=c(612,261))

##### Accuracy LOOCV comparison 3 v 11####
x<-c(1,2,3,4,5,6)
y<-seq(1,6,1)
plot(x,y,type="n", ylim=c(0.5,1),
      main="Comparison of models under LOOCV",
      xlab="Methods", ylab="Accuracies")
grid()
points(x,accuracies4, pch=0, col=612)
points(x,accuracies5, pch=1, col=261)
legend(4.5,0.6,c("3 predictors","11 predictors"),pch=0:2,col=c(612,261))

```