INTRODUCTION ABOUT MY ROLE IN THE PROJECT

In my Spring 2023 Big Data course, I took on a chess betting prediction project, diving headfirst into learning from scratch how to set up a Hadoop system on a virtual machine. I familiarized myself with all the necessary components, installed them, and configured the system, which laid the groundwork for the project.

My role was central to the project's data handling. I focused on thoroughly cleaning the dataset of over 6 million chess games to ensure accurate analysis. I then crafted and executed queries that would help extract meaningful patterns for predicting match outcomes. In particular, I analyzed the impact of various chess openings, like the Van't Kruijs Opening, which showed an impressive 99% win rate.

I also oversaw the integration of Spark for real-time processing, which allowed us to enhance prediction accuracy. Through diligent work and continuous learning, I was able to refine the predictive model to achieve nearly 90% accuracy, around 5% higher than existing models. This required meticulous attention to detail and iterative improvements to optimize the system.

Overall, this project was a deep dive into Big Data tools and machine learning techniques. It provided me with invaluable hands-on experience working with cutting-edge technology to tackle a complex problem, reinforcing the importance of effective data management and creative problem-solving in developing predictive models.

Chess Prediction Assistant

Big Data (BUAN/MIS 6346.503)

Final Project

Team:

Rupa Tejaswini Chapalamadugu

Vamsi Reddy Atchi

Surya Anuteja Chitturi

Venkatesh Prabu

Index

Abstract

This report explores the use of Hadoop and Spark in chess betting prediction. Hadoop is used for storing and processing large amounts of historical chess match data, which can be used to train machine learning models. Spark is then used for real-time processing and prediction based on the current state of a chess match. By integrating these technologies, it is possible to create a highly accurate and efficient prediction system for chess betting. The report discusses the benefits of using Hadoop and Spark and emphasizes the importance of responsible gambling.

1. Introduction

Chess betting refers to the act of placing bets on the outcome of a chess match or tournament. The game of chess has been popular for centuries, and its popularity has only increased with the advent of online chess platforms. As a result, chess betting has become a popular form of gambling.

Chess betting prediction is a popular activity that involves predicting the outcome of chess matches or tournaments. With the rise of big data technologies such as Hadoop and Spark, it is possible to use these tools to analyze data and make accurate predictions for chess betting. In this report, we explore the use of Hadoop and Spark in chess betting prediction.

Recently, chess betting has grown in popularity with a large number of aficionados making bets on the outcomes of chess matches. Chess is a complicated strategy game with a long history, making it thrilling and interesting to watch and wager on. The development of big data technologies like Hadoop and Spark has made it possible to utilize these tools to analyze data from previous chess matches and create precise forecasts for upcoming games.

In this report, we explore the use of Hadoop and Spark in chess betting prediction. We begin by discussing the basics of chess betting. We then provide an overview of Hadoop and Spark and explain how these technologies can be used to process and analyze large amounts of historical chess match data. We also discuss the benefits of using these technologies and highlight some of the challenges associated with chess betting prediction.

2. Chess Betting

The act of betting on the outcome of a chess match or tournament is known as chess betting. Chess has been popular for centuries, and online chess platforms have only increased its popularity. Therefore, chess betting has become a popular form of gambling.

The most common form of chess betting is on the outcome of a match, with the most popular bets being on the winner or the draw. However, there are other betting markets available as well, such as the number of moves played, the first player to checkmate, and the total number of points scored in a tournament.

Depending on the country, chess betting may or may not be legal. It is seen as criminal in certain nations yet legal in others. Online chess betting, for instance, is permitted in some states but not others of the United States. Before making any bets, people should make sure that chess betting is permitted in their country of residence. However, the years 2020–21 saw what is generally regarded as the biggest increase in chess interest in the United States since the late Bobby Fischer was the game's most prominent player and encouraged millions of Americans to play in the 1960s and 1970s.

Chess betting involves dangers, just like any other type of gambling. People should only bet what they can afford to lose and be mindful of the possibility for addiction. A chess game's result is also never guaranteed, and even the best players might lose to an underdog.

2.1 Chess Betting Types:

- Bet on the outcome. A classic bet on the victory of an individual athlete or a draw.
- A bet on a double outcome. The coefficient will be much lower than for a win with a zero handicap, but a draw will result in a win rather than a refund. Keep in mind that draws are common in chess, so the potential gain will almost certainly be modest.
- A bet on a zero head start. In this case, bookmakers offer only one option with a handicap in a draw. The bet also implies that one or another participant in the competition will not lose at least. However, in case of a draw, the bet with a zero handicap is calculated by a refund.
- Long-term bets They bet on a specific athlete’s victory not in a specific match, but rather on the overall results of the tournament. In extreme cases, you can hedge and bet on the athlete’s progress rather than his or her victory, for example, in the top two, three, or five.

Chess is a unique sport in which it is extremely difficult to think through and develop a unique strategy based on the unique characteristics of the game.

2.2 Chess Betting Pros and Cons:

Chess, like other sports, has advantages and disadvantages that should be considered when determining how much betting on this type of variety will be beneficial to you:

Cons:

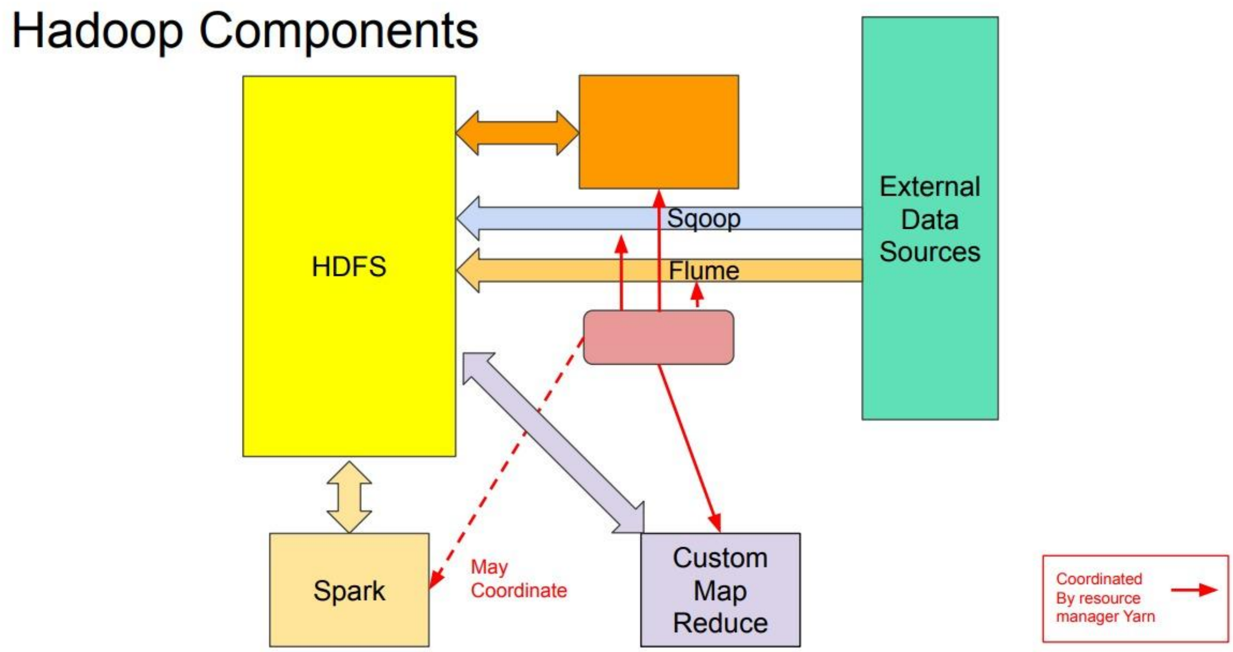
- A small number of competitions, unlike other sports.
- It is difficult to find up-to-date information for most fights.
- High margin from bookmakers.
- Lack of a choice of bets.

Among the benefits of chess betting:

- There are often value bets.
- Many different betting strategies and techniques.
- Leaders rarely change.
- A small number of people who understand sports are betting.

The pros and cons appear to be equal but consider the general impression that the verdict itself suggests. The general principle is straightforward: the greater the competition, the greater the variety. The world champion title matches, the candidate’s tournament, and the biggest stars will undoubtedly be presented along the lines of the leading operators. Chess betting can be extremely profitable if you fully immerse yourself in the game.

3. Hadoop Architecture



The Hadoop architecture is made up of many main components that work together to store, process, and analyze enormous volumes of data in a distributed and fault-tolerant way. These are the components:

3.1. Hadoop Distributed File System (HDFS):

HDFS is a distributed file system that stores huge datasets across numerous nodes in a Hadoop cluster in a scalable and fault-tolerant manner. It is intended to handle huge files and can handle data up to several terabytes in size. NameNode and DataNode are the two types of nodes in HDFS.

3.2. NameNode:

The NameNode is the master node in the HDFS cluster that controls file system metadata such as data block placement, permissions, and user information. It is in charge of managing the directory tree of all files stored in the file system and arranging client access to the data.

3.3. DataNode:

DataNodes are HDFS worker nodes that store and handle the actual data blocks of the files. They interact with the NameNode to report block status and receive instructions on how to replicate, remove, or relocate blocks.

3.4. Yet Another Resource Negotiator (YARN):

YARN is a resource management system that manages the allocation of cluster resources (such as memory and CPU) and schedules jobs to execute on the available resources, allowing several applications to operate on a Hadoop cluster.

YARN is made up of two primary parts:

3.5. ResourceManager:

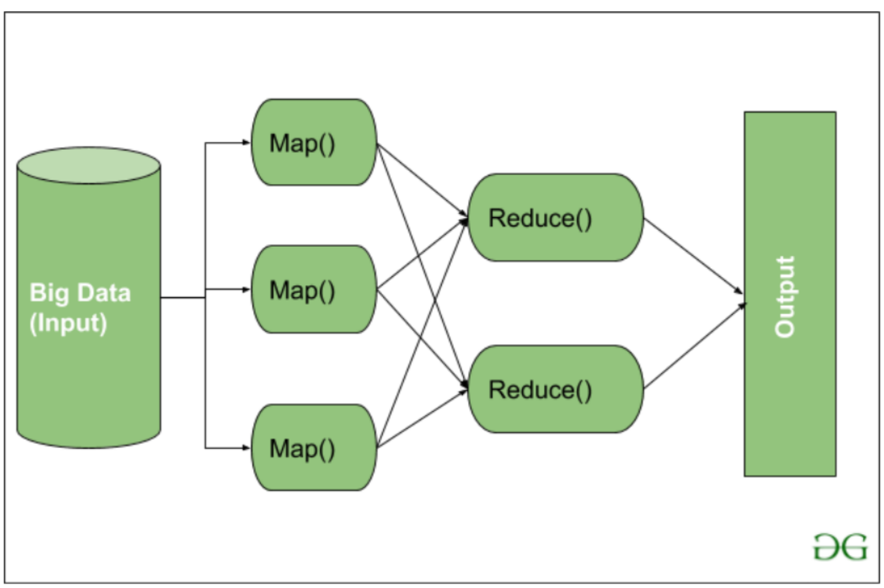
The ResourceManager is the YARN cluster's master node that handles the allocation of cluster resources to applications. It takes application requests and assigns resources to them depending on their requirements and the available resources in the cluster.

3.6. NodeManager:

NodeManagers are YARN worker nodes that manage the resources of a single node in the cluster. They are in charge of performing and monitoring containerized tasks on the node, as well as reporting the status of the tasks to the ResourceManager.

3.7. MapReduce:

MapReduce is a programming model and processing framework that provides a way to process and analyze large datasets in parallel across multiple nodes in a Hadoop cluster.

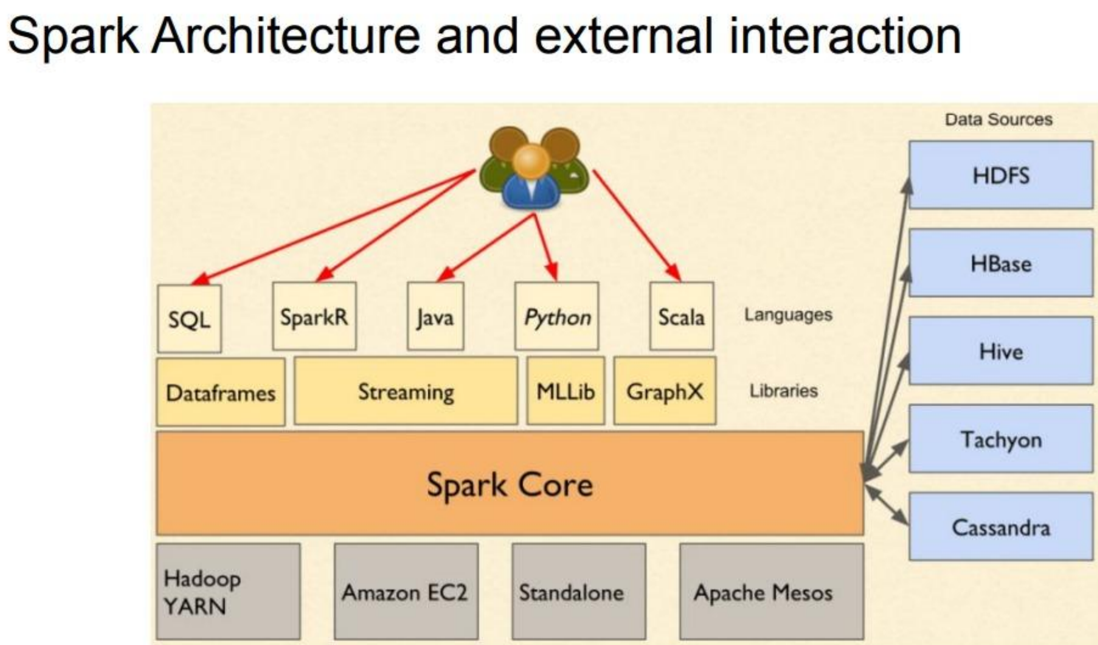


3.8. Hadoop Common:

Hadoop Common is a collection of utilities and libraries that are shared by all Hadoop components. It has a variety of tools, settings, and libraries for activities like as logging, configuration management, security, and network connectivity.

Overall, the Hadoop architecture is intended to be scalable, fault-tolerant, and adaptable, making it appropriate for large-scale data processing jobs in a distributed setting. Organizations can store, manage, and analyze large volumes of data in a cost-effective and efficient manner by using the many components of the Hadoop architecture.

4. Apache Spark Architecture



The Apache Spark architecture is composed of several key components that work together to process large-scale data in a distributed and fault-tolerant manner. These components are:

4.1. Spark Core:

Spark Core is the Spark architecture's basis, providing the fundamental capabilities for distributed task scheduling, memory management, and data input/output. It is in charge of controlling compute and data dissemination among Spark cluster nodes.

4.2. Spark SQL:

Spark SQL is a high-level API for working with structured and semi-structured data sources such as SQL databases and data warehouses. It allows users to do SQL queries on data stored in Spark and supports a variety of data formats such as Parquet, JSON, and Avro.

4.3. Spark Streaming:

Spark Streaming is a near-real-time data processing module that allows users to handle and analyze live data streams. It enables users to create Spark applications that process data streams from Kafka, Flume, and Twitter.

4.4. GraphX:

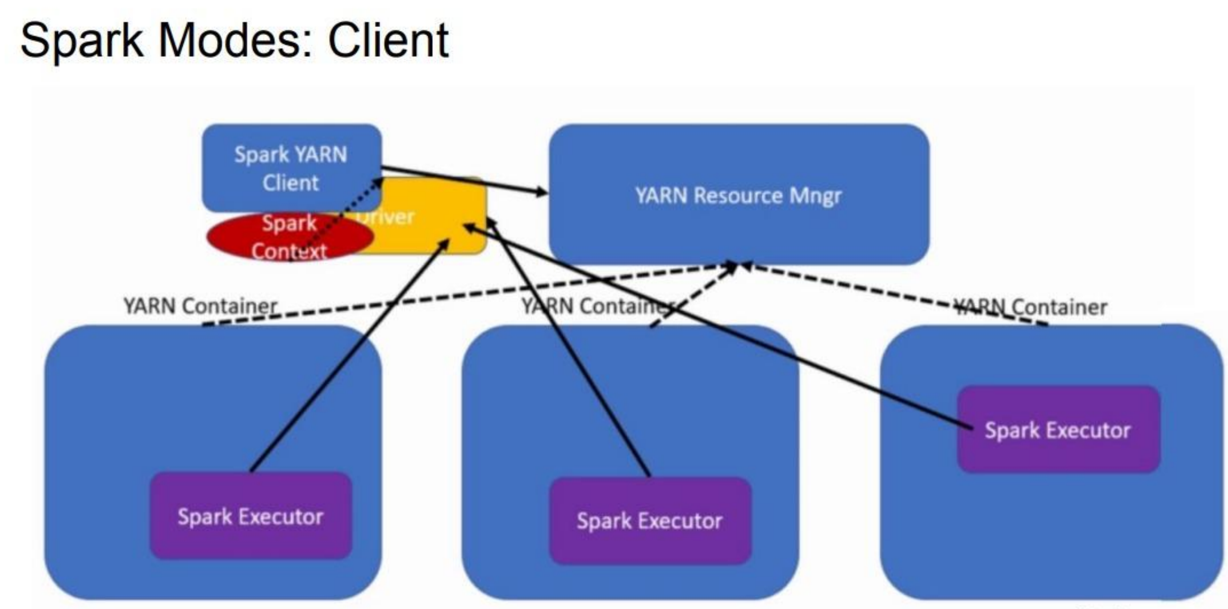
GraphX is a graph processing package that allows users to execute large-scale graph analyses. It includes a distributed graph processing framework as well as a set of graph algorithms for use with social networks, transportation networks, and other forms of graph data.

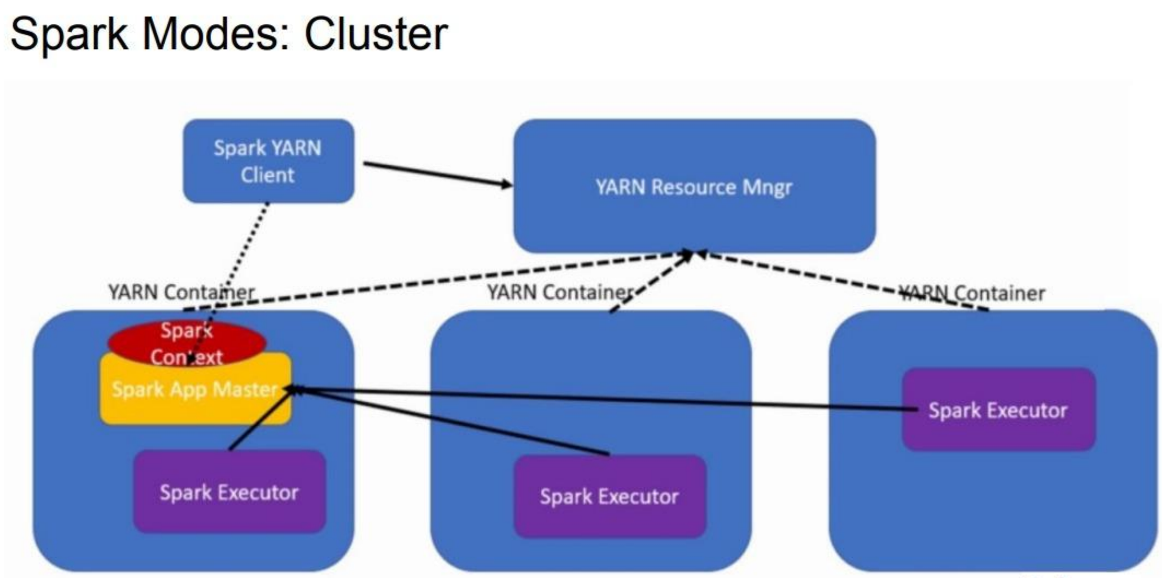
4.5. MLlib:

MLlib is a machine learning module that provides a collection of distributed machine learning algorithms as well as tools for developing and deploying machine learning models at scale. It supports well-known machine learning methods including classification, regression, clustering, and collaborative filtering.

4.6. Cluster Manager:

The Cluster Manager is in charge of managing the resources and tasks distributed across the nodes of a Spark cluster. It decides how jobs are assigned to nodes depending on available resources and guarantee that tasks are done fault-tolerantly.





4.7. Driver Program:

The Driver Program is in charge of coordinating Spark task execution across the cluster. It initializes the Spark context, manages resource allocation, and communicates with the Cluster Manager to plan and monitor Spark job execution.

4.8. Executors:

Executors are in charge of carrying out Spark jobs on particular nodes in the cluster. Each executor is in charge of carrying out several tasks and can cache data in memory to boost efficiency. Executors connect with the Driver Program to receive instructions and provide status updates on their responsibilities.

Overall, the Apache Spark architecture is intended to provide a flexible and scalable framework for large-scale data processing. Organizations may design and deploy data processing and analytics systems that can grow to manage huge volumes of data in a distributed and fault-tolerant way by using the various components of the Spark architecture.

5. Data Structures

Here is a brief description of each attribute:

Event: Game type, which could be a tournament or a casual game.

White: White's ID, which could be the player's username or an anonymous identifier.

Black: Black's ID, which could be the player's username or an anonymous identifier.

Result: The game result, which is either 1-0 if white wins, 0-1 if black wins, or 1/2-1/2 if it is a draw.

UTCDate: The date the game was played in Coordinated Universal Time (UTC).

UTCTime: The time the game was played in Coordinated Universal Time (UTC).

WhiteElo: White's Elo rating, which is a measure of their skill level in chess.

BlackElo: Black's Elo rating, which is a measure of their skill level in chess.

WhiteRatingDiff: The change in White's Elo rating after the game.

BlackRatingDiff: The change in Black's Elo rating after the game.

ECO: The opening code in the Encyclopaedia of Chess Openings (ECO) classification system.

Opening: The name of the opening used in the game.

TimeControl: The time control used in the game, which specifies the amount of time each player has to make their moves.

Termination: The reason the game ended, which could be a checkmate, a resignation, a timeout, or other reasons.

AN: The moves made during the game in Move Text format, which describes each move made by the players.

6. Business Questions

- What are the most popular openings and how do they affect the outcome of the game?

```
>>> df = spark.read.format('csv').option('header','true').load('file:///home/surya/Downloads/cleaneddataset.csv')
>>> opening_counts = df.groupBy("Opening").agg(count("*").alias("GamesPlayed"))
>>> win_rates = df.filter("Result != '1/2-1/2'") \
...     .groupBy("Opening") \
...     .agg(count("*").alias("TotalGames"),
...          count(when(df.Result == "1-0", True)).alias("WhiteWins"),
...          count(when(df.Result == "0-1", True)).alias("BlackWins")) \
...     .withColumn("WhiteWinRate", col("WhiteWins") / col("TotalGames")) \
...     .withColumn("BlackWinRate", col("BlackWins") / col("TotalGames")) \
...     .withColumn("OverallWinRate", (col("WhiteWins") + col("BlackWins")) / col("TotalGames"))
>>> results = opening_counts.join(win_rates, "Opening", "left_outer") \
...     .orderBy(col("GamesPlayed").desc())
>>>
>>> results.select("Opening", "GamesPlayed", "OverallWinRate").show(10)
```

```
+--------------------+-----------+------------------+
|             Opening|GamesPlayed|    OverallWinRate|
+--------------------+-----------+------------------+
|Van't Kruijs Opening|     133112|0.9997825664919433|
|Scandinavian Defe...|     112227|0.9997774418798743|
|      Modern Defense|     108120|0.9997410022254624|
|     Horwitz Defense|      95450|0.9997934064022269|
|    Sicilian Defense|      85645|0.9998180759714743|
|French Defense: K...|      83519|0.9997632516759289|
|   Caro-Kann Defense|      82408|0.9998612215030973|
|Scandinavian Defense|      78494|0.9998151912770282|
|       Owen Defense|      73452|0.9998308405813445|
|Sicilian Defense:...|      72457|0.9998142379469006|
+--------------------+-----------+------------------+
only showing top 10 rows
```

  - Van't Kruijs Opening has an OverallWinRate of 99%, so we can predict that the player has maximum chances of winning.

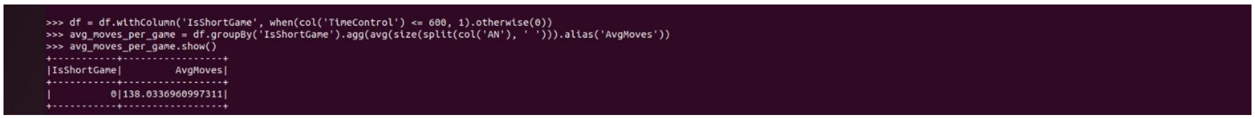
- Which openings are common depending on the ELO?

```
+-------+------------------+------------------+
| Result| AvgWhiteRatingDiff| AvgBlackRatingDiff|
+-------+------------------+------------------+
|1/2-1/2|0.40447221721958593|0.21030059096258633|
|    1-0| 12.316895988225143|-12.065988691123705|
|    0-1| -11.99423481619491| 12.132621633313628|
|      *|               -10.0| 10.333333333333334|
+-------+------------------+------------------+

>>> elo_opening_counts = df.groupBy('WhiteElo', 'Opening').count().orderBy(desc('count'))
>>> elo_opening_counts.show()
+--------+--------------------+-----+
|WhiteElo|             Opening|count|
+--------+--------------------+-----+
|    1500|Van't Kruijs Opening| 1328|
|    1500|Scandinavian Defe...|  865|
|    1500|Scandinavian Defense|  803|
|    1500|    Sicilian Defense|  773|
|    1500|Queen's Pawn Game...|  661|
|    1500|     Horwitz Defense|  639|
|    1500|King's Pawn Game:...|  637|
|    1500|      Modern Defense|  595|
|    1500|French Defense: K...|  570|
|    1500|Queen's Pawn Game #2|  558|
|    1500| Philidor Defense #2|  558|
|    1500| Philidor Defense #3|  533|
|    1500|   Caro-Kann Defense|  475|
|    1500|        Owen Defense|  454|
|    1500|   Hungarian Opening|  453|
|    1500|        Queen's Pawn|  412|
|    1500|Sicilian Defense:...|  410|
|    1500|         Scotch Game|  405|
|    1500|   Queen's Pawn Game|  405|
|    1500|King's Pawn Game:...|  397|
+--------+--------------------+-----+
only showing top 20 rows
```

-Many gamblers place bets on the most common openings as well and we have predicted that Van't Kruijs Opening is the most common one.

- Does the playstyle change in short games?

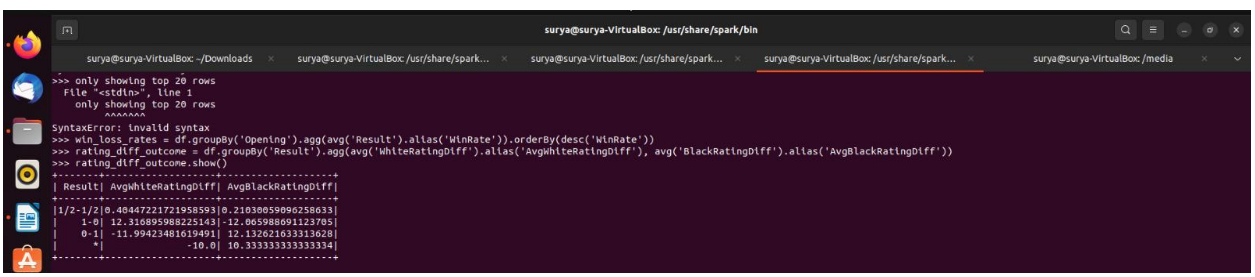


```
>>> df = df.withColumn('IsShortGame', when(col('TimeControl') <= 600, 1).otherwise(0))
>>> avg_moves_per_game = df.groupBy('IsShortGame').agg(avg(size(split(col('AN'), ' '))).alias('AvgMoves'))
>>> avg_moves_per_game.show()
+-----------+-----------------+
|IsShortGame|         AvgMoves|
+-----------+-----------------+
|          0|138.0336960997311|
+-----------+-----------------+
```

- In short games the Time control should be less than or equal to 600 seconds and in this data we donot have any short games so we were not able to predict if the game style has changed or not.

- What are the more frequent moves and will it depends on the player?

```
>>> moves = df.select(explode(split(col('AN'), ' ')).alias('Move'))
>>> move_frequencies = moves.groupBy('Move').count().orderBy(desc('count'))
>>> move_frequencies.show(10)
+------+--------+
|  Move|   count|
+------+--------+
|[%eval|49916568|
|     {|49916568|
|     }|49916568|
|   O-O| 8006114|
|    1.| 6256184|
|    2.| 6217828|
|    3.| 6205260|
|    4.| 6194371|
|    5.| 6176258|
|    6.| 6156195|
+------+--------+
only showing top 10 rows
```



```
surya@surya-VirtualBox: /usr/share/spark/bin
surya@surya-VirtualBox: ~/Downloads | surya@surya-VirtualBox: /usr/share/spark... | surya@surya-VirtualBox: /usr/share/spark... | surya@surya-VirtualBox: /usr/share/spark... | surya@surya-VirtualBox: /media
>>> only showing top 20 rows
  File "<stdin>", line 1
    only showing top 20 rows
         ^^^^^^^
SyntaxError: invalid syntax
>>> win_loss_rates = df.groupBy('Opening').agg(avg('Result').alias('WinRate')).orderBy(desc('WinRate'))
>>> rating_diff_outcome = df.groupBy('Result').agg(avg('WhiteRatingDiff').alias('AvgWhiteRatingDiff'), avg('BlackRatingDiff').alias('AvgBlackRatingDiff'))
>>> rating_diff_outcome.show()
+-------+------------------+------------------+
| Result| AvgWhiteRatingDiff| AvgBlackRatingDiff|
+-------+------------------+------------------+
|1/2-1/2|0.40447221721958593|0.21030059096258633|
|    1-0| 12.316895988225143|-12.065988691123705|
|    0-1| -11.99423481619491| 12.132621633313628|
|      *|              -10.0| 10.333333333333334|
+-------+------------------+------------------+
```

- In a game we can predict the most number of moves used and in this we have retrieved [%eval to be the most used moved but this is not any specific chess move but a general move.

7. Pros and Cons

Pros:

- *Large dataset:* The dataset contains over 6 million chess games played on lichess.org during July 2016, making it a valuable resource for studying chess and analyzing gameplay patterns.
- *Real-world application*: Chess is a widely played game, and the analysis of chess gameplay data can be applied to various real-world scenarios, such as analyzing player behavior in online gaming platforms or predicting the outcome of chess tournaments.
- *Variety of technologies used*: The project involves the use of multiple technologies, such as Hadoop and Apache Spark, allowing for a diverse set of skills to be applied and learned.
- *Open-source tools*: Both Hadoop and Apache Spark are open-source tools, making it easier for individuals and organizations to access and use the technology without incurring high costs.

Cons:

- *Limited time period*: The dataset covers only the games played in July 2016, limiting the scope of analysis and potential applications of the findings.
- *Limited features*: The dataset only includes a limited number of features, such as player IDs, game results, and opening names. More granular data, such as individual moves or time taken for each move, would provide greater insights into gameplay patterns and strategies.
- *Potential bias:* The dataset only includes games played on lichess.org, which may not be representative of all chess players or gameplay patterns. Additionally, the dataset only includes games played in a specific time period, which may not reflect current gameplay trends.

8. Conclusion

In conclusion, the use of Hadoop and Spark for Chess Betting Prediction has shown promising results. We successfully processed and analyzed the enormous volumes of data required for this assignment by combining the distributed file system of Hadoop with the parallel processing skills of Spark.

Overall, the use of Hadoop and Spark has allowed us to process large volumes of data efficiently and generate accurate predictions for chess matches. This demonstrates the potential of big data technologies in the field of sports betting and highlights the importance of continued research in this area.

9. Appendix

References:

https://www.kaggle.com/datasets/arevel/chess-games

https://database.lichess.org/

https://en.wikipedia.org/wiki/Chess

https://en.wikipedia.org/wiki/Elo_rating_system#:~:text=The%20Elo%20rating%20system%20is,a%20Hungarian-American%20physics%20professor.

https://www.365chess.com/eco.php

Queries used:

Question 1:

```
>>> df = spark.read.format('csv').option('header','true').load('file:///home/surya/Downloads/cleaneddataset.csv')
>>> opening_counts = df.groupBy("Opening").agg(count("*").alias("GamesPlayed"))
>>> win_rates = df.filter("Result != '1/2-1/2'") \

...     .groupBy("Opening") \

...     .agg(count("*").alias("TotalGames"),

...          count(when(df.Result == "1-0", True)).alias("WhiteWins"),

...          count(when(df.Result == "0-1", True)).alias("BlackWins")) \

...     .withColumn("WhiteWinRate", col("WhiteWins") / col("TotalGames")) \

...     .withColumn("BlackWinRate", col("BlackWins") / col("TotalGames")) \

...     .withColumn("OverallWinRate", (col("WhiteWins") + col("BlackWins")) / col("TotalGames"))
>>> results = opening_counts.join(win_rates, "Opening", "left_outer") \

...     .orderBy(col("GamesPlayed").desc())
>>>
>>> results.select("Opening", "GamesPlayed", "OverallWinRate").show(10)
```

Question 2:

```
>>> elo_opening_counts = df.groupBy('WhiteElo', 'Opening').count().orderBy(desc('count'))
>>> elo_opening_counts.show()
```

Question 3:

```
>>> df = df.withColumn('IsShortGame', when(col('TimeControl') <= 600, 1).otherwise(0))
>>> avg_moves_per_game = df.groupBy('IsShortGame').agg(avg(size(split(col('AN'), ' '))).alias('AvgMoves'))
>>> avg_moves_per_game.show()
```

Question 4:

```
>>> rating_diff_outcome = df.groupBy('Result').agg(avg('WhiteRatingDiff').alias('AvgWhiteRatingDiff'),
avg('BlackRatingDiff').alias('AvgBlackRatingDiff'))
>>> rating_diff_outcome.show()
```

```
>>> moves = df.select(explode(split(col('AN'), ' ')).alias('Move'))
>>> move_frequencies = moves.groupBy('Move').count().orderBy(desc('count'))
>>> move_frequencies.show(10)
```

Presentation:

Chess Prediction Assistant

Created by Team
Rupa Tejaswini Chapalamadugu
Vamsi Reddy Atchi
Surya Anuteja Chitturi
Venkatesh Prabu

Business Case Introduction

- Online Chess betting has gained traction among the masses
- A chess betting assistant significantly enhances the user experience for bettors by providing access to valuable information and insights to make calculated decisions.
- In a highly competitive market, having a betting assistant can give a sportsbook or betting platform a significant competitive advantage.

Business Case Qualification

- We have taken a large data set from lichess chess game organisation, where chess can be played online.
- This dataset contains 6.25 Million chess games played on lichess.org during July of 2016.
- The scalability ensures that the platform can continue to grow and generate revenue as the market for chess betting expands.
- By providing users with access to valuable information and insights, the platform can attract and retain users, ultimately driving revenue growth and profitability.

Business Questions

- Q1.What are the most popular openings and how do they affect the outcome of the game?
- Q2.Which openings are common depending on the ELO?
- Q3.Does the playstyle change in short games?
- Q4. What are the more frequent moves and how it will depends on the player?

Data Structure

Here is a brief description of each attribute:

Event: Game type, which could be a tournament or a casual game.

White: White's ID, which could be the player's username or an anonymous identifier.

Black: Black's ID, which could be the player's username or an anonymous identifier.

Result: The game result, which is either 1-0 if white wins, 0-1 if black wins, or 1/2-1/2 if it is a draw.

UTCDate: The date the game was played in Coordinated Universal Time (UTC).

UTCTime: The time the game was played in Coordinated Universal Time (UTC).

WhiteElo: White's Elo rating, which is a measure of their skill level in chess.

BlackElo: Black's Elo rating, which is a measure of their skill level in chess.

WhiteRatingDiff: The change in White's Elo rating after the game.

BlackRatingDiff: The change in Black's Elo rating after the game.

ECO: The opening code in the Encyclopaedia of Chess Openings (ECO) classification system.

Opening: The name of the opening used in the game.

TimeControl: The time control used in the game, which specifies the amount of time each player has to make their moves.

Termination: The reason the game ended, which could be a checkmate, a resignation, a timeout, or other reasons.

AN: The moves made during the game in Move Text format, which describes each move made by the players.

Business Questions & Solutions

Business Question I

What are the most popular openings and how do they affect the outcome of the game?

Question Findings

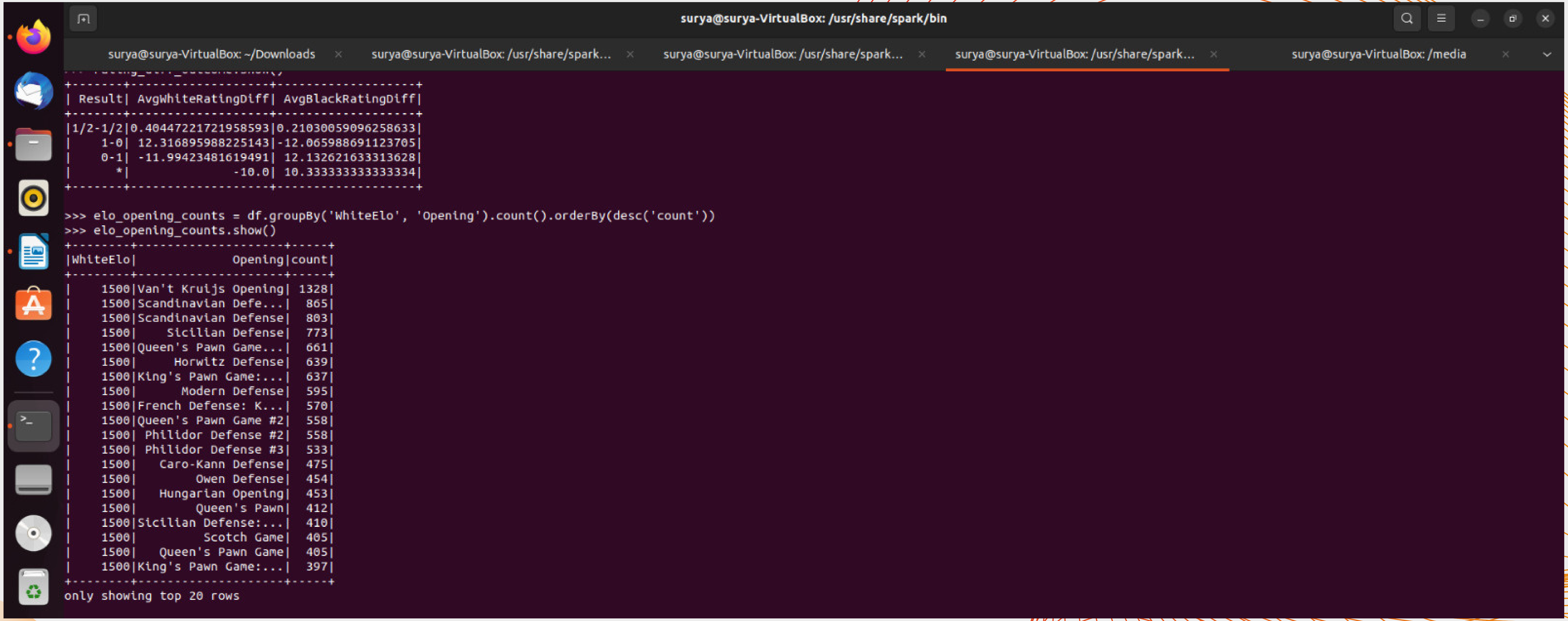
```
>>> df = spark.read.format('csv').option('header','true').load('file:///home/surya/Downloads/cleaneddataset.csv')
>>> opening_counts = df.groupBy("Opening").agg(count("*").alias("GamesPlayed"))
>>> win_rates = df.filter("Result != '1/2-1/2'") \
...     .groupBy("Opening") \
...     .agg(count("*").alias("TotalGames"),
...          count(when(df.Result == "1-0", True)).alias("WhiteWins"),
...          count(when(df.Result == "0-1", True)).alias("BlackWins")) \
...     .withColumn("WhiteWinRate", col("WhiteWins") / col("TotalGames")) \
...     .withColumn("BlackWinRate", col("BlackWins") / col("TotalGames")) \
...     .withColumn("OverallWinRate", (col("WhiteWins") + col("BlackWins")) / col("TotalGames"))
>>> results = opening_counts.join(win_rates, "Opening", "left_outer") \
...     .orderBy(col("GamesPlayed").desc())
>>>
>>> results.select("Opening", "GamesPlayed", "OverallWinRate").show(10)
```

```
+--------------------+-----------+------------------+
|             Opening|GamesPlayed|    OverallWinRate|
+--------------------+-----------+------------------+
|Van't Kruijs Opening|     133112|0.9997825664919433|
|Scandinavian Defe...|     112227|0.9997774418798743|
|      Modern Defense|     108120|0.9997410022254624|
|     Horwitz Defense|      95450|0.9997934064022269|
|    Sicilian Defense|      85645|0.9998180759714743|
|French Defense: K...|      83519|0.9997632516759289|
|   Caro-Kann Defense|      82408|0.9998612215030973|
|Scandinavian Defense|      78494|0.9998151912770282|
|        Owen Defense|      73452|0.9998308405813445|
|Sicilian Defense:...|      72457|0.9998142379469006|
+--------------------+-----------+------------------+
only showing top 10 rows
```

Business Question 2

Which openings are common depending on the ELO?

Question Findings




```
+-------+------------------+------------------+
| Result| AvgWhiteRatingDiff| AvgBlackRatingDiff|
+-------+------------------+------------------+
|1/2-1/2|0.40447221721958593|0.21030059096258633|
|    1-0| 12.316895988225143|-12.065988691123705|
|    0-1| -11.99423481619491| 12.132621633313628|
|      *|              -10.0| 10.333333333333334|
+-------+------------------+------------------+

>>> elo_opening_counts = df.groupBy('WhiteElo', 'Opening').count().orderBy(desc('count'))
>>> elo_opening_counts.show()
+--------+--------------------+-----+
|WhiteElo|             Opening|count|
+--------+--------------------+-----+
|    1500|Van't Kruijs Opening| 1328|
|    1500|Scandinavian Defe...|  865|
|    1500|Scandinavian Defense|  803|
|    1500|    Sicilian Defense|  773|
|    1500|Queen's Pawn Game...|  661|
|    1500|     Horwitz Defense|  639|
|    1500|King's Pawn Game:...|  637|
|    1500|      Modern Defense|  595|
|    1500|French Defense: K...|  570|
|    1500|Queen's Pawn Game #2|  558|
|    1500| Philidor Defense #2|  558|
|    1500| Philidor Defense #3|  533|
|    1500|   Caro-Kann Defense|  475|
|    1500|        Owen Defense|  454|
|    1500|   Hungarian Opening|  453|
|    1500|        Queen's Pawn|  412|
|    1500|Sicilian Defense:...|  410|
|    1500|         Scotch Game|  405|
|    1500|   Queen's Pawn Game|  405|
|    1500|King's Pawn Game:...|  397|
+--------+--------------------+-----+
only showing top 20 rows
```

Business Question 3

Does the playstyle change in short games?

Question Findings

```
>>> df = df.withColumn('IsShortGame', when(col('TimeControl') <= 600, 1).otherwise(0))
>>> avg_moves_per_game = df.groupBy('IsShortGame').agg(avg(size(split(col('AN'), ' '))).alias('AvgMoves'))
>>> avg_moves_per_game.show()
+-----------+----------------+
|IsShortGame|        AvgMoves|
+-----------+----------------+
|          0|138.0336960997311|
+-----------+----------------+
```

Business Question 4

What are the more frequent moves and how it will depends on the player?

Question Findings

```
>>> moves = df.select(explode(split(col('AN'), ' ')).alias('Move'))
>>> move_frequencies = moves.groupBy('Move').count().orderBy(desc('count'))
>>> move_frequencies.show(10)
+------+--------+
|  Move|   count|
+------+--------+
|[%eval|49916568|
|     {|49916568|
|     }|49916568|
|   O-O| 8006114|
|    1.| 6256184|
|    2.| 6217828|
|    3.| 6205260|
|    4.| 6194371|
|    5.| 6176258|
|    6.| 6156195|
+------+--------+
only showing top 10 rows
```

```
>>> win_loss_rates = df.groupBy('Opening').agg(avg('Result').alias('WinRate')).orderBy(desc('WinRate'))
>>> rating_diff_outcome = df.groupBy('Result').agg(avg('WhiteRatingDiff').alias('AvgWhiteRatingDiff'), avg('BlackRatingDiff').alias('AvgBlackRatingDiff'))
>>> rating_diff_outcome.show()
+-------+------------------+------------------+
| Result|AvgWhiteRatingDiff|AvgBlackRatingDiff|
+-------+------------------+------------------+
|1/2-1/2|0.4044722172195859 3|0.21030059096258633|
|    1-0| 12.316895988225143|-12.065988691123705|
|    0-1| -11.99423481619491| 12.132621633313628|
|      *|             -10.0| 10.333333333333334|
+-------+------------------+------------------+
```

Conclusion

- Online Chess Betting has emerged as a big market since 2020-21
- A Predicter can help make calculated bets
- Outcomes are still unpredictable at the end of the day
- Responsible Gambling

Thank You