IRON HACK

Data Analytics

An analysis of the moves played by female grandmasters

François-Régis André

December, 2022

Table of content

Introduction

Chess is a popular game that has been played for centuries. It is the most obvious example of game that requires strategy planning in order to succeed. It is complex and rich enough to be played on a professional level on the international scene and the strongest players are well known by everyone.

I have collected data with over 300 000 games played by female grandmasters on a popular online platform between 2009 and 2021. The goal of this project is to classify these games by opening by recouping with a database of openings, find the most performant moves and try to give predictions on the result of a game.

I started by doing the planning of my project on Jira. The page is here : https://frandre.atlassian.net/jira/software/projects/P9C/boards/5/roadmap

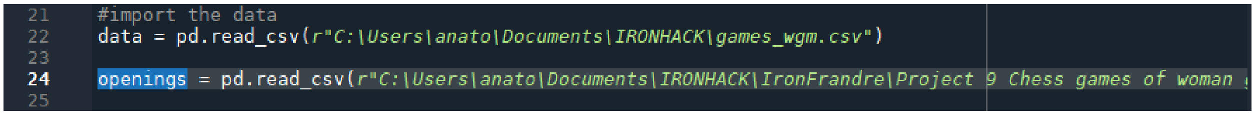
Data and data sources

I have collected two tables from **www.kaggle.com**. The first table describes games played by female grandmasters. Features include a text with the moves of the game and a code for the opening, the username for white and black players, an indication of whether the female grandmaster is playing white or black, the result of the game, the ELO ranking for both players on the platform and the rules used for the timer.

The other table is the classification of the openings by ECO code with the name and the first significant moves for this opening.

Data collection

Once I found the datasets I wanted to work on, I downloaded them from the internet site kaggle. I obtained csv files that were ready to be imported by Python.



```
#import the data
data = pd.read_csv(r"C:\Users\anato\Documents\IRONHACK\games_wgm.csv")

openings = pd.read_csv(r"C:\Users\anato\Documents\IRONHACK\IronFrandre\Project 9 Chess games of woman 
```

Data cleaning and data preparation

I used Python on the spyder interface to perform the data cleaning and the data preparation. My data cleaning consisted of removing columns that hold little useful info and identifying rows that are not pertinent for my analysis or holds an issue that prevents me to work on them. The largest work was the data preparation. I implemented scripts on Python that allowed me to scan the raw text with the details of each game and process the moves in order to extract additional info about the game.

My row data consisted of two tables, one with the games and the other classifies the openings. The opening denotes the first few moves of the game. The possibilities for these first moves are well known and my second table is a classification of the openings with an ID code called ECO for each opening, the name of the opening and the moves at the start of the game that define this particular opening.

The first table holds the most important data, the details of the game played by the female grandmasters. It has 304767 rows and 15 columns. Info includes the pseudo for both players, their ELO ranking on the platform, which of the white player or the black player is the female grandmaster, the rules used for the game (most games use usual chess rules, but a few use a variant), the rule for the timer, the result of the game. The most important column with title “png” contains a raw text that needs to be parsed. There are the details of the moves of the game and a code for the opening.

I started by dropping the data that is not useful to me. I dropped the columns with the game ID and the game URL. These columns identify uniquely each game, but I already have the indexing of my dataframe, so I don’t need those. I dropped all games that use an other set of rules than traditional chess, then dropped the column ‘rules’ since I kept only one value for this column, so it does not hold any info anymore.

I have two columns that detail the reason why the game ended with categorical values. I transformed them in a numerical field, the score of the white player : 0 for a defeat, 1 for a victory and 0.5 for a draw. I do not need to calculate the score for the black player since it can easily be derived from the score of the whites by the function x -> 1 - x.

The major part of the work was to extract useful data from the raw text in the “png” field. For this work, the main tool is the regular expressions librairie in Python that allows me to find the useful part in the text and process it in a suitable way. I extracted the ECO code for the opening that allows to classify the game thanks to the second table.

Then I extracted the string with only the moves of the game. A problem I encountered here was that some games do not start on turn 1. These games appear to have been played with imposed opening and start with a position after a few moves from the usual initial position. In that case, I used the classification by opening. I used the ECO code to retrieve in the second table the missing moves at the start of the game. I checked at what turn the game starts and fusionned the two strings at this point. In some rare cases, the starting moves that I fetched in the second table with the ECO code were not enough to fill the missing part in the game. I dropped these problematic games since I had no way to fix this problem.

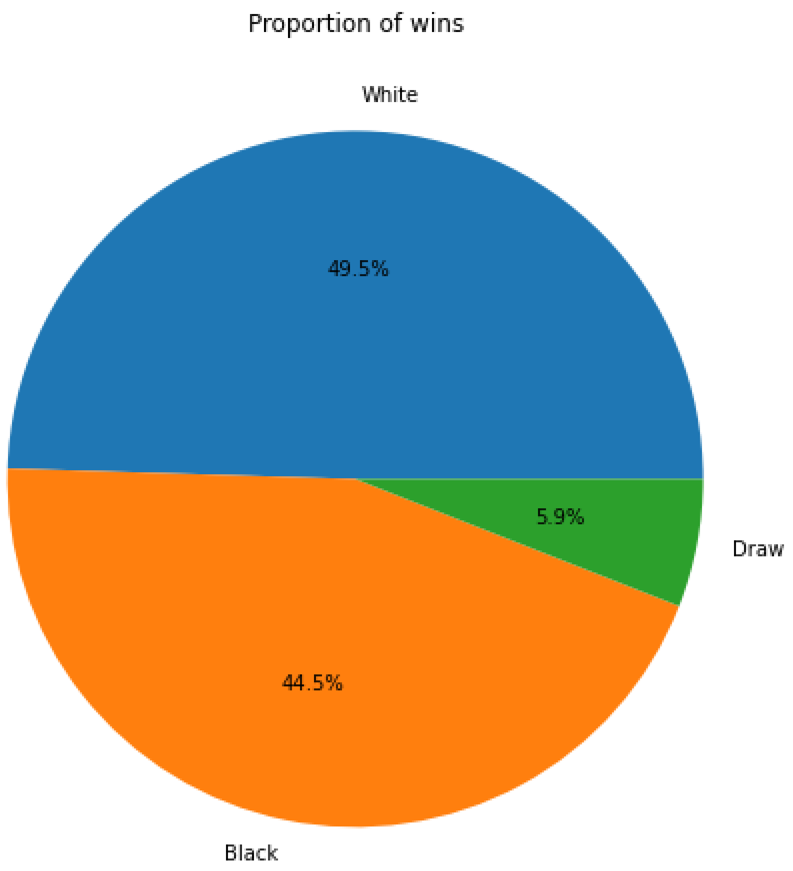
After this, I implemented a program that scans the moves of a game and calculate the position on the chessboard after every move. I coded a position as a string that details all the pieces on the board. My program calculates how the position changes with a move. The challenge here was that the moves are coded so that I know what type of piece moves and where it moves to, but I do not know where it moves from and if it captures an opposing piece, what that is. I need to find these info from the current position, and in the case where the board has more than one piece of the type of the one that moves, I need to decide which is the one that actually moves. Another challenge was that I needed to treat separately all types of pieces and if the move is a capture, I also had to treat it separately, so I had a large number of cases to treat individually.

Thanks to this program, I could parse the full game of each row and calculate the final position of each game. From the final position of each game, I was able to calculate the material balance at the end of the game, so what pieces are still on the board as the game ends.

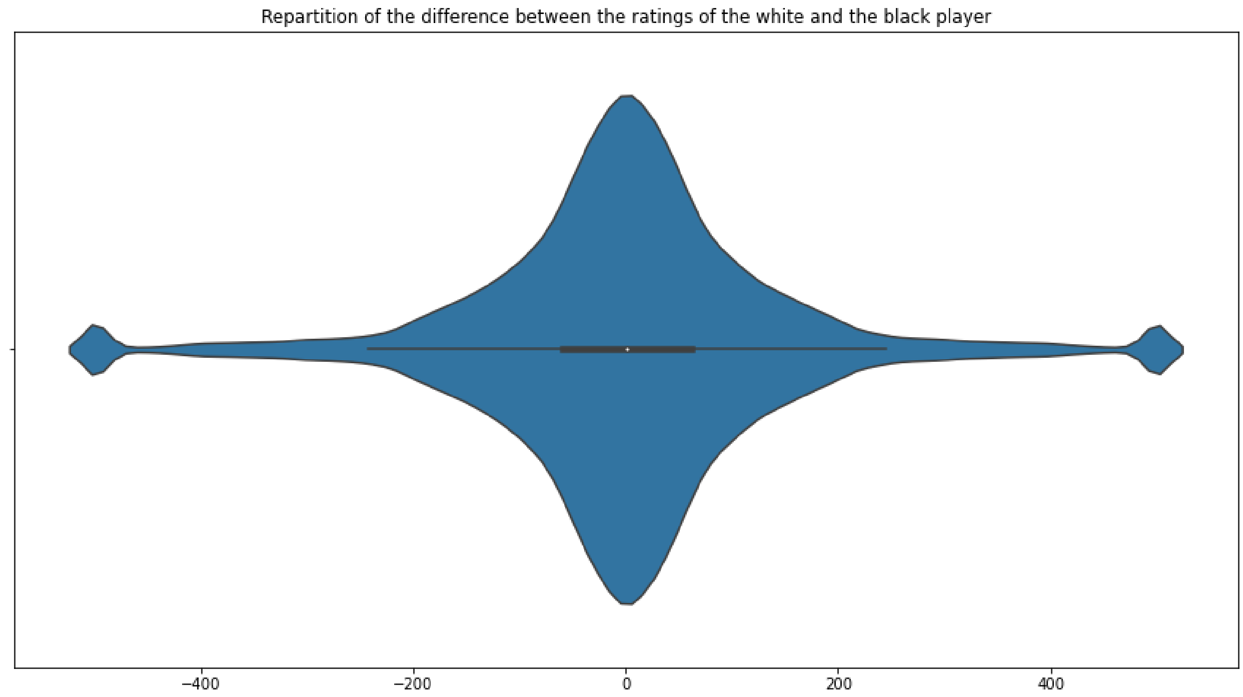
At the end of the process of data cleaning and data preparation, I have 296 129 rows remaining, so 8639 rows were dropped in the process (3%).

Exploratory data analysis

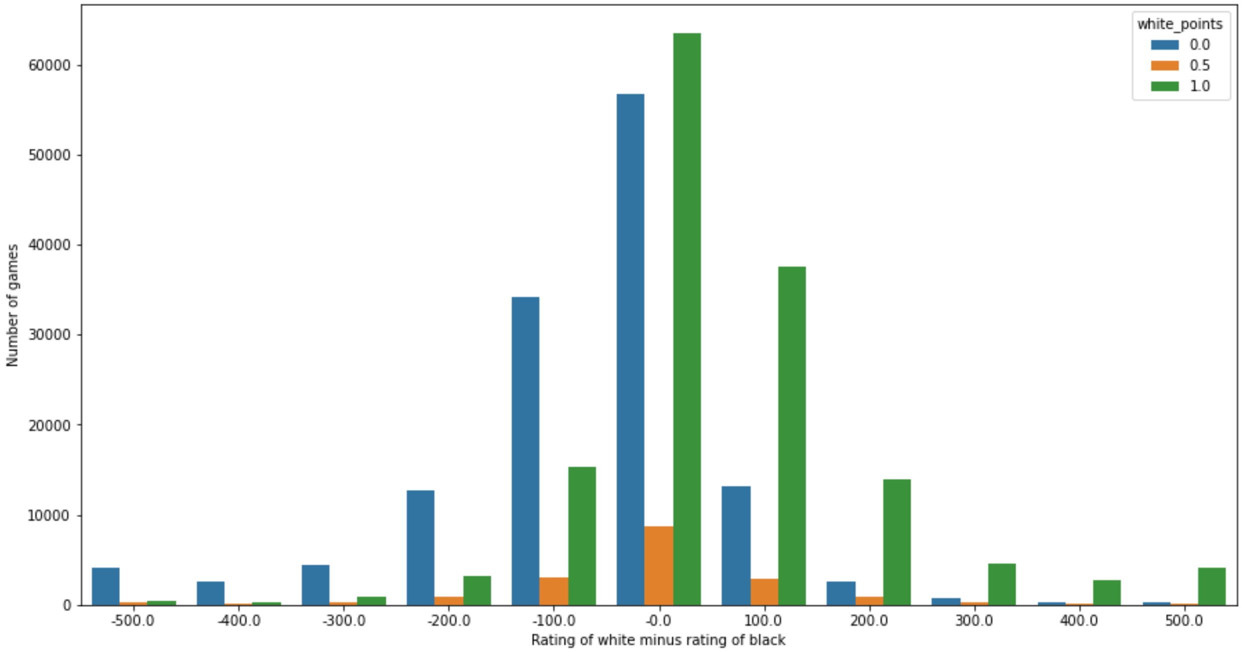
I will now proceed to give some insights on my data. My first statistics is the proportion of games won by the white or black player and the proportion of draws.



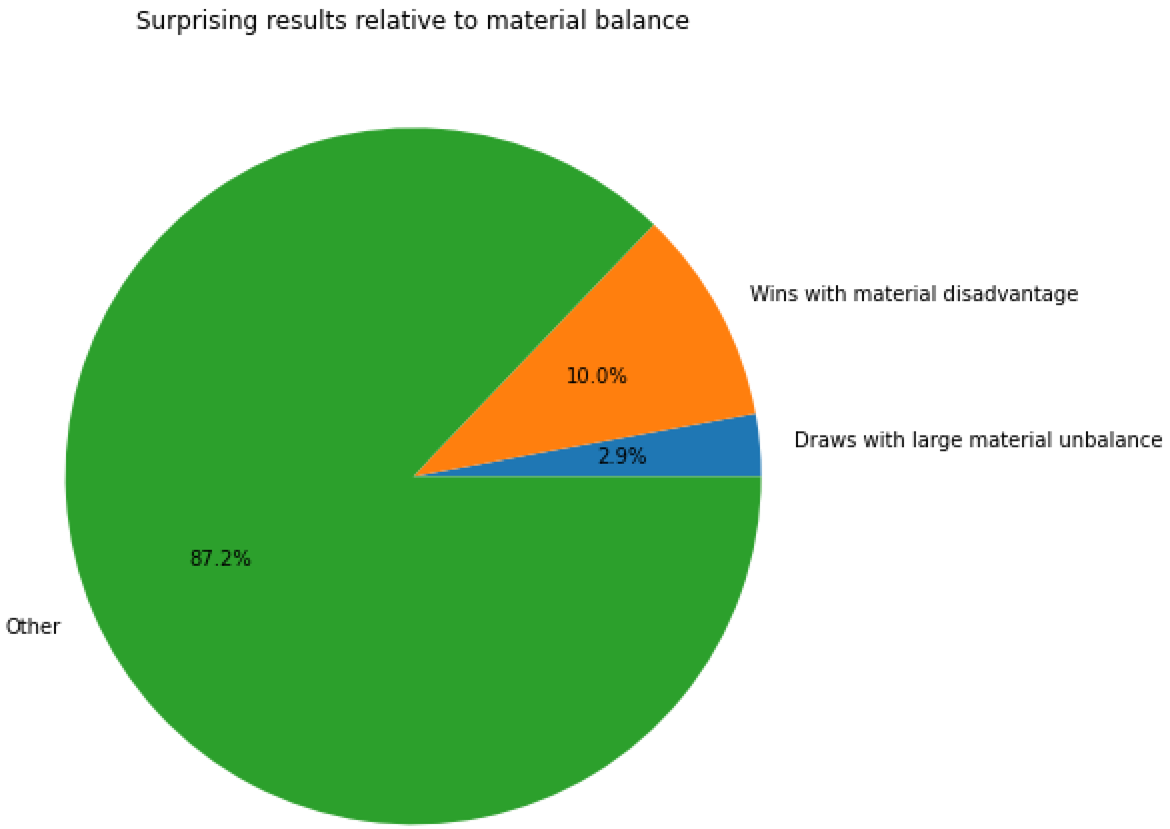
Next, I tried to see how the difference of rating between the white and the black player is distributed. I made a violin diagram to see this. Note that I grouped the games where the difference is more than 500 or less than -500 together so that the center of the diagram is fully visible.



Then, I showed the repartition of white and black wins by the difference in rating between the white and the black player. The difference of rating is binned by slices of 100 and I have grouped the games where the difference is less than -500 or more than 500.



I looked for games with an unusual material balance at the end of the game. I checked the proportion of games where there is a winner and he has more than a pawn of material disadvantage at the end. I also checked the games that ended as a draw and one of the players has more than a pawn of material advantage.



Choice of the database type

Now I need to choose what kind of database I want to use to handle my data. There are two main possibilities, a relational database, so SQL based, or a NoSQL database that is a document store.

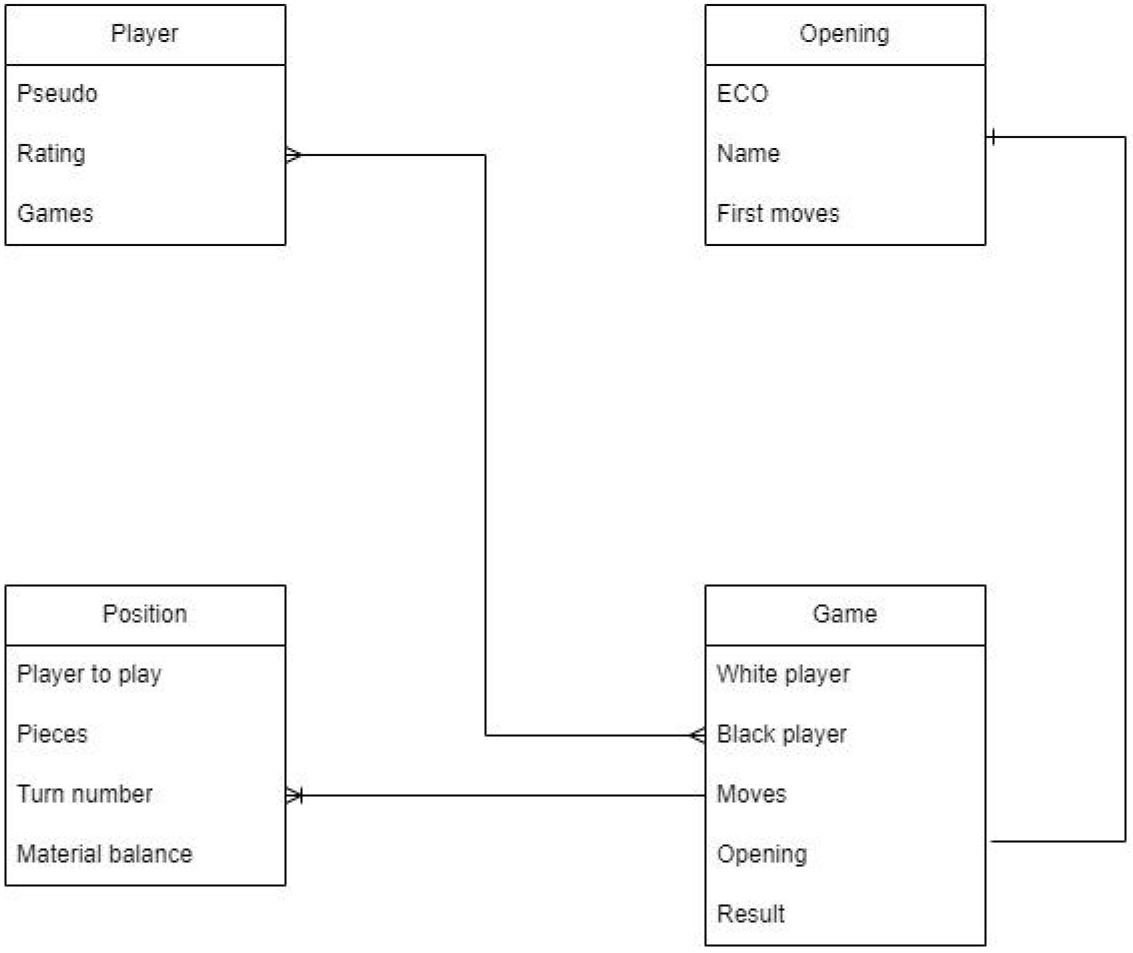
The relational databases use tables and queries make relations between the different rows or the different tables by using a join command. This kind of database allows transactions at the level of cells. Updating is fast, but the querying can be slow, especially when the database is large.

The NoSQL databases use less structured data like documents or json objects. It works better with a large database, but does not allow transactions at the level of cells. The updating can be slower than with SQL, but the querying is much faster and independent of the size of the database.

Since I work with tables, I chose to use a relational database : I will perform queries on MySQL.

Entities. ERD

My data details the relations between four types of entities, players, games, openings and positions. The relations are shown in the following diagram.

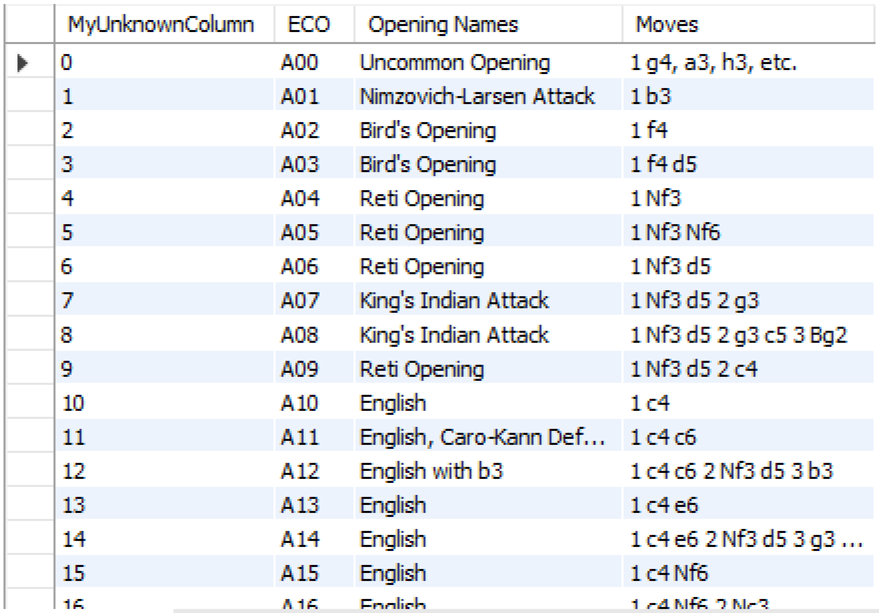


SQL Queries

The first step on my work on MySQL workbench is to create the database. I imported a sample of my data on MySQL and performed some queries to give additional insights.

- `CREATE DATABASE chess;`
- `USE chess;`

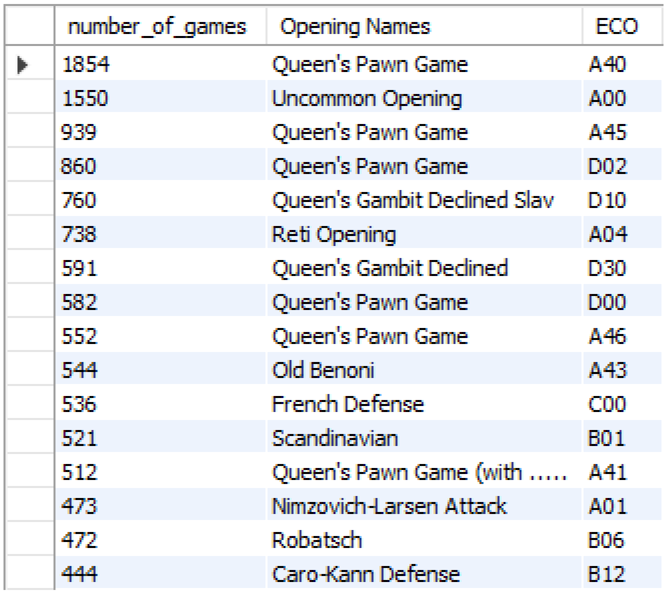
- `SELECT * FROM openings;`



	MyUnknownColumn	ECO	Opening Names	Moves
▶	0	A00	Uncommon Opening	1 g4, a3, h3, etc.
	1	A01	Nimzovich-Larsen Attack	1 b3
	2	A02	Bird's Opening	1 f4
	3	A03	Bird's Opening	1 f4 d5
	4	A04	Reti Opening	1 Nf3
	5	A05	Reti Opening	1 Nf3 Nf6
	6	A06	Reti Opening	1 Nf3 d5
	7	A07	King's Indian Attack	1 Nf3 d5 2 g3
	8	A08	King's Indian Attack	1 Nf3 d5 2 g3 c5 3 Bg2
	9	A09	Reti Opening	1 Nf3 d5 2 c4
	10	A10	English	1 c4
	11	A11	English, Caro-Kann Def...	1 c4 c6
	12	A12	English with b3	1 c4 c6 2 Nf3 d5 3 b3
	13	A13	English	1 c4 e6
	14	A14	English	1 c4 e6 2 Nf3 d5 3 g3 ...
	15	A15	English	1 c4 Nf6
	16	A16	English	1 c4 Nf6 2 Nc3

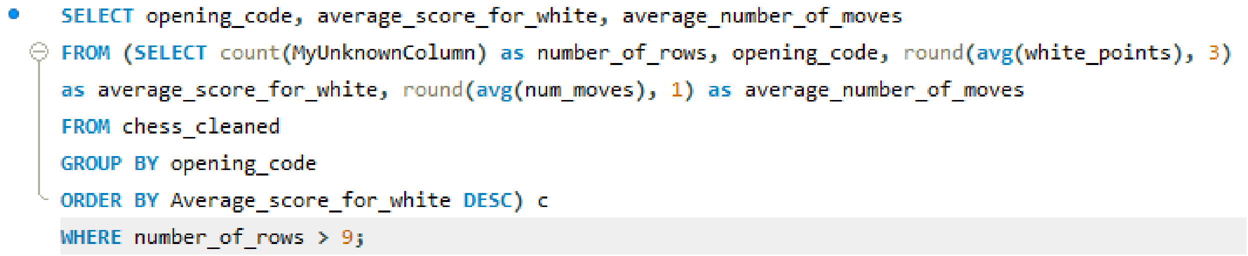
My first query ranks the openings by the number of games with the opening in my data.

-
  ```
  SELECT count(c.MyUnknownColumn) as number_of_games, o.`Opening Names`, o.ECO
  FROM chess_cleaned c
  LEFT JOIN openings o ON c.opening_code = o.ECO
  GROUP BY c.opening_code
  ORDER BY number_of_games DESC;
  ```

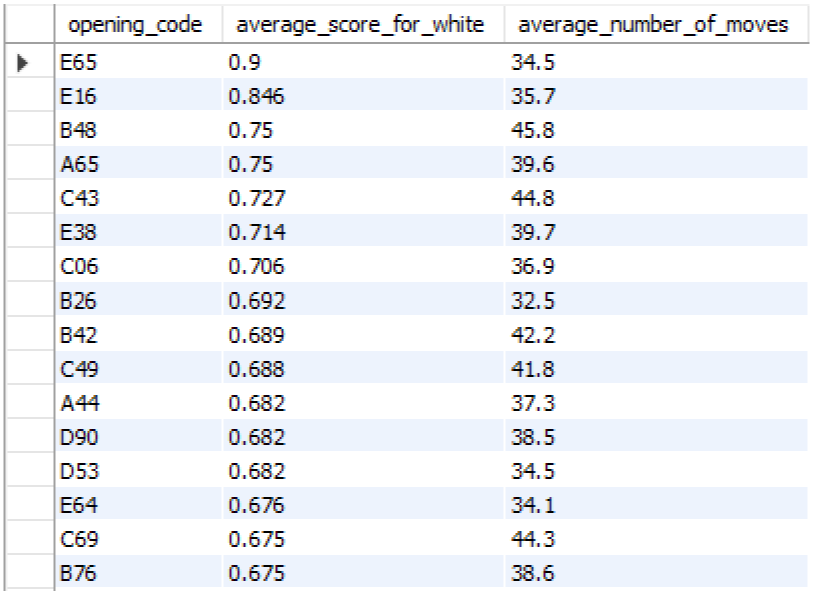


number_of_games	Opening Names	ECO
1854	Queen's Pawn Game	A40
1550	Uncommon Opening	A00
939	Queen's Pawn Game	A45
860	Queen's Pawn Game	D02
760	Queen's Gambit Declined Slav	D10
738	Reti Opening	A04
591	Queen's Gambit Declined	D30
582	Queen's Pawn Game	D00
552	Queen's Pawn Game	A46
544	Old Benoni	A43
536	French Defense	C00
521	Scandinavian	B01
512	Queen's Pawn Game (with	A41
473	Nimzovich-Larsen Attack	A01
472	Robatsch	B06
444	Caro-Kann Defense	B12

My next query ranks the openings by average score for the white player and gives the average number of moves of the games with the opening.

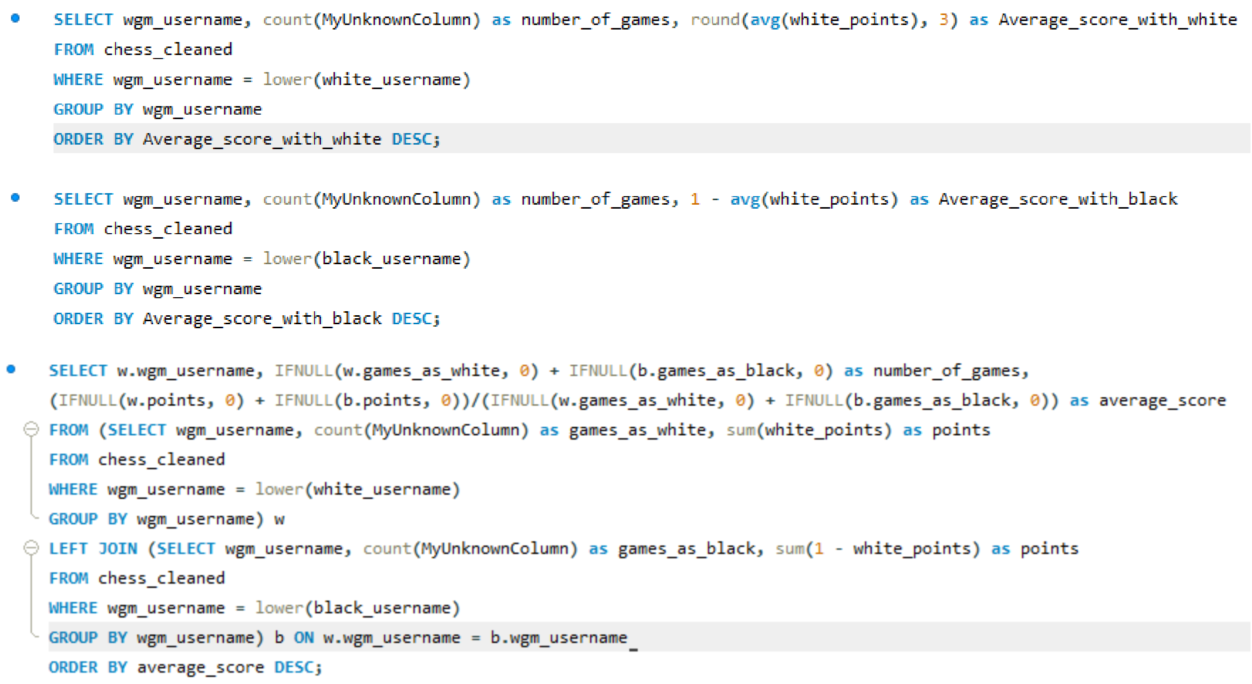


```
SELECT opening_code, average_score_for_white, average_number_of_moves
FROM (SELECT count(MyUnknownColumn) as number_of_rows, opening_code, round(avg(white_points), 3)
as average_score_for_white, round(avg(num_moves), 1) as average_number_of_moves
FROM chess_cleaned
GROUP BY opening_code
ORDER BY Average_score_for_white DESC) c
WHERE number_of_rows > 9;
```

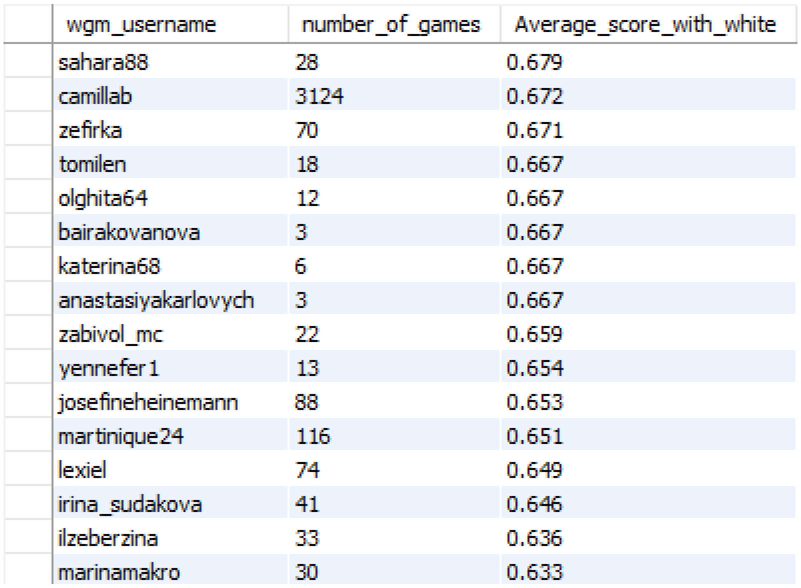


opening_code	average_score_for_white	average_number_of_moves
E65	0.9	34.5
E16	0.846	35.7
B48	0.75	45.8
A65	0.75	39.6
C43	0.727	44.8
E38	0.714	39.7
C06	0.706	36.9
B26	0.692	32.5
B42	0.689	42.2
C49	0.688	41.8
A44	0.682	37.3
D90	0.682	38.5
D53	0.682	34.5
E64	0.676	34.1
C69	0.675	44.3
B76	0.675	38.6

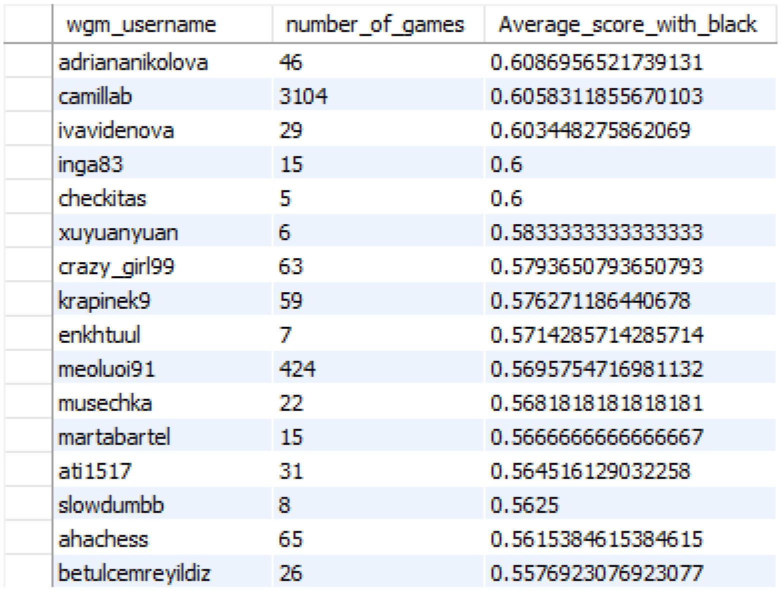
Then I ranked the female grandmasters by average score as white player, black player and both colors with their number of games played. I chose to show the part where camillab appears as she is the grandmaster with the most games in my dataset.



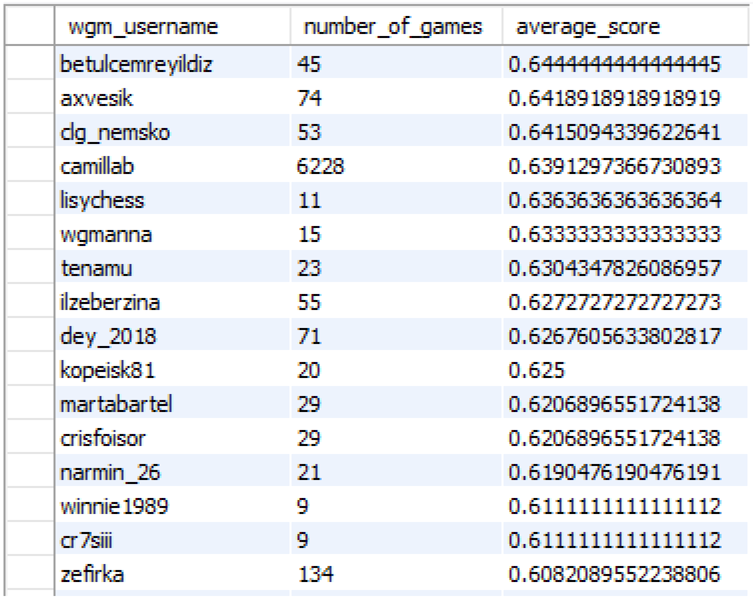
```sql
SELECT wgm_username, count(MyUnknownColumn) as number_of_games, round(avg(white_points), 3) as Average_score_with_white
FROM chess_cleaned
WHERE wgm_username = lower(white_username)
GROUP BY wgm_username
ORDER BY Average_score_with_white DESC;

SELECT wgm_username, count(MyUnknownColumn) as number_of_games, 1 - avg(white_points) as Average_score_with_black
FROM chess_cleaned
WHERE wgm_username = lower(black_username)
GROUP BY wgm_username
ORDER BY Average_score_with_black DESC;

SELECT w.wgm_username, IFNULL(w.games_as_white, 0) + IFNULL(b.games_as_black, 0) as number_of_games,
(IFNULL(w.points, 0) + IFNULL(b.points, 0))/(IFNULL(w.games_as_white, 0) + IFNULL(b.games_as_black, 0)) as average_score
FROM (SELECT wgm_username, count(MyUnknownColumn) as games_as_white, sum(white_points) as points
FROM chess_cleaned
WHERE wgm_username = lower(white_username)
GROUP BY wgm_username) w
LEFT JOIN (SELECT wgm_username, count(MyUnknownColumn) as games_as_black, sum(1 - white_points) as points
FROM chess_cleaned
WHERE wgm_username = lower(black_username)
GROUP BY wgm_username) b ON w.wgm_username = b.wgm_username
ORDER BY average_score DESC;
```



wgm_username	number_of_games	Average_score_with_white
sahara88	28	0.679
camillab	3124	0.672
zefirka	70	0.671
tomilen	18	0.667
olghita64	12	0.667
bairakovanova	3	0.667
katerina68	6	0.667
anastasiyakarlovych	3	0.667
zabivol_mc	22	0.659
yennefer1	13	0.654
josefineheinemann	88	0.653
martinique24	116	0.651
lexiel	74	0.649
irina_sudakova	41	0.646
ilzeberzina	33	0.636
marinamakro	30	0.633

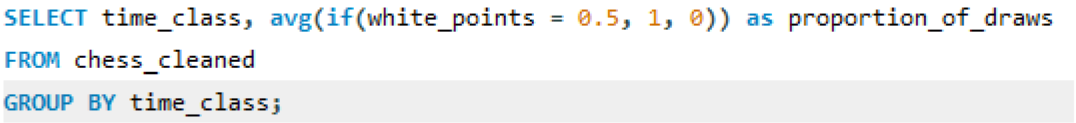


wgm_username	number_of_games	Average_score_with_black
adriananikolova	46	0.6086956521739131
camillab	3104	0.6058311855670103
ivavidenova	29	0.603448275862069
inga83	15	0.6
checkitas	5	0.6
xuyuanyuan	6	0.5833333333333333
crazy_girl99	63	0.5793650793650793
krapinek9	59	0.576271186440678
enkhtuul	7	0.5714285714285714
meoluoi91	424	0.5695754716981132
musechka	22	0.5681818181818181
martabartel	15	0.5666666666666667
ati1517	31	0.564516129032258
slowdumbb	8	0.5625
ahachess	65	0.5615384615384615
betulcemreyildiz	26	0.5576923076923077

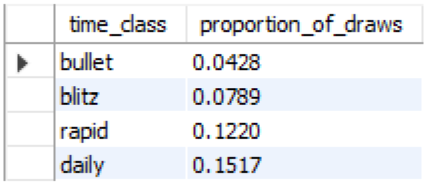


	wgm_username	number_of_games	average_score
	betulcemreyildiz	45	0.6444444444444445
	axvesik	74	0.6418918918918919
	dg_nemsko	53	0.6415094339622641
	camillab	6228	0.6391297366730893
	lisychess	11	0.6363636363636364
	wgmanna	15	0.6333333333333333
	tenamu	23	0.6304347826086957
	ilzeberzina	55	0.6272727272727273
	dey_2018	71	0.6267605633802817
	kopeisk81	20	0.625
	martabartel	29	0.6206896551724138
	crisfoisor	29	0.6206896551724138
	narmin_26	21	0.6190476190476191
	winnie1989	9	0.6111111111111112
	cr7siii	9	0.6111111111111112
	zefirka	134	0.6082089552238806

Next I showed the proportion of games that resulted as a draw for each of the four types of rule for the timer. We can see that the proportion of draws increases as the rules allow slower games.

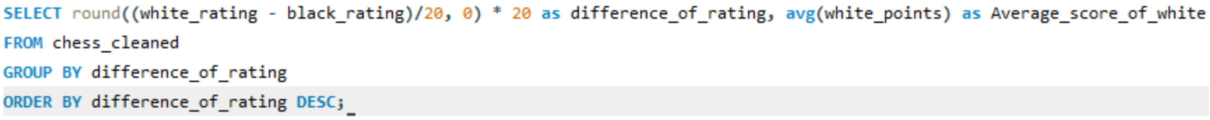
- 

```
SELECT time_class, avg(if(white_points = 0.5, 1, 0)) as proportion_of_draws
FROM chess_cleaned
GROUP BY time_class;
```

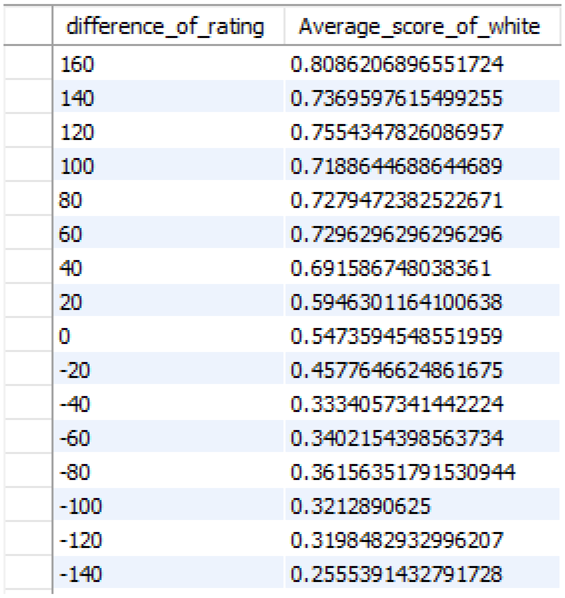


	time_class	proportion_of_draws
▶	bullet	0.0428
	blitz	0.0789
	rapid	0.1220
	daily	0.1517

Finally, I calculated the average score of the white player by the difference of rating between the white player and the black player. I binned the games by this difference using slices of 20. I showed the part of the result centered around 0 to show how the average score of the white players changes as the difference varies around 0.

- 

```
SELECT round((white_rating - black_rating)/20, 0) * 20 as difference_of_rating, avg(white_points) as Average_score_of_white
FROM chess_cleaned
GROUP BY difference_of_rating
ORDER BY difference_of_rating DESC;
```



	difference_of_rating	Average_score_of_white
	160	0.8086206896551724
	140	0.7369597615499255
	120	0.7554347826086957
	100	0.7188644688644689
	80	0.7279472382522671
	60	0.7296296296296296
	40	0.691586748038361
	20	0.5946301164100638
	0	0.5473594548551959
	-20	0.4577646624861675
	-40	0.3334057341442224
	-60	0.3402154398563734
	-80	0.36156351791530944
	-100	0.3212890625
	-120	0.3198482932996207
	-140	0.2555391432791728

Conclusion

In this study, we have examined a sample of games played by grandmasters on an online platform. We focused on the correlation between the difference of rating between the two players and the final result of the game. A large part of the work has been made on the raw text with the moves of the game to extract additional info about the game. The next step of this study is to implement supervised machine learning algorithms to try to make predictions on the result of a game using the info we have on the game.