lichess

April 24, 2025

1 Projet Lichess

Traitements et données large échelle

Zoé Marquis & Charlotte Kruzic

1.1 Introduction

L'**objectif** de ce projet était d'explorer et d'analyser un ensemble de données provenant de LiChess, une plateforme d'échecs en ligne open-source qui accueille quotidiennement des millions de joueurs de tous niveaux, et qui publie les parties jouées et annotées par le moteur d'échecs Stockfish.

Nous avons utilisé ces **données** pour réaliser ce projet, plus particulièrement les parties jouées en septembre 2020, disponible sur Kaggle.

Pour analyser ce jeu de données contenant environ 3.74 millions de parties et 40 colonnes décrivant différents éléments des parties d'échecs (indicateurs de niveau, type de partie, ouvertures, erreurs…), nous avons utilisé Spark, un outil de traitement de données large échelle.

Nous avons commencé par répondre aux **troix questions** principales du projet en utilisant ces données, et avons donc analysé les erreurs par catégorie ELO dans les parties Blitz, calculé la probabilité de victoire en fonction de l'ouverture, et cherché à prédire le résultat d'une partie.

Enfin, nous avons élargi notre analyse en répondant à des **questions supplémentaires** notamment l'impact des ouvertures jouées sur les matchs nuls, et la relation entre l'ELO et la durée d'une partie.

Les analyses réalisées dans ce projet ont permis de répondre aux questions principales tout en ouvrant des perspectives intéressantes grâce aux analyses supplémentaires. Les résultats obtenus sont accompagnés de visualisations et d'interprétations détaillées, afin d'avoir une meilleure compréhension des dynamiques des parties sur LiChess.

1.2 Installation et importation des bibliothèques nécessaires

[1]:
```
!pip install kagglehub
```

```
Requirement already satisfied: kagglehub in /usr/local/lib/python3.10/dist-
packages (0.3.5)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from kagglehub) (24.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from kagglehub) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
```

```
(from kagglehub) (4.67.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->kagglehub) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->kagglehub) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->kagglehub) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->kagglehub) (2024.12.14)
```

[2]:
```
!pip install -q findspark
```

[3]:
```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-
packages (3.5.3)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-
packages (from pyspark) (0.10.9.7)
```

[4]:
```
# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import json
import os
import kagglehub
from collections import defaultdict

import findspark
from pyspark.sql import SparkSession
```

[5]:
```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://downloads.apache.org/spark/spark-3.5.3/spark-3.5.3-bin-hadoop3.
↪tgz
!tar xf spark-3.5.3-bin-hadoop3.tgz
```

[6]:
```
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.5.3-bin-hadoop3"
```

1.3 Préparation des données et de l'environnement

Nous commençons par charger les données depuis Kaggle, puis nous faisons une analyse exploratoire et prétraitons les données afin d'avoir une base pour nos analyses.

1.3.1 Chargement des données

[7]:
```
path = kagglehub.dataset_download("noobiedatascientist/
  ↪lichess-september-2020-data")
print("Chemin vers le fichier du dataset : ", path)
```

```
Warning: Looks like you're using an outdated `kagglehub` version (installed:
0.3.5), please consider upgrading to the latest version (0.3.6).
Chemin vers le fichier du dataset :
/root/.cache/kagglehub/datasets/noobiedatascientist/lichess-
september-2020-data/versions/3
```

[8]:
```
files = os.listdir(path)
print("Fichiers du dataset : ", files)
```

```
Fichiers du dataset :  ['Sept_20_analysis.csv', 'Column information.txt',
'Sept_20_analysis.RDS']
```

[9]:
```
filename = f"{path}/Sept_20_analysis.csv"
print("Nom du fichier : ", filename)
```

```
Nom du fichier :  /root/.cache/kagglehub/datasets/noobiedatascientist/lichess-
september-2020-data/versions/3/Sept_20_analysis.csv
```

[10]:
```
# voir le contenu du .txt
filename_txt = f"{path}/Column information.txt"
with open(filename_txt, 'r') as f:
    print(f.read())
```

```
GAME: Game ID (not from lichess.org)

BlackElo: Elo rating of the player with the black pieces

BlackRatingDiff: Rating change (gain/loss) after game conclusion for the player
with the black pieces

Date: Date the game was played

ECO: Game opening (ECO notation)

Event: Event where the game was played

Opening: Game opening

Result: Result of the game

        1-0 -- White victory
        0-1 -- Black victory
        1/2-1/2 -- Draw
        * -- Undecided
```

Site: URL of the game

Termination: Way the game terminated

- Time forfeit -- One of the players ran out of time
- Normal -- Game terminated with check mate
- Rules infraction -- Game terminated due to rule breaking
- Abandoned -- Game was abandoned

TimeControl: Timecontrol in seconds that was used for the game (Starting time: Increment)

UTCTime: Time the game was played

WhiteElo: Elo rating of the player with the white pieces

WhiteRatingDiff: Rating change (gain/loss) after game conclusion for the player with the white pieces

Black_elo_category: ELO category of the player with the black pieces

- Low rating -- Rating below 1900
- High rating -- Rating above 1900 and below 2400
- GM rating -- Rating above 2400

starting_time: The time in seconds that the players have available at the start of the game (taken from TimeControl)

- EMPTY -- Correspondence games

increment: Time increment in seconds that was used in the game (taken from TimeControl)

- EMPTY -- Correspondence games

Game_type: Type of game based on TimeControl

- Bullet -- Starting time below 2 minutes
- Blitz -- Starting time between 2 and 10 minutes
- Rapid -- Starting time between 10 and 15 minutes
- Classical -- Starting time above 15 minutes or increment 2 minutes or higher
- Correspondence -- No time information

Total_moves: Total number of moves in the game

Black_blunders: Number of blunders by the player with the black pieces (move

annotation ?? in the PGN)

White_blunders: Number of blunders by the player with the white pieces (move annotation ?? in the PGN)

Black_mistakes: Number of mistakes by the player with the black pieces (move annotation ? in the PGN)

White_mistakes: Number of mistakes by the player with the white pieces (move annotation ? in the PGN)

Black_inaccuracies: Number of inaccuracies by the player with the black pieces (move annotation ?! in the PGN)

White_inaccuracies: Number of inaccuracies by the player with the white pieces (move annotation ?! in the PGN)

Black_inferior_moves: Black_blunders + Black_mistakes + Black_inaccuracies

White_inferior_moves: White_blunders + White_mistakes + White_inaccuracies

Black_ts_moves: Number of moves by the player with the black pieces in time scramble (remaining time less than or equal to 10% of the starting time)

White_ts_moves: Number of moves by the player with the white pieces in time scramble (remaining time less than or equal to 10% of the starting time)

Black_ts_blunders: Number of blunders by the player with the black pieces in time scramble (remaining time less than or equal to 10% of the starting time)

White_ts_blunders: Number of blunders by the player with the white pieces in time scramble (remaining time less than or equal to 10% of the starting time)

Black_ts_mistakes: Number of mistakes by the player with the black pieces in time scramble (remaining time less than or equal to 10% of the starting time)

White_ts_mistakes: Number of mistakes by the player with the white pieces in time scramble (remaining time less than or equal to 10% of the starting time)

Black_long_moves: Number of moves by the player with the black pieces that required more than 10% of the starting time

White_long_moves: Number of moves by the player with the white pieces that required more than 10% of the starting time

Black_bad_long_moves: Number of long moves by the player with the black pieces that were inferior

White_bad_long_moves: Number of long moves by the player with the white pieces that were inferior

Game_flips: Number of times in the game where the balance of the game changed

Game_flips_ts: Number of times in the game where the balance of the game changed and the players were in time scramble

1.3.2 Lancement de Spark

[11]:
```
# Imports des éléments Spark
from pyspark.sql.functions import col, when, isnull, floor, count, min as␣
 ↪spark_min, max as spark_max
from pyspark.sql.functions import countDistinct, row_number, split, concat_ws,␣
 ↪sum as spark_sum, rank
from pyspark.ml.functions import vector_to_array
from pyspark.sql import functions as F
from pyspark.sql.window import Window
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler,␣
 ↪MinMaxScaler, StandardScaler
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.stat import Correlation, ChiSquareTest
from pyspark.ml import Pipeline
```

[12]:
```
findspark.init()
spark = SparkSession.builder.master("local[*]").getOrCreate()
```

[13]:
```
sc = spark.sparkContext
df_spark = spark.read.csv(filename, header=True, inferSchema=True)
```

[14]:
```
df_spark.printSchema()
```

```
root
 |-- GAME: integer (nullable = true)
 |-- BlackElo: integer (nullable = true)
 |-- BlackRatingDiff: integer (nullable = true)
 |-- Date: string (nullable = true)
 |-- ECO: string (nullable = true)
 |-- Event: string (nullable = true)
 |-- Opening: string (nullable = true)
 |-- Result: string (nullable = true)
```

```
 |-- Site: string (nullable = true)
 |-- Termination: string (nullable = true)
 |-- TimeControl: string (nullable = true)
 |-- UTCTime: timestamp (nullable = true)
 |-- WhiteElo: integer (nullable = true)
 |-- WhiteRatingDiff: integer (nullable = true)
 |-- Black_elo_category: string (nullable = true)
 |-- White_elo_category: string (nullable = true)
 |-- starting_time: integer (nullable = true)
 |-- increment: integer (nullable = true)
 |-- Game_type: string (nullable = true)
 |-- Total_moves: integer (nullable = true)
 |-- Black_blunders: integer (nullable = true)
 |-- White_blunders: integer (nullable = true)
 |-- Black_mistakes: integer (nullable = true)
 |-- White_mistakes: integer (nullable = true)
 |-- Black_inaccuracies: integer (nullable = true)
 |-- White_inaccuracies: integer (nullable = true)
 |-- Black_inferior_moves: integer (nullable = true)
 |-- White_inferior_moves: integer (nullable = true)
 |-- Black_ts_moves: integer (nullable = true)
 |-- White_ts_moves: integer (nullable = true)
 |-- Black_ts_blunders: integer (nullable = true)
 |-- White_ts_blunders: integer (nullable = true)
 |-- Black_ts_mistakes: integer (nullable = true)
 |-- White_ts_mistake: integer (nullable = true)
 |-- Black_long_moves: integer (nullable = true)
 |-- White_long_moves: integer (nullable = true)
 |-- Black_bad_long_moves: integer (nullable = true)
 |-- White_bad_long_moves: integer (nullable = true)
 |-- Game_flips: integer (nullable = true)
 |-- Game_flips_ts: integer (nullable = true)
```

[15]:
```
# nombre lignes
df_spark.count()
```

[15]: 3739909

[16]:
```
df_spark.show(5)
```

```
+----+--------+--------------+----------+---+---------------+----------------
---+------+--------------------+------------+-----------+-------------------+---
-----+---------------+-----------------+-----------------+-------------+------
---+---------+-----------+--------------+--------------+--------------+---------
-----+------------------+------------------+--------------------+---------------
-----+--------------+--------------+-----------------+----------------+--------
---------+----------------+----------------+----------------+-------------------
```

```
-+--------------------+----------+-------------+
|GAME|BlackElo|BlackRatingDiff|      Date|ECO|           Event|
Opening|Result|                Site| Termination|TimeControl|            UTCTime
|WhiteElo|WhiteRatingDiff|Black_elo_category|White_elo_category|starting_time|in
crement|Game_type|Total_moves|Black_blunders|White_blunders|Black_mistakes|White
_mistakes|Black_inaccuracies|White_inaccuracies|Black_inferior_moves|White_infer
ior_moves|Black_ts_moves|White_ts_moves|Black_ts_blunders|White_ts_blunders|Blac
k_ts_mistakes|White_ts_mistake|Black_long_moves|White_long_moves|Black_bad_long_
moves|White_bad_long_moves|Game_flips|Game_flips_ts|
+----+--------+--------------+----------+---+----------------+----------------
---+------+--------------------+-----------+-----------+--------------------+---
-----+---------------+------------------+------------------+-------------+------
---+---------+-----------+--------------+--------------+--------------+---------
-----+------------------+------------------+--------------------+---------------
-----+--------------+--------------+-----------------+-----------------+--------
---------+----------------+----------------+----------------+--------------------
-+--------------------+----------+-------------+
|  11|    1143|              6|2020.09.01|A02|Rated Blitz game|        Bird
Opening|   0-1|https://lichess.o…|Time forfeit|      300+0|2025-01-07
00:00:00|    1180|             -7|        Low rating|        Low rating|
300|        0|    Blitz|         66|              4|             2|
0|             3|                 3|                 1|                   7|
6|             8|             8|                0|                0|
0|               0|              2|               1|                  1|
1|         8|            0|
|  14|    1504|           NULL|2020.09.01|A04|Rated Blitz game|        Réti
Opening|   0-1|https://lichess.o…|      Normal|      300+0|2025-01-07
00:00:00|    1381|           NULL|        Low rating|        Low rating|
300|        0|    Blitz|         64|              2|             1|
1|             1|                 7|                 5|                  10|
7|             0|             0|                0|                0|
0|               0|              0|               1|                  0|
0|         6|            0|
|  29|    1933|              1|2020.09.01|C41|Rated Blitz game|    Philidor
Defense|   0-1|https://lichess.o…|Time forfeit|      300+2|2025-01-07
00:00:00|    1485|             -1|       High rating|        Low rating|
300|        2|    Blitz|         70|              0|             1|
1|             2|                 8|                 8|                   9|
11|             0|             2|                0|                0|
0|               0|              1|               1|                  1|
0|         5|            0|
|  40|    1710|             10|2020.09.01|B23|Rated Blitz game|Sicilian
Defense:…|   0-1|https://lichess.o…|      Normal|      180+2|2025-01-07
00:00:00|    2040|            -11|        Low rating|       High rating|
180|        2|    Blitz|         86|              4|             2|
1|             5|                 3|                 4|                   8|
11|            18|             0|                4|                0|
0|               0|              3|               1|                  1|
```

```
0|         8|            1|
|  55|    1598|             -1|2020.09.01|B03|Rated Rapid game|    Alekhine
Defense|   1-0|https://lichess.o…|      Normal|      600+0|2025-01-07
00:00:00|    2163|              0|        Low rating|       High rating|
600|        0|    Rapid|         71|             1|             0|
1|             1|                 6|                2|                   8|
3|             0|             0|                0|                0|
0|               0|               0|               0|                   0|
0|         2|            0|
+----+--------+---------------+----------+---+----------------+----------------
---+------+-------------------+-----------+----------+------------------+---
-----+---------------+-----------------+-----------------+-------------+------
---+---------+-----------+--------------+--------------+-------------+---------
-----+------------------+------------------+--------------------+---------------
-----+--------------+--------------+-----------------+----------------+--------
---------+----------------+----------------+----------------+------------------
-+--------------------+----------+-------------+
only showing top 5 rows
```

1.3.3 Préparation générale des données

Maintenant que les données sont chargées, nous y ajoutons les catégories ELO basées sur les plages données dans l'énoncé du projet.

En plus des 5 catégories définies dans l'énoncé, nous ajoutons "other lower bound" et "other upper bound" pour les valeurs ELO hors des plages.

Calcule des catégories ELO

[17]:
```
# Ajout des catégories ELO
# Catégorie ELO du joueur Noir
df_spark_plus = df_spark.withColumn("Black_ELO_category",
                                    when((col("BlackElo") >= 1200) & (col("BlackElo")␣
 ↪<= 1499), "occasional player")
                                    .when((col("BlackElo") >= 1500) &␣
 ↪(col("BlackElo") <= 1799), "good club player")
                                    .when((col("BlackElo") >= 1800) &␣
 ↪(col("BlackElo") <= 1999), "very good club player")
                                    .when((col("BlackElo") >= 2000) &␣
 ↪(col("BlackElo") <= 2399), "national and international level")
                                    .when((col("BlackElo") >= 2400) &␣
 ↪(col("BlackElo") <= 2800), "GMI, World Champions")
                                    .when((col("BlackElo") < 1200), "other lower␣
 ↪bound")
                                    .otherwise("other upper bound")
                                    )

# Catégorie ELO du joueur Blanc
```

```
df_spark_plus = df_spark_plus.withColumn("White_ELO_category",
                              when((col("WhiteElo") >= 1200) & (col("WhiteElo")␣
 ↪<= 1499), "occasional player")
                              .when((col("WhiteElo") >= 1500) &␣
 ↪(col("WhiteElo") <= 1799), "good club player")
                              .when((col("WhiteElo") >= 1800) &␣
 ↪(col("WhiteElo") <= 1999), "very good club player")
                              .when((col("WhiteElo") >= 2000) &␣
 ↪(col("WhiteElo") <= 2399), "national and international level")
                              .when((col("WhiteElo") >= 2400) &␣
 ↪(col("WhiteElo") <= 2800), "GMI, World Champions")
                              .when((col("WhiteElo") < 1200), "other lower␣
 ↪bound")
                              .otherwise("other upper bound")
                              )

# Catégorie ELO moyenne des 2 joueurs
df_spark_plus = df_spark_plus.withColumn("Avg_ELO_category", (col("BlackElo") +␣
 ↪col("WhiteElo")) / 2)

df_spark_plus = df_spark_plus.withColumn("Avg_ELO_category",
                              when((col("Avg_ELO_category") >= 1200) &␣
 ↪(col("Avg_ELO_category") <= 1499), "occasional player")
                              .when((col("Avg_ELO_category") >= 1500) &␣
 ↪(col("Avg_ELO_category") <= 1799), "good club player")
                              .when((col("Avg_ELO_category") >= 1800) &␣
 ↪(col("Avg_ELO_category") <= 1999), "very good club player")
                              .when((col("Avg_ELO_category") >= 2000) &␣
 ↪(col("Avg_ELO_category") <= 2399), "national and international level")
                              .when((col("Avg_ELO_category") >= 2400) &␣
 ↪(col("Avg_ELO_category") <= 2800), "GMI, World Champions")
                              .when((col("Avg_ELO_category") < 1200), "other␣
 ↪lower bound")
                              .otherwise("other upper bound")
                              )
```

[18]:
```
# vérifier combien de "other ..."
other_lower_bound_black = df_spark_plus.filter(col("Black_ELO_category") ==␣
 ↪"other lower bound").count()
other_lower_bound_white = df_spark_plus.filter(col("White_ELO_category") ==␣
 ↪"other lower bound").count()
other_upper_bound_black = df_spark_plus.filter(col("Black_ELO_category") ==␣
 ↪"other upper bound").count()
other_upper_bound_white = df_spark_plus.filter(col("White_ELO_category") ==␣
 ↪"other upper bound").count()
```

```
print(f"Nombre de parties avec other lower bound pour le joueur noir : {other_lower_bound_black}")
print(f"Nombre de parties avec other lower bound pour le joueur blanc : {other_lower_bound_white}")
print(f"Nombre de parties avec other upper bound pour le joueur noir : {other_upper_bound_black}")
print(f"Nombre de parties avec other upper bound pour le joueur blanc : {other_upper_bound_white}")
```

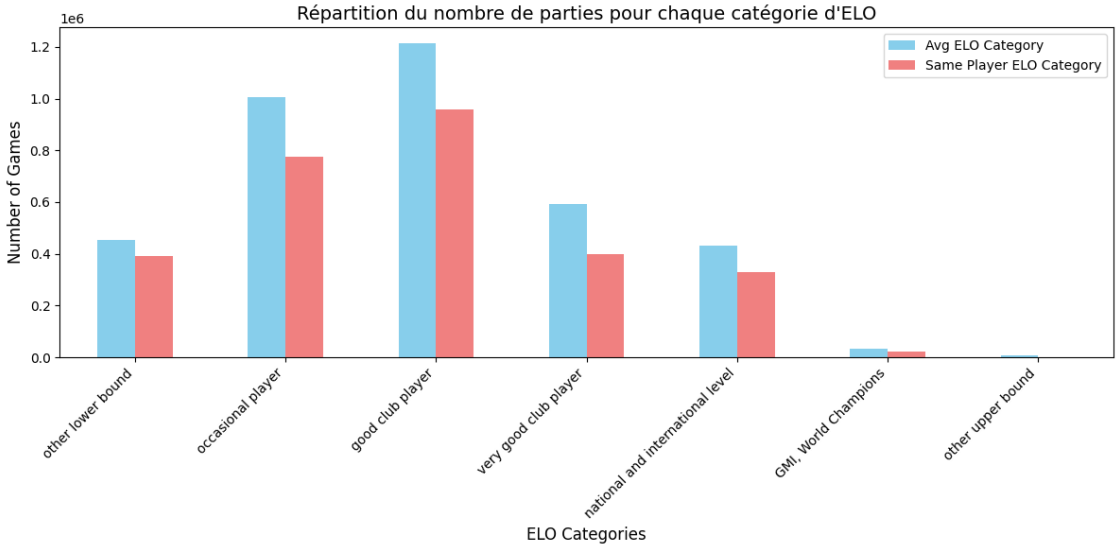
```
Nombre de parties avec other lower bound pour le joueur noir : 474566
Nombre de parties avec other lower bound pour le joueur blanc : 477997
Nombre de parties avec other upper bound pour le joueur noir : 1730
Nombre de parties avec other upper bound pour le joueur blanc : 1721
```

[19]:
```
# répartition du nombre de parties pour avg
avg_other_lower_bound = df_spark_plus.filter(col("Avg_ELO_category") == "other lower bound").count()
avg_occasional_player = df_spark_plus.filter(col("Avg_ELO_category") == "occasional player").count()
avg_good_club_player = df_spark_plus.filter(col("Avg_ELO_category") == "good club player").count()
avg_very_good_club_player = df_spark_plus.filter(col("Avg_ELO_category") == "very good club player").count()
avg_national_international_level = df_spark_plus.filter(col("Avg_ELO_category") == "national and international level").count()
avg_GMI_World_Champions = df_spark_plus.filter(col("Avg_ELO_category") == "GMI, World Champions").count()
avg_other_upper_bound = df_spark_plus.filter(col("Avg_ELO_category") == "other upper bound").count()

# répartition du nombre de parties quand les 2 joueurs sont dans la même catégorie
same_player_other_lower_bound = df_spark_plus.filter((col("Black_ELO_category") == "other lower bound") & (col("White_ELO_category") == "other lower bound")).count()
same_player_occasional_player = df_spark_plus.filter((col("Black_ELO_category") == "occasional player") & (col("White_ELO_category") == "occasional player")).count()
same_player_good_club_player = df_spark_plus.filter((col("Black_ELO_category") == "good club player") & (col("White_ELO_category") == "good club player")).count()
same_player_very_good_club_player = df_spark_plus.filter((col("Black_ELO_category") == "very good club player") & (col("White_ELO_category") == "very good club player")).count()
```

```
same_player_national_international_level = df_spark_plus.
 ↪filter((col("Black_ELO_category") == "national and international level") &␣
 ↪(col("White_ELO_category") == "national and international level")).count()
same_player_GMI_World_Champions = df_spark_plus.
 ↪filter((col("Black_ELO_category") == "GMI, World Champions") &␣
 ↪(col("White_ELO_category") == "GMI, World Champions")).count()
same_player_other_upper_bound = df_spark_plus.filter((col("Black_ELO_category")␣
 ↪== "other upper bound") & (col("White_ELO_category") == "other upper␣
 ↪bound")).count()
```

[20]:
```
# Visualiser la répartition des catégories
categories = [
    "other lower bound", "occasional player", "good club player",
    "very good club player", "national and international level", "GMI, World␣
 ↪Champions",
    "other upper bound"
]
avg_counts = [
    avg_other_lower_bound, avg_occasional_player, avg_good_club_player,
    avg_very_good_club_player, avg_national_international_level,␣
 ↪avg_GMI_World_Champions,
    avg_other_upper_bound
]
same_player_counts = [
    same_player_other_lower_bound, same_player_occasional_player,␣
 ↪same_player_good_club_player,
    same_player_very_good_club_player,␣
 ↪same_player_national_international_level, same_player_GMI_World_Champions,
    same_player_other_upper_bound
]

df_counts = pd.DataFrame({
    'Category': categories,
    'Avg_ELO_category': avg_counts,
    'Same_Player_ELO_category': same_player_counts
})

plt.figure(figsize=(12, 6))
df_counts.set_index('Category')[['Avg_ELO_category',␣
 ↪'Same_Player_ELO_category']].plot(kind='bar', ax=plt.gca(),␣
 ↪color=['skyblue', 'lightcoral'])
plt.xlabel('ELO Categories', fontsize=12)
plt.ylabel('Number of Games', fontsize=12)
plt.title('Répartition du nombre de parties pour chaque catégorie d\'ELO',␣
 ↪fontsize=14)
plt.xticks(rotation=45, ha='right')
```

```
plt.legend(['Avg ELO Category', 'Same Player ELO Category'])
plt.tight_layout()
plt.show()
```



La répartition des données, bien que déséquilibrée entre les catégories, semble réaliste.

En effet, le nombre de parties avec des joueurs de niveaux intermédiaires est plus élevé, car ils représentent la plupart des joueurs actifs qui jouent régulièrement. Au contraire, le nombre de parties avec des joueurs ayant des niveaux extrêmes (faible ou élevé) est plus faible. Nous pouvons expliquer cela par le fait que les joueurs de bas niveau évoluent rapidement ou ne jouent pas beaucoup de parties, et les joueurs avec de hauts niveaux sont plus rares dû à la difficulté d'atteindre ces niveaux.

Ce déséquilibre naturel pourrait introduire un biais dans l'analyse, car certaines catégories sont sur/sous-représentées.

Récupérer le nombre de mouvements par joueur

[21]:
```
# Compter le nombre de moves pour chaque joueur (c'est White qui commence)
df_spark_plus = df_spark_plus.withColumn("white_moves",
                                         when(col("Total_moves") % 2 == 0,␣
 ↪col("Total_moves") / 2)
                                         .otherwise(floor(col("Total_moves") /␣
 ↪2) + 1)
                                         )
df_spark_plus = df_spark_plus.withColumn("black_moves",
                                         when(col("Total_moves") % 2 == 0,␣
 ↪col("Total_moves") / 2)
                                         .otherwise(floor(col("Total_moves") /␣
 ↪2))
```

```
)
```

[22]:
```
df_spark_plus.select("Total_moves", "white_moves", "black_moves").show(5)
```

```
+-----------+-----------+-----------+
|Total_moves|white_moves|black_moves|
+-----------+-----------+-----------+
|         66|       33.0|       33.0|
|         64|       32.0|       32.0|
|         70|       35.0|       35.0|
|         86|       43.0|       43.0|
|         71|       36.0|       35.0|
+-----------+-----------+-----------+
only showing top 5 rows
```

Exploration des données

Valeurs NULL

[23]:
```
# Calcule valeurs null par colonnes
null_counts = df_spark.select(
    *[
        count(when(col(c).isNull(), c)).alias(c)
        for c in df_spark.columns
    ]
)

null_counts.show()
```

```
+----+--------+--------------+----+---+-----+-------+------+----+-----------+--
---------+-------+--------+--------------+-----------------+-----------------
+-------------+---------+---------+-----------+-------------+-------------+---
-----------+--------------+-----------------+-----------------+---------------
-----+--------------------+--------------+--------------+-----------------+-----
------------+-----------------+----------------+----------------+--------------
-+--------------------+--------------------+----------+-------------+
|GAME|BlackElo|BlackRatingDiff|Date|ECO|Event|Opening|Result|Site|Termination|Ti
meControl|UTCTime|WhiteElo|WhiteRatingDiff|Black_elo_category|White_elo_category
|starting_time|increment|Game_type|Total_moves|Black_blunders|White_blunders|Bla
ck_mistakes|White_mistakes|Black_inaccuracies|White_inaccuracies|Black_inferior_
moves|White_inferior_moves|Black_ts_moves|White_ts_moves|Black_ts_blunders|White
_ts_blunders|Black_ts_mistakes|White_ts_mistake|Black_long_moves|White_long_move
s|Black_bad_long_moves|White_bad_long_moves|Game_flips|Game_flips_ts|
+----+--------+--------------+----+---+-----+-------+------+----+-----------+--
---------+-------+--------+--------------+-----------------+-----------------
+-------------+---------+---------+-----------+-------------+-------------+---
-----------+--------------+-----------------+-----------------+---------------
-----+--------------------+--------------+--------------+-----------------+-----
```

```
-----------+----------------+---------------+---------------+--------------
-+--------------------+--------------------+----------+-------------+
|   0|       0|          13036|   0|  0|    0|      0|     0|   0|          0|
0|      0|       0|          13057|                 0|                  0|
10344|    10344|        0|          0|             0|             0|
0|             0|                0|                 0|                  0|
0|             0|             0|                0|                0|
0|               0|               0|               0|                  0|
0|         0|            0|
+----+--------+---------------+----+---+-----+-------+------+----+-----------+--
---------+-------+--------+---------------+-----------------+-----------------
+-------------+---------+---------+-----------+--------------+--------------+---
----------+-------------+-----------------+-----------------+--------------
-----+--------------------+--------------+--------------+-----------------+-----
-----------+----------------+---------------+---------------+--------------
-+--------------------+--------------------+----------+-------------+
```

Colonnes Opening et ECO Les colonnes "Opening" et "ECO" correspondent aux ouvertures et aux codes d'ouvertures, nous regardons si elles sont en liens.

[24]:
```
# Nombre valeurs opening
print(f"Nombre de valeurs opening : {df_spark.select('Opening').distinct().
  ↪count()}")
print(f"Nombre de valeurs ECO : {df_spark.select('ECO').distinct().count()}")
```

```
Nombre de valeurs opening : 2790
Nombre de valeurs ECO : 492
```

[25]:
```
#  Checker si une valeur de Opening = une valeur de ECO
alignment_check_1 = df_spark.groupBy("ECO").agg(countDistinct("Opening").
  ↪alias("Unique_Openings"))
misaligned_rows_1 = alignment_check_1.filter(col("Unique_Openings") > 1)
```

[26]:
```
# Afficher les résultats
if misaligned_rows_1.count() > 0:
    print("Il existe plusieurs Openings pour un code ECO.")
    misaligned_rows_1.show(5)
    print("Nombre de lignes : ", misaligned_rows_1.count())
else:
    print("Il existe un seul Opening pour un code ECO.")
```

```
Il existe plusieurs Openings pour un code ECO.
+---+---------------+
|ECO|Unique_Openings|
+---+---------------+
|B05|              4|
|B79|              2|
```

```
|A47|              3|
|E83|              3|
|B34|              3|
+---+---------------+
only showing top 5 rows

Nombre de lignes :  353
```

[27]:
```
# Checker si une valeur de ECO = une valeur de Opening
alignment_check_2 = df_spark.groupBy("Opening").agg(countDistinct("ECO").
 ↪alias("Unique_ECOs"))
misaligned_rows_2 = alignment_check_2.filter(col("Unique_ECOs") > 1)
```

[28]:
```
# Afficher les résultats
if misaligned_rows_2.count() > 0:
    print("Il existe plusieurs ECO pour un code Opening.")
    misaligned_rows_2.show(5)
    print("Nombre de lignes : ", misaligned_rows_2.count())
else:
    print("Il existe un seul ECO pour un code Opening.")
```

```
Il existe plusieurs ECO pour un code Opening.
+--------------------+-----------+
|             Opening|Unique_ECOs|
+--------------------+-----------+
|  St. George Defense|          2|
|Nimzo-Indian Defe…|          5|
|        Torre Attack|          2|
|   Caro-Kann Defense|          4|
|Sicilian Defense:…|          2|
+--------------------+-----------+
only showing top 5 rows

Nombre de lignes :  95
```

Nous remarquons qu'un Opening peut avoir plusieurs ECO, et qu'un ECO peut également avoir plusieurs Opening.
Nous allons regarder si c'est normal.

[29]:
```
# Filtrer pour les ouvertures ayant plusieurs ECO
misaligned_rows_2 = alignment_check_2.filter(col("Unique_ECOs") > 1)
opening_eco_counts = df_spark.groupBy("Opening", "ECO").agg(count("*").
 ↪alias("count"))
multiple_opening_eco_counts = misaligned_rows_2.join(opening_eco_counts,␣
 ↪on="Opening", how="inner")
multiple_opening_eco_counts.orderBy("Opening", "count", ascending=False).
 ↪show(truncate=False)
```

```
+-------------------------------------------------------------------------+-----
```

```
-----+---+-----+
|Opening
|Unique_ECOs|ECO|count|
+-------------------------------------------------------------------------+------
-----+---+-----+
|Vienna Game: Vienna Gambit                                               |2
|C28|3648 |
|Vienna Game: Vienna Gambit                                               |2
|C25|1193 |
|Torre Attack                                                             |2
|A46|1704 |
|Torre Attack                                                             |2
|A48|1587 |
|St. George Defense                                                       |2
|B00|4265 |
|St. George Defense                                                       |2
|C00|690  |
|Slav Defense: Exchange Variation                                         |2
|D10|5637 |
|Slav Defense: Exchange Variation                                         |2
|D13|1076 |
|Sicilian Defense: Scheveningen Variation, Classical Variation            |2
|B84|944  |
|Sicilian Defense: Scheveningen Variation, Classical Variation            |2
|B83|111  |
|Sicilian Defense: Richter-Rauzer Variation, Neo-Modern Variation         |2
|B67|90   |
|Sicilian Defense: Richter-Rauzer Variation, Neo-Modern Variation         |2
|B68|17   |
|Sicilian Defense: Richter-Rauzer Variation, Modern Variation             |2
|B60|58   |
|Sicilian Defense: Richter-Rauzer Variation, Modern Variation             |2
|B61|51   |
|Sicilian Defense: Richter-Rauzer Variation, Classical Variation          |3
|B63|124  |
|Sicilian Defense: Richter-Rauzer Variation, Classical Variation          |3
|B64|23   |
|Sicilian Defense: Richter-Rauzer Variation, Classical Variation          |3
|B65|15   |
|Sicilian Defense: Richter-Rauzer Variation                               |2
|B60|360  |
|Sicilian Defense: Richter-Rauzer Variation                               |2
|B62|10   |
|Sicilian Defense: Paulsen Variation, Bastrikov Variation, English Attack|2
|B48|242  |
+-------------------------------------------------------------------------+------
-----+---+-----+
only showing top 20 rows
```

Cela ne semble pas être des erreurs, il n'y a pas de ECO ou Opening largement dominant, nous allons donc garder ces éléments comme cela et les considérer comme 2 colonnes distinctes, n'ayant pas de lien particulier.

Après quelques recherche sur internet, nous avons constaté que ce sont bien deux colonnes distinctes. Pour la question 2, comme aucune indication n'est mentionnée dans l'énoncé, nous utiliserons seulement Opening.

Colonnes starting_time, increment, et TimeControl Nous allons maintenant vérifier si les colonnes "starting_time", "increment", et "TimeControl" sont bien en accord avec la documentation.

[30]:
```
# Même nombre null pour starting_time et increment, on vérifie que c'est aligné
df_spark.filter(col("starting_time").isNull() & col("increment").isNull()).
 ↪count()
```

[30]: 10344

[31]:
```
# Afficher les type de game quand ces 2 colonnes sont NULL
df_spark.filter(col("starting_time").isNull() & col("increment").isNull()).
 ↪select("Game_type").distinct().show()
```

```
+--------------+
|     Game_type|
+--------------+
|Correspondence|
+--------------+
```

[32]:
```
# Afficher les parties avec type de jeu Correspondence et starting_time ou␣
 ↪increment non null
df_spark.filter((col("Game_type") == "Correspondence") & (col("starting_time").
 ↪isNotNull() | col("increment").isNotNull())).count()
```

[32]: 0

Cela correspond à ce qui est attendu.

Maintenant, nous vérifions que TimeControl correspond bien à `starting_time+increment`.

[33]:
```
# Extraire starting_time et increment à partir de TimeControl
df_spark_check = df_spark.withColumn(
    "starting_time_extracted",
    when(col("TimeControl") != "-", split(col("TimeControl"), "\+")[0].
 ↪cast("int"))
    .otherwise(None))

df_spark_check = df_spark_check.withColumn(
```

```
    "increment_extracted",
    when(col("TimeControl") != "-", split(col("TimeControl"), "\+")[1].
 ↪cast("int"))
    .otherwise(None))

# Recréer TimeControl avec les colonnes extraites
df_spark_check = df_spark_check.withColumn(
    "TimeControl_reconstructed",
    when(col("starting_time_extracted").isNull() & col("increment_extracted").
 ↪isNull(), "-")
    .otherwise(concat_ws("+", col("starting_time_extracted"),␣
 ↪col("increment_extracted")))
)

# Comparer TimeControl avec la recréation
df_spark_check = df_spark_check.withColumn(
    "is_matching",
    col("TimeControl") == col("TimeControl_reconstructed")
)

# Checker les résultats
df_spark_check.select("TimeControl", "starting_time", "increment",␣
 ↪"starting_time_extracted", "increment_extracted",␣
 ↪"TimeControl_reconstructed", "is_matching").show(5)
mismatch_count = df_spark_check.filter(col("is_matching") == False).count()
print(f"Nombre de différences : {mismatch_count}")
```

```
+-----------+-------------+---------+----------------------+------------------
+-------------------------+-----------+
|TimeControl|starting_time|increment|starting_time_extracted|increment_extracted
|TimeControl_reconstructed|is_matching|
+-----------+-------------+---------+----------------------+------------------
+-------------------------+-----------+
|      300+0|          300|        0|                   300|
0|                   300+0|       true|
|      300+0|          300|        0|                   300|
0|                   300+0|       true|
|      300+2|          300|        2|                   300|
2|                   300+2|       true|
|      180+2|          180|        2|                   180|
2|                   180+2|       true|
|      600+0|          600|        0|                   600|
0|                   600+0|       true|
+-----------+-------------+---------+----------------------+------------------
+-------------------------+-----------+
only showing top 5 rows
```

```
Nombre de différences : 0
```

Il n'y a pas de différence, cela correspond également à la documentation.

1.4 Réponses aux questions

1.4.1 Question 1

Q1: What is the rate of blunders, errors and inaccuracies per move, per level category and on Blitz type games (Blitz type is by far the most played on these online sites). A game has two players, whose ELOs are most likely different. You will be able to classify a game into a category, either by considering the average ELO of both players, or by considering only the games where both players are in the same category.

Hypothèse : Les joueurs appartenant à des catégories plus expérimentées devraient présenter un taux d'erreurs plus faible.

[34]:
```
# Filtre les parties avec le type de jeu Blitz
df_blitz = df_spark_plus.filter(col("Game_type") == "Blitz")
df_blitz.count()
```

[34]:
```
1812120
```

Calcule des taux par partie

[35]:
```
# Calcule taux de blunders
df_blitz = df_blitz.withColumn("Black_blunders_rate", col("Black_blunders") /␣
 ↪col("black_moves")) \
                   .withColumn("White_blunders_rate", col("White_blunders") /␣
 ↪col("white_moves"))
```

[36]:
```
# Calcule taux d'errors
df_blitz = df_blitz.withColumn("Black_errors_rate", col("Black_mistakes") /␣
 ↪col("black_moves")) \
                   .withColumn("White_errors_rate", col("White_mistakes") /␣
 ↪col("white_moves"))
```

[37]:
```
# Calcule taux d'inaccuracies
df_blitz = df_blitz.withColumn("Black_inaccuracies_rate",␣
 ↪col("Black_inaccuracies") / col("black_moves")) \
                   .withColumn("White_inaccuracies_rate",␣
 ↪col("White_inaccuracies") / col("white_moves"))
```

Calcule des taux moyens par catégorie ELO (on considère le score ELO moyen des 2 joueurs)

[38]:
```
df_avg_elo_summary = df_blitz.groupBy("Avg_ELO_category").agg(
    {"Black_blunders_rate": "avg", "White_blunders_rate": "avg",
     "Black_errors_rate": "avg", "White_errors_rate": "avg",
     "Black_inaccuracies_rate": "avg", "White_inaccuracies_rate": "avg"}
).withColumnRenamed("avg(Black_blunders_rate)", "Avg_Black_blunders_rate") \
```

```
    .withColumnRenamed("avg(White_blunders_rate)", "Avg_White_blunders_rate") \
    .withColumnRenamed("avg(Black_errors_rate)", "Avg_Black_errors_rate") \
    .withColumnRenamed("avg(White_errors_rate)", "Avg_White_errors_rate") \
    .withColumnRenamed("avg(Black_inaccuracies_rate)",␣
    ↪"Avg_Black_inaccuracies_rate") \
    .withColumnRenamed("avg(White_inaccuracies_rate)",␣
    ↪"Avg_White_inaccuracies_rate")
```

[39]:
```
df_avg_elo_summary.show(5)
```

```
+--------------------+--------------------------+----------------------+------
--------------+--------------------+----------------------+-----------------
---------+
|    Avg_ELO_category|Avg_White_inaccuracies_rate|Avg_White_blunders_rate|Avg_Bl
ack_errors_rate|Avg_White_errors_rate|Avg_Black_blunders_rate|Avg_Black_inaccura
cies_rate|
+--------------------+--------------------------+----------------------+------
--------------+--------------------+----------------------+-----------------
---------+
|   other upper bound|        0.08923646820987065|   0.053883314861216475|
0.09786062988127485|  0.09316825168156774|    0.05713253835834972|
0.09422886449242146|
|   other lower bound|        0.09123664425761414|    0.09505881705406403|
0.11316469323850303|  0.10800044716165165|    0.09865934120262021|
0.09602531148536161|
|GMI, World Champions|        0.07247529070529041|    0.03077546627182289|
0.06824732477795835|  0.06620118207859484|    0.03325752004748459|
0.0774731883738546|
|very good club pl…|        0.08844571685155392|    0.04855612537788302|
0.09295170486304197|   0.0909842906123529|    0.05123485093711867|
0.09472554219716672|
|    good club player|        0.09093954331324328|    0.05938955675514127|
0.10091977180018212|  0.09810289300309284|   0.062224417146813066|
0.09697079506921225|
+--------------------+--------------------------+----------------------+------
--------------+--------------------+----------------------+-----------------
---------+
only showing top 5 rows
```

[40]:
```
df_avg_elo_summary_pandas = df_avg_elo_summary.toPandas()
```

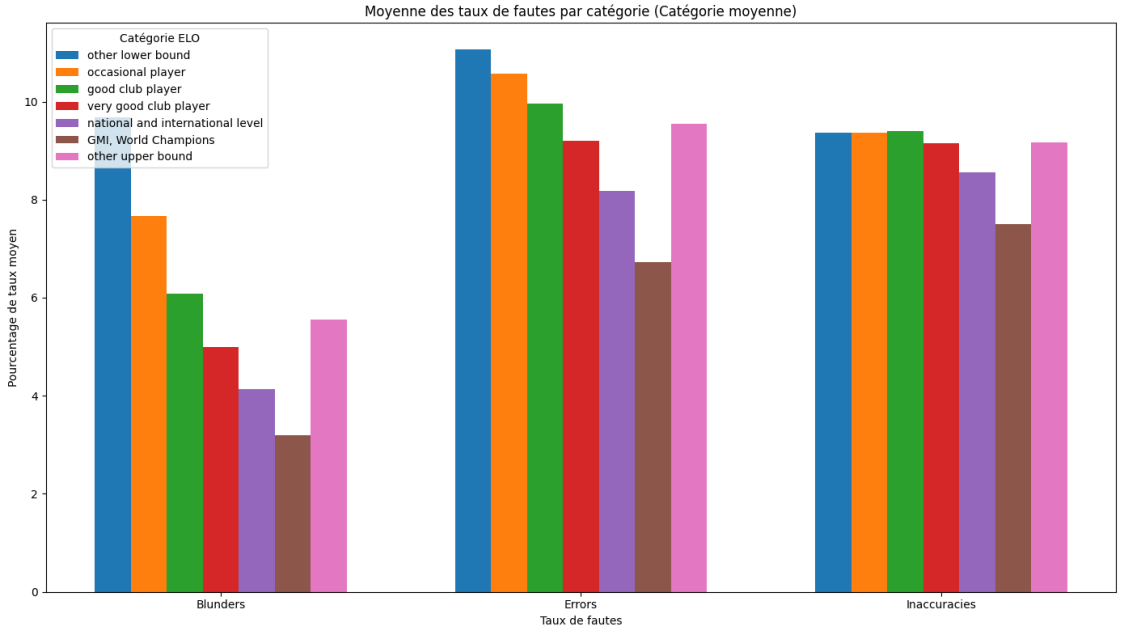
[41]:
```
df_avg_elo_summary_pandas.isna().sum()
```

[41]:
```
Avg_ELO_category              0
Avg_White_inaccuracies_rate   0
Avg_White_blunders_rate       0
Avg_Black_errors_rate         0
```

```
Avg_White_errors_rate          0
Avg_Black_blunders_rate        0
Avg_Black_inaccuracies_rate    0
dtype: int64
```

[42]:
```
# Ordonner les catégories de joueurs
category_order = ["other lower bound", "occasional player", "good club player",␣
 ↪"very good club player",
                  "national and international level", "GMI, World Champions",␣
 ↪"other upper bound"]
df_avg_elo_summary_pandas['Avg_ELO_category'] = pd.
 ↪Categorical(df_avg_elo_summary_pandas['Avg_ELO_category'], ␣
 ↪categories=category_order,  ordered=True)
df_avg_elo_summary_pandas = df_avg_elo_summary_pandas.
 ↪sort_values('Avg_ELO_category')


categories = df_avg_elo_summary_pandas['Avg_ELO_category']
error_types = ['Blunders', 'Errors', 'Inaccuracies']

# Données par type d'erreur et catégorie
blunders = df_avg_elo_summary_pandas[['Avg_Black_blunders_rate',␣
 ↪'Avg_White_blunders_rate']].mean(axis=1)*100
mistakes = df_avg_elo_summary_pandas[['Avg_Black_errors_rate',␣
 ↪'Avg_White_errors_rate']].mean(axis=1)*100
inaccuracies = df_avg_elo_summary_pandas[['Avg_Black_inaccuracies_rate',␣
 ↪'Avg_White_inaccuracies_rate']].mean(axis=1)*100

# Matrice (erreurs x catégorie)
data = np.array([blunders, mistakes, inaccuracies]).T
```

[43]:
```
# Graphique
plt.figure(figsize=(14, 8))
x = np.arange(len(error_types))
bar_width = 0.1

for i, category in enumerate(categories):
    plt.bar(x + i * bar_width, data[i], width=bar_width, label=str(category))

plt.xlabel('Taux de fautes')
plt.ylabel('Pourcentage de taux moyen')
plt.title('Moyenne des taux de fautes par catégorie (Catégorie moyenne)')
plt.xticks(x + bar_width * (len(categories) - 1) / 2, error_types)
plt.legend(title="Catégorie ELO")
plt.tight_layout()
plt.show()
```



Analyse des taux de blunders, errors et inaccuracies selon les catégories d'ELO Lorsque l'on analyse les taux de blunders, errors et inaccuracies en fonction des différentes catégories ELO, plusieurs tendances intéressantes émergent.
En excluant la catégorie "Other Upper Bound", on observe une diminution progressive des taux de **bourdes** (blunders) à mesure que les catégories augmentent, avec une chute de moins en moins marquée. Cela suggère qu'il y a une amélioration du niveau des joueurs en fonction de leur catégorie ELO, bien qu'elle soit de moins en moins évidente pour les catégories les plus expérimentées.

Concernant les taux d'**erreurs** (errors), la diminution est relativement constante à travers les catégories, mais on remarque une chute plus importante lorsque l'on atteint les deux meilleures catégories ("National and International Level" et "GMI, World Champions"). Cela indique une amélioration notable dans la gestion des erreurs pour les joueurs de niveau supérieur.

Quant aux taux d'**imprécisions** (inaccuracies), la tendance reste assez stable pour les premières catégories, avec une diminution qui s'accélère à mesure que l'on approche des deux catégories les plus élevées. Cette tendance suggère également une amélioration du jeu des joueurs plus expérimentés.

En résumé, **les taux diminuent sensiblement à mesure que l'on progresse dans les catégories ELO**, avec une amélioration plus marquée pour les catégories "National and International Level" et "GMI, World Champions", ce qui reflète probablement un meilleur contrôle stratégique et une plus grande expérience des joueurs.

Analyse de la catégorie "Other Upper Bound" Lorsque l'on considère la catégorie "Other Upper Bound", une tendance différente se dessine.
Les taux de blunders, errors et inaccuracies semblent augmenter pour atteindre une **valeur entre celle des catégories "Good Club Player" et "Very Good Club Player"**.

Cette anomalie pourrait suggérer plusieurs pistes d'interprétation.

Tout d'abord, il est possible que cette catégorie contienne des **données** qui ne sont **pas représentatives** du reste des catégories en raison d'un échantillon trop faible ou non nettoyé correctement.

Une autre hypothèse pourrait être que la **performance** des joueurs dans cette catégorie est **influencée** par la moyenne des ELO des deux joueurs, et non seulement par l'ELO individuel.
Par exemple, un joueur avec un ELO élevé qui affronte un adversaire de niveau inférieur pourrait être amené à prendre plus de risques ou adopter des stratégies différentes, ce qui expliquerait certains résultats.
De plus, il est possible qu'une partie classée dans la catégorie "Other upper bound" n'inclue qu'un seul joueur avec un ELO très élevé, ce qui fausse la moyenne des deux scores ELO et pourrait influencer l'analyse des performances.

Calcule des taux moyens par catégorie ELO (ici on considère seulement les parties où les joueurs sont dans la même catégorie)

[44]:
```
df_same_category = df_blitz.filter(col("Black_ELO_category") ==␣
 ↪col("White_ELO_category"))
```

[45]:
```
tot_blitz = df_blitz.count()
tot_same_cat = df_same_category.count()
print(f"Nombre de parties total : {tot_blitz}")
print(f"Nombre de parties avec 2 joueurs de la même catégorie : {tot_blitz}")
print(f"Pourcentage même catégorie : {tot_same_cat / tot_blitz * 100} %")
```

```
Nombre de parties total : 1812120
Nombre de parties avec 2 joueurs de la même catégorie : 1812120
Pourcentage même catégorie : 77.09219036266914 %
```

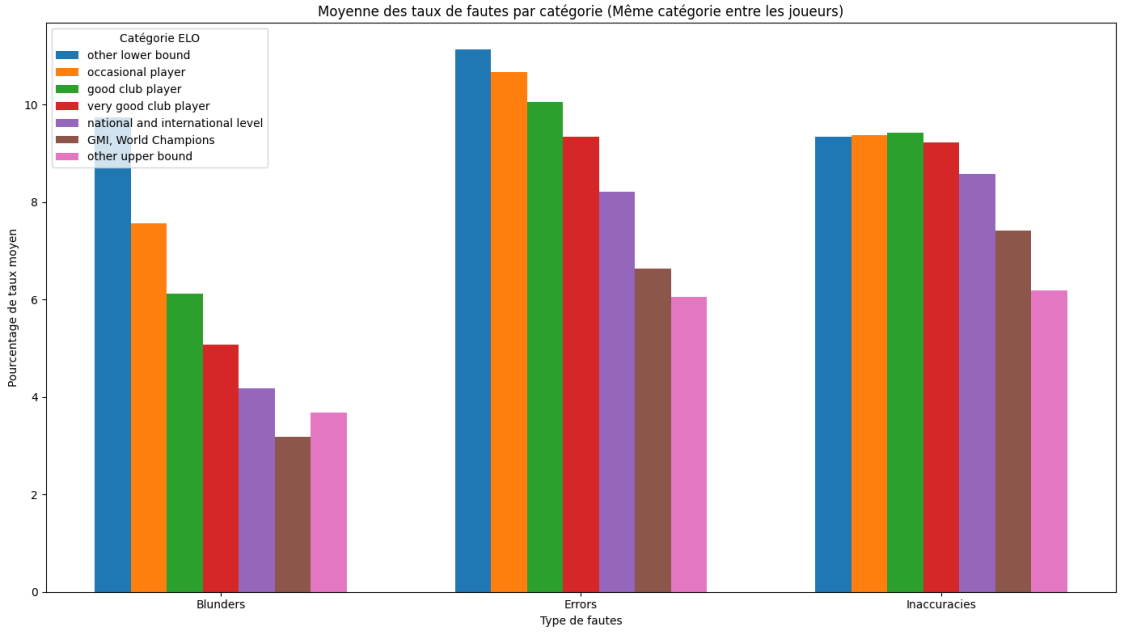
[46]:
```
df_same_category_summary = df_same_category.groupBy("Black_ELO_category").agg(
    {"Black_blunders_rate": "avg", "White_blunders_rate": "avg",
     "Black_errors_rate": "avg", "White_errors_rate": "avg",
     "Black_inaccuracies_rate": "avg", "White_inaccuracies_rate": "avg"}
).withColumnRenamed("avg(Black_blunders_rate)", "Avg_Black_blunders_rate") \
 .withColumnRenamed("avg(White_blunders_rate)", "Avg_White_blunders_rate") \
 .withColumnRenamed("avg(Black_errors_rate)", "Avg_Black_errors_rate") \
 .withColumnRenamed("avg(White_errors_rate)", "Avg_White_errors_rate") \
 .withColumnRenamed("avg(Black_inaccuracies_rate)",␣
 ↪"Avg_Black_inaccuracies_rate") \
 .withColumnRenamed("avg(White_inaccuracies_rate)",␣
 ↪"Avg_White_inaccuracies_rate")
```

[47]:
```
df_same_category_summary_pandas = df_same_category_summary.toPandas()
```

[48]:
```
# Ordonner les catégories de joueurs
category_order = ["other lower bound", "occasional player", "good club player",␣
 ↪"very good club player",
                  "national and international level", "GMI, World Champions",␣
 ↪"other upper bound"]
```

```
df_same_category_summary_pandas['Black_ELO_category'] = pd.
 ↪Categorical(df_same_category_summary_pandas['Black_ELO_category'], ␣
 ↪categories=category_order,  ordered=True)
df_same_category_summary_pandas = df_same_category_summary_pandas.
 ↪sort_values('Black_ELO_category')


categories = df_same_category_summary_pandas['Black_ELO_category']
error_types = ['Blunders', 'Errors', 'Inaccuracies']

# Données par type d'erreur et catégorie (moyenne)
blunders = df_same_category_summary_pandas[['Avg_Black_blunders_rate',␣
 ↪'Avg_White_blunders_rate']].mean(axis=1)*100
mistakes = df_same_category_summary_pandas[['Avg_Black_errors_rate',␣
 ↪'Avg_White_errors_rate']].mean(axis=1)*100
inaccuracies = df_same_category_summary_pandas[['Avg_Black_inaccuracies_rate',␣
 ↪'Avg_White_inaccuracies_rate']].mean(axis=1)*100

# Matrice (erreurs x catégorie)
data = np.array([blunders, mistakes, inaccuracies]).T
```

[49]:
```
# Graphique
plt.figure(figsize=(14, 8))
x = np.arange(len(error_types))
bar_width = 0.1
for i, category in enumerate(categories):
    plt.bar(x + i * bar_width, data[i], width=bar_width, label=str(category))

plt.xlabel('Type de fautes')
plt.ylabel('Pourcentage de taux moyen')
plt.title('Moyenne des taux de fautes par catégorie (Même catégorie entre les␣
 ↪joueurs)')
plt.xticks(x + bar_width * (len(categories) - 1) / 2, error_types)
plt.legend(title="Catégorie ELO")
plt.tight_layout()
plt.show()
```



Pour ces parties, où les deux joueurs appartiennent à la même catégorie, **les observations sont similaires** à celles faites pour la moyenne des ELO des joueurs.

Cependant, pour la catégorie **"Other upper bound"**, on ne constate **pas de réaugmentation** des erreurs et inexactitudes, ce qui confirme que la moyenne des ELO des deux joueurs influençait les résultats dans cette catégorie.

Inversement, une légère réaugmentation des blunders est observée entre les catégories "National and International level" et "GMI, World Champions".

Résultats globaux Nous observons des **résultats similaires** entre les parties où les catégories sont basées sur la moyenne des ELO des deux joueurs et celles où les deux joueurs appartiennent à la même catégorie ELO. Nous voyons, en effet, que tous **les taux de fautes ont tendance à diminuer** à mesure que la catégorie ELO augmente.

Ce résultat est en accord avec l'hypothèse initiale : Les joueurs appartenant à des catégories plus expérimentées devraient présenter un taux d'erreurs plus faible.

1.4.2 Question 2

Q2: Win probability depending on opening:

Hypothèse : L'opening choisi influence significativement les chances de victoire pour les blancs et les noirs. Certains openings sont particulièrement avantageux pour les blancs tandis que d'autres peuvent mieux convenir aux noirs, en fonction du niveau des joueurs et du type de jeu (Blitz, Rapide, Classique).

Aux échecs, l'opening désigne les premiers coups joués par chaque joueur. Ces coups établissent la structure de la partie et influencent grandement les stratégies ultérieures. Les blancs ont un

avantage naturel en débutant la partie, mais cet avantage peut être renforcé ou annulé en fonction de l'opening choisi par les deux joueurs.

****Q2a: With which opening does White have the best chance to win, by level category (*) and by type of game (Blitz, Fast, Classic).** Premières observations :**
Nous avons constaté que certaines configurations n'étaient jouées que très rarement et aboutissaient systématiquement à une victoire des Blancs.
Cela introduit un biais et ne permet pas d'identifier correctement quel opening offre réellement le plus de chances de gagner.
En effet, plus de 3 800 configurations présentaient une White_win_probability égale à 1.

Nous avons donc décidé de conserver uniquement les configurations avec un nombre de parties jouées supérieure à 100 afin d'obtenir des résultats plus pertinents.

Nous allons analyser les configurations possibles, c'est à dire les combinaisons de `Opening`, `White_ELO_category`, et `Game_type`.

[50]:
```
# Calculer le nombre de parties pour chaque configuration
config_game_counts = df_spark_plus.groupBy("Opening", "White_ELO_category",␣
 ↪"Game_type").agg(count("*").alias("Total_games_count"))
config_game_counts.orderBy("Total_games_count", ascending=False).show(5)
```

```
+--------------------+------------------+---------+-----------------+
|             Opening|White_ELO_category|Game_type|Total_games_count|
+--------------------+------------------+---------+-----------------+
|Queen's Pawn Game…| occasional player|    Blitz|            11963|
|Queen's Pawn Game…|  good club player|    Blitz|            11456|
|    Philidor Defense| occasional player|    Blitz|            10246|
|    Philidor Defense|  good club player|    Blitz|             9800|
|    Sicilian Defense|  good club player|    Blitz|             9702|
+--------------------+------------------+---------+-----------------+
only showing top 5 rows
```

[51]:
```
# Nombre total de configurations uniques
print(f"Nombre total de configurations possibles : {config_game_counts.
 ↪count()}")
print(f"Nombre de configurations possibles pour les différents types de jeux :")
config_game_counts.groupBy("Game_type").count().orderBy("count",␣
 ↪ascending=False).show()
```

```
Nombre total de configurations possibles : 44438
Nombre de configurations possibles pour les différents types de jeux :
+-------------+-----+
|    Game_type|count|
+-------------+-----+
|        Blitz|12957|
|       Bullet|11401|
|        Rapid|10377|
```

```
|     Classical| 6892|
|Correspondence| 2811|
+--------------+-----+
```

[52]:
```
# Nombre d'Opening par configurations
df_spark_plus.groupBy("White_ELO_category", "Game_type").count().
 ↪orderBy("count", ascending=False).show(34)
```

```
+--------------------+--------------+------+
|  White_ELO_category|     Game_type| count|
+--------------------+--------------+------+
|    good club player|         Blitz|549747|
|   occasional player|         Blitz|450990|
|    good club player|         Rapid|350294|
|very good club pl…|         Blitz|287985|
|   occasional player|         Rapid|262582|
|national and inte…|         Blitz|254839|
|    good club player|        Bullet|245115|
|   other lower bound|         Blitz|243316|
|   occasional player|        Bullet|227062|
|very good club pl…|         Rapid|162121|
|   other lower bound|        Bullet|132848|
|very good club pl…|        Bullet|106572|
|   other lower bound|         Rapid| 95046|
|national and inte…|         Rapid| 94355|
|national and inte…|        Bullet| 81839|
|    good club player|     Classical| 63684|
|   occasional player|     Classical| 31710|
|very good club pl…|     Classical| 29430|
|GMI, World Champions|         Blitz| 24942|
|national and inte…|     Classical| 12934|
|GMI, World Champions|        Bullet| 11352|
|   other lower bound|     Classical|  6686|
|    good club player|Correspondence|  5128|
|very good club pl…|Correspondence|  2366|
|GMI, World Champions|         Rapid|  2163|
|   occasional player|Correspondence|  1597|
|   other upper bound|        Bullet|  1411|
|national and inte…|Correspondence|  1113|
|   other upper bound|         Blitz|   301|
|GMI, World Champions|     Classical|   232|
|   other lower bound|Correspondence|   101|
|GMI, World Champions|Correspondence|    39|
|   other upper bound|         Rapid|     8|
|   other upper bound|     Classical|     1|
+--------------------+--------------+------+
```

Nous pouvons voir que le nombre de parties jouées par type de jeux n'est pas répartie de façon uniforme, et que certaines configurations sont très sous représentées.

[53]:
```
# Filtrer les configurations avec plus de 100 parties jouées
filtered_configurations = config_game_counts.filter(col("Total_games_count") >␣
↪100)
filtered_configurations.orderBy("Total_games_count", ascending=False).show(5)
print(f"Nombre de configurations avec plus de 100 parties jouées :␣
↪{filtered_configurations.count()}")
```

```
+--------------------+-----------------+---------+-----------------+
|             Opening|White_ELO_category|Game_type|Total_games_count|
+--------------------+-----------------+---------+-----------------+
|Queen's Pawn Game…| occasional player|    Blitz|            11963|
|Queen's Pawn Game…|  good club player|    Blitz|            11456|
|    Philidor Defense| occasional player|    Blitz|            10246|
|    Philidor Defense|  good club player|    Blitz|             9800|
|    Sicilian Defense|  good club player|    Blitz|             9702|
+--------------------+-----------------+---------+-----------------+
only showing top 5 rows

Nombre de configurations avec plus de 100 parties jouées : 5955
```

[54]:
```
filtered_df = df_spark_plus.join(filtered_configurations.select("Opening",␣
↪"White_ELO_category", "Game_type"), on=["Opening", "White_ELO_category",␣
↪"Game_type"], how="inner")
```

[55]:
```
# Quels sont les différentes valeurs de Game_type ?
filtered_df.select("Game_type").distinct().show()
```

```
+--------------+
|     Game_type|
+--------------+
|        Bullet|
|         Blitz|
|     Classical|
|         Rapid|
|Correspondence|
+--------------+
```

L'énoncé précise "by type of game (Blitz, Fast, Classic)" mais on voit bien ici que c'est Blitz, "Rapid" et "Classical".

[56]:
```
# Comment sont explicités les différentes fin de partie ?
filtered_df.select("Result").distinct().show()
```

```
+-------+
| Result|
+-------+
```

```
|      *|
|1/2-1/2|
|    1-0|
|    0-1|
+-------+
```

1-0 : Victoire des blancs
0-1 : Victoire des noirs
1/2-1/2 : Match nul

[57]:
```
# Récupération des parties voulues (blancs gagnes +  type de jeux Blitz, Fast, Classic)
df_white_wins = filtered_df.filter((col("Result") == "1-0") & (col("Game_type").isin(["Blitz", "Rapid", "Classical"])))
df_total_games = filtered_df.filter(col("Game_type").isin(["Blitz", "Rapid", "Classical"]))
```

[58]:
```
# Pour chaque ouverture, catégorie et type de jeu on calcule le nombre de victoires des blancs
df_white_wins_groupby = df_white_wins.groupBy("Opening", "White_ELO_category", "Game_type").agg(count("*").alias("White_win_count"))

# Pareil mais on calcule le total de parties jouées
df_total_games_groupby = df_total_games.groupBy("Opening", "White_ELO_category", "Game_type").agg(count("*").alias("Total_games_count"))
```

[59]:
```
df_white_wins_groupby.show(5)
```

```
+---------------+-------------------+---------+---------------+
|        Opening| White_ELO_category|Game_type|White_win_count|
+---------------+-------------------+---------+---------------+
|Alekhine Defense|    good club player|    Blitz|            519|
|Alekhine Defense|    good club player|Classical|             68|
|Alekhine Defense|    good club player|    Rapid|            267|
|Alekhine Defense|national and inte…|    Blitz|            216|
|Alekhine Defense|national and inte…|    Rapid|             81|
+---------------+-------------------+---------+---------------+
only showing top 5 rows
```

[60]:
```
df_total_games_groupby.show(5)
```

```
+---------------+-------------------+---------+-----------------+
|        Opening| White_ELO_category|Game_type|Total_games_count|
+---------------+-------------------+---------+-----------------+
|Alekhine Defense|    good club player|    Blitz|             1010|
|Alekhine Defense|    good club player|Classical|              133|
|Alekhine Defense|    good club player|    Rapid|              514|
```

```
|Alekhine Defense|national and inte…|    Blitz|              424|
|Alekhine Defense|national and inte…|    Rapid|              141|
+---------------+--------------------+---------+-----------------+
only showing top 5 rows
```

[61]:
```
# Calcule de la probabilité de gagner en fonction de l'ouverture
df_opening_stats = df_white_wins_groupby.join(df_total_games_groupby,␣
 ↪on=["Opening", "White_ELO_category", "Game_type"])
df_opening_stats = df_opening_stats.withColumn("White_win_probability",␣
 ↪col("White_win_count") / col("Total_games_count"))
```

[62]:
```
df_opening_stats.show(5)
```

```
+---------------+--------------------+---------+---------------+---------------
--+----------------------+
|         Opening|  White_ELO_category|Game_type|White_win_count|Total_games_cou
nt|White_win_probability|
+---------------+--------------------+---------+---------------+---------------
--+----------------------+
|Alekhine Defense|    good club player|    Blitz|            519|
1010|   0.5138613861386139|
|Alekhine Defense|    good club player|Classical|             68|
133|   0.5112781954887218|
|Alekhine Defense|    good club player|    Rapid|            267|
514|    0.519455252918288|
|Alekhine Defense|national and inte…|    Blitz|            216|
424|   0.5094339622641509|
|Alekhine Defense|national and inte…|    Rapid|             81|
141|    0.574468085106383|
+---------------+--------------------+---------+---------------+---------------
--+----------------------+
only showing top 5 rows
```

[63]:
```
df_opening_stats.count()
```

[63]: 4548

[64]:
```
# Y a t il toutes les combinaisons de catégorie / type de partie ?
df_opening_stats.groupBy("White_ELO_category", "Game_type").count().
 ↪orderBy("count", ascending=False).show(truncate=False)
```

```
+-------------------------------+---------+-----+
|White_ELO_category             |Game_type|count|
+-------------------------------+---------+-----+
|good club player               |Blitz    |680  |
|national and international level|Blitz    |549  |
|very good club player          |Blitz    |533  |
```

```
|occasional player                |Blitz    |519  |
|good club player                 |Rapid    |494  |
|occasional player                |Rapid    |364  |
|very good club player            |Rapid    |342  |
|other lower bound                |Blitz    |332  |
|national and international level|Rapid    |237  |
|other lower bound                |Rapid    |185  |
|good club player                 |Classical|149  |
|occasional player                |Classical|69   |
|very good club player            |Classical|46   |
|GMI, World Champions             |Blitz    |32   |
|other lower bound                |Classical|9    |
|national and international level|Classical|8    |
+---------------------------------+---------+-----+
```

Nous pouvons voir que le nombre d'Opening différents (count) pour les combinaisons de catégories ELO et type de jeux n'est pas répartie uniformément, et certaines sont même absentes.

Cependant, dans le jeu de données initiale, ces configurations étaient très sous représentées. Notamment : - GMI, World Champions - Rapid - 2163 - other upper bound - Blitz - 301 - GMI, World Champions - Classical - 232 - other upper bound - Rapid - 8 - other upper bound - Classical - 1

[65]:
```
# Récupération du meilleur opening pour chaque catégorie de joueur et type de␣
 ↪partie
window_spec = Window.partitionBy("White_ELO_category", "Game_type").
 ↪orderBy(col("White_win_probability").desc())
best_openings = df_opening_stats.withColumn("rank", rank().over(window_spec))
best_openings = best_openings.filter(col("rank") == 1).
 ↪select("White_ELO_category", "Game_type", "Opening", "White_win_probability")
```

Résultats

Nous affichons maintenant pour chaque catégorie ELO et type de jeu, l'opening permettant le plus de gagner pour les blancs.

[66]:
```
best_openings.orderBy("White_ELO_category", "Game_type").show(17)
```

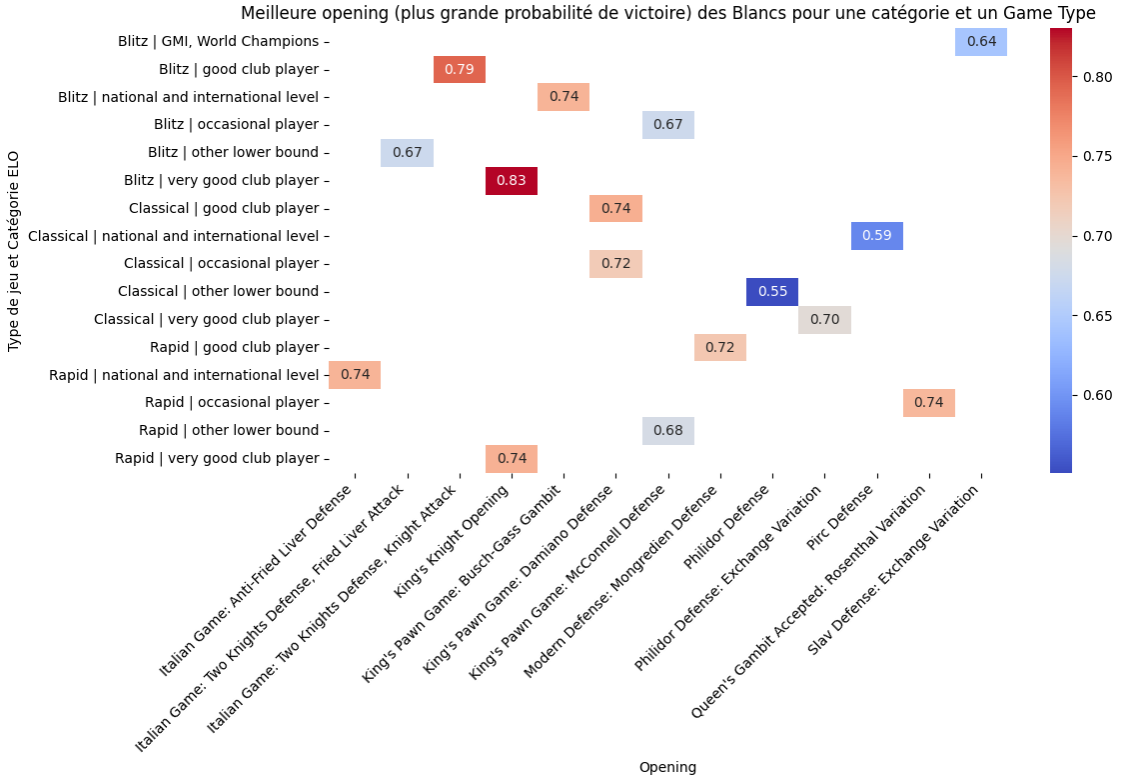
```
+--------------------+---------+--------------------+---------------------+
|  White_ELO_category|Game_type|             Opening|White_win_probability|
+--------------------+---------+--------------------+---------------------+
|GMI, World Champions|    Blitz|Slav Defense: Exc…|   0.6422764227642277|
|    good club player|    Blitz|Italian Game: Two…|   0.7928571428571428|
|    good club player|Classical|King's Pawn Game:…|   0.7448979591836735|
|    good club player|    Rapid|Modern Defense: M…|   0.7230769230769231|
|national and inte…|    Blitz|King's Pawn Game:…|    0.740506329113924|
|national and inte…|Classical|        Pirc Defense|   0.5897435897435898|
|national and inte…|    Rapid|Italian Game: Ant…|    0.740506329113924|
|   occasional player|    Blitz|King's Pawn Game:…|   0.6744868035190615|
|   occasional player|Classical|King's Pawn Game:…|     0.71900826446281|
```

```
|   occasional player|    Rapid|Queen's Gambit Ac…|   0.7368421052631579|
|   other lower bound|    Blitz|Italian Game: Two…|   0.6739811912225705|
|   other lower bound|Classical|    Philidor Defense|   0.5510204081632653|
|   other lower bound|    Rapid|King's Pawn Game:…|   0.6823266219239373|
|very good club pl…|    Blitz|King's Knight Ope…|   0.8305084745762712|
|very good club pl…|Classical|Philidor Defense:…|   0.6956521739130435|
|very good club pl…|    Rapid|King's Knight Ope…|   0.7425742574257426|
+-------------------+---------+------------------+---------------------+
```

[67]:
```
best_openings_pandas = best_openings.toPandas()
```

[68]:
```
best_openings_pandas["GameType_Category"] = (best_openings_pandas["Game_type"]␣
  ↪+ " | " + best_openings_pandas["White_ELO_category"])

# Table pivot pour voir le meilleur opening pour chaque configuration
pivot_table = best_openings_pandas.pivot_table(
    index="GameType_Category",
    columns="Opening",
    values="White_win_probability"
)
```

[69]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_table, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)
plt.title("Meilleure opening (plus grande probabilité de victoire) des Blancs␣
  ↪pour une catégorie et un Game Type")
plt.xlabel("Opening")
plt.ylabel("Type de jeu et Catégorie ELO")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```



Nous pouvons voir que l'opening obtenant un meilleur résultat est généralement différent entre les configurations. Cela pourrait montrer qu'il y a un lien entre la victoire, le type de jeu, le niveau du joueur et l'opening choisi.

Plus la probabilité de victoire associée à un opening est proche de 0.5, plus la chance de gagner avec cet opening est réduite, même si elle reste légèrement favorable (> 0.5). Cela reflète une situation où l'opening ne confère qu'un léger avantage, sans être déterminant. À l'inverse, une probabilité de victoire élevée, comme 0.83 dans le cas du "King Knight Opening" pour les joueurs de très bon niveau en Blitz, indique un opening particulièrement efficace pour maximiser les chances de victoire des blancs dans ce contexte spécifique.

Q2b: same question with black. You don't need to write again the same but only the results with black.

[70]:
```
# Calculer le nombre de parties pour chaque configuration
config_game_counts = df_spark_plus.groupBy("Opening", "Black_ELO_category",␣
↪"Game_type").agg(count("*").alias("Total_games_count"))
```

[71]:
```
# Filtrer les configurations avec plus de 100 parties jouées
filtered_configurations = config_game_counts.filter(col("Total_games_count") >␣
↪100)
```

[72]:
```
```

```
filtered_df = df_spark_plus.join(filtered_configurations.select("Opening",␣
 ↪"Black_ELO_category", "Game_type"), on=["Opening", "Black_ELO_category",␣
 ↪"Game_type"], how="inner")
```

[73]:
```
df_black_wins = filtered_df.filter((col("Result") == "0-1") & (col("Game_type").
 ↪isin(["Blitz", "Rapid", "Classical"])))
df_total_games = filtered_df.filter(col("Game_type").isin(["Blitz", "Rapid",␣
 ↪"Classical"]))
```

[74]:
```
df_black_wins_groupby = df_black_wins.groupBy("Opening", "Black_ELO_category",␣
 ↪"Game_type").agg(count("*").alias("Black_win_count"))
df_total_games_groupby = df_total_games.groupBy("Opening",␣
 ↪"Black_ELO_category", "Game_type").agg(count("*").alias("Total_games_count"))
```

[75]:
```
df_opening_stats = df_black_wins_groupby.join(df_total_games_groupby,␣
 ↪on=["Opening", "Black_ELO_category", "Game_type"])
df_opening_stats = df_opening_stats.withColumn("Black_win_probability",␣
 ↪col("Black_win_count") / col("Total_games_count"))
```

Nous affichons maintenant pour chaque catégorie ELO et type de jeu, l'opening (des blancs) permettant le plus de gagner pour les noirs.

[76]:
```
window_spec = Window.partitionBy("Black_ELO_category", "Game_type").
 ↪orderBy(col("Black_win_probability").desc())
best_openings = df_opening_stats.withColumn("rank", rank().over(window_spec))
best_openings = best_openings.filter(col("rank") == 1).
 ↪select("Black_ELO_category", "Game_type", "Opening",␣
 ↪"Black_win_probability") # Rank pour garder les égalités
best_openings.orderBy("Black_ELO_category", "Game_type").show(truncate=False)
```

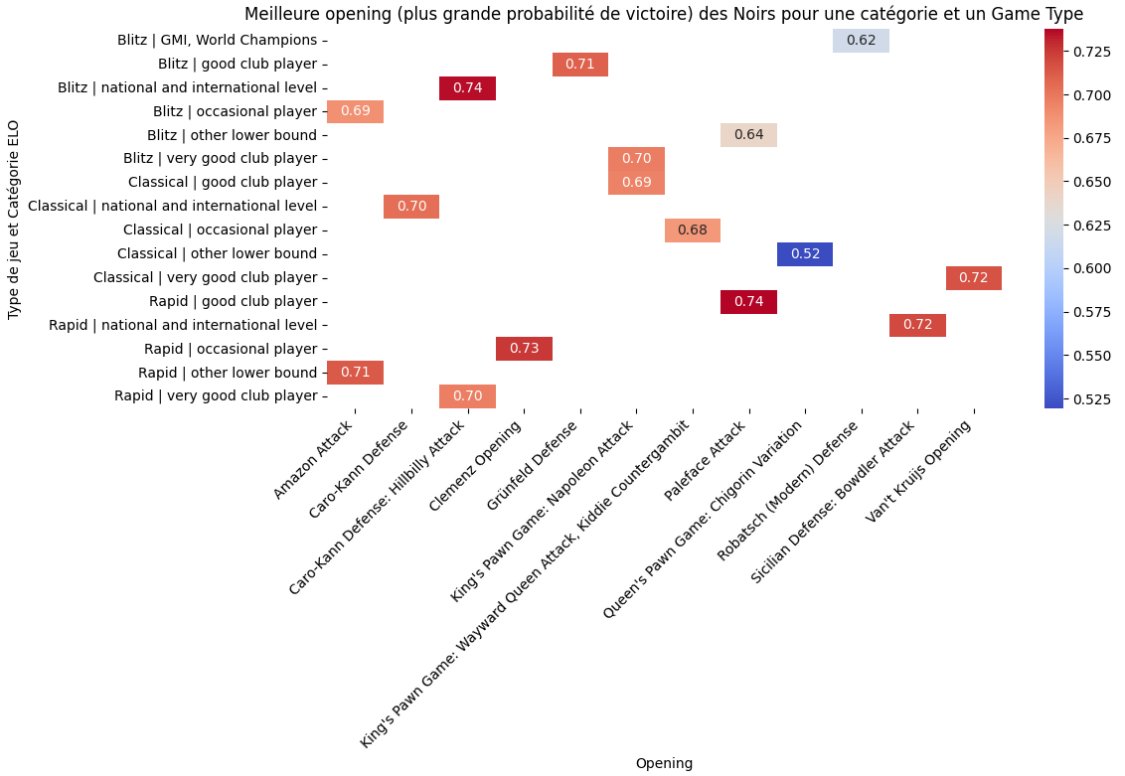
```
+-------------------------------+---------+-----------------------------------
------------------------+---------------------+
|Black_ELO_category             |Game_type|Opening
|Black_win_probability|
+-------------------------------+---------+-----------------------------------
------------------------+---------------------+
|GMI, World Champions           |Blitz    |Robatsch (Modern) Defense
|0.6190476190476191   |
|good club player               |Blitz    |Grünfeld Defense
|0.7099236641221374   |
|good club player               |Classical|King's Pawn Game: Napoleon Attack
|0.6946564885496184   |
|good club player               |Rapid    |Paleface Attack
|0.7378640776699029   |
|national and international level|Blitz    |Caro-Kann Defense: Hillbilly Attack
|0.7359550561797753   |
|national and international level|Classical|Caro-Kann Defense
|0.7043478260869566   |
```

```
|national and international level|Rapid    |Sicilian Defense: Bowdler Attack
|0.7196969696969697   |
|occasional player                |Blitz    |Amazon Attack
|0.6875               |
|occasional player                |Classical|King's Pawn Game: Wayward Queen
Attack, Kiddie Countergambit|0.6833333333333333   |
|occasional player                |Rapid    |Clemenz Opening
|0.7253886010362695   |
|other lower bound                |Blitz    |Paleface Attack
|0.6403508771929824   |
|other lower bound                |Classical|Queen's Pawn Game: Chigorin
Variation                       |0.5192307692307693   |
|other lower bound                |Rapid    |Amazon Attack
|0.7130434782608696   |
|very good club player            |Blitz    |King's Pawn Game: Napoleon Attack
|0.696969696969697    |
|very good club player            |Classical|Van't Kruijs Opening
|0.7163120567375887   |
|very good club player            |Rapid    |Caro-Kann Defense: Hillbilly Attack
|0.6964285714285714   |
+---------------------------------+---------+-----------------------------------
------------------------+--------------------+
```

[77]:
```
best_openings_pandas = best_openings.toPandas()
```

[78]:
```
best_openings_pandas["GameType_Category"] = (best_openings_pandas["Game_type"]␣
↪+ " | " + best_openings_pandas["Black_ELO_category"])
pivot_table = best_openings_pandas.pivot_table(
    index="GameType_Category",
    columns="Opening",
    values="Black_win_probability"
)
```

[79]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_table, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)
plt.title("Meilleure opening (plus grande probabilité de victoire) des Noirs␣
↪pour une catégorie et un Game Type")
plt.xlabel("Opening")
plt.ylabel("Type de jeu et Catégorie ELO")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```



Nous observons que les openings offrant le plus de chances de victoire aux noirs sont différents de ceux favorisant les blancs. De plus, l'opening associé aux meilleures chances de victoire varie également en fonction des configurations, qu'il s'agisse du type de jeu (Blitz, Rapide, Classique) ou du niveau des joueurs.

Par exemple, dans les parties classiques pour les joueurs d'un niveau "other lower bound", l'opening "*Queen's Pawn Game: Chigorin Variation*" affiche une probabilité de victoire de 0.52, ce qui suggère un léger avantage pour les noirs, bien que réduit.

Nous remarquons également que certains openings semblent efficaces à travers différents types de jeu et niveaux, démontrant une certaine polyvalence. Cela souligne l'importance de l'adaptation stratégique des joueurs selon le contexte pour maximiser leurs chances de succès.

Réponse à l'hypothèse :

- Les données montrent que l'opening choisi influence effectivement les chances de victoire, et cet impact varie selon plusieurs facteurs :

Différence entre blancs et noirs :

- Certains openings favorisent nettement les blancs, tandis que d'autres conviennent davantage aux noirs. Cela illustre une dynamique stratégique où les choix initiaux de chaque joueur influencent fortement l'évolution de la partie.

Variabilité selon les configurations :

- Dans les parties Blitz, par exemple, des openings spécifiques comme le "*King's Knight Opening*" pour les blancs offrent des probabilités de victoire élevées, atteignant 0.83. Cela suggère que la rapidité du jeu peut accentuer l'efficacité de certains openings.
- Dans des catégories comme "other lower bound" en classique, des openings comme "*Queen's Pawn Game : Chigorin Variation*" n'apportent qu'un avantage limité (0.52), montrant une moindre influence stratégique à ce niveau.

Proximité avec 0.5 :

- Plus la probabilité de victoire liée à un opening est proche de 0.5, moins cet opening semble décisif. Cela signifie qu'il joue un rôle plus neutre dans l'issue de la partie, même s'il reste un léger avantage pour l'un des camps.

Polyvalence de certains openings :

- Certains openings apparaissent comme efficaces à travers différents types de jeux et niveaux de joueurs, indiquant qu'ils peuvent être des choix stratégiques universels.

1.4.3 Question 3

Q3: (difficult). Does a line of data in the file predict the outcome of the game (column Result), and with what probability? In other words, can any of the variables, such as the number of errors (mistakes, blunders, inacurracies, ts_blunders), the difference in ELO between the two players, etc., explain the outcome (win/loss)? You are free to define explain as you wish. It can be a correlation, linear or not, or any other relationship that allows this prediction.
Note that the ELO is itself computed from a probability (normal distribution) of victory depending on the difference in ELO of the two players. For instance, for a difference of 100 ELO points, the higher ranked player is expected to win with probability 0.64. For a 200 points difference, it is 0.76.
As we have more data than the ELO difference, your prediction should be more accurate than that.

Pour répondre à cette question, nous avons adopté une approche combinant des analyses exploratoires et des techniques de machine learning afin d'évaluer la capacité des variables à expliquer ou prédire le résultat d'une partie (colonne *Result*). Cependant, la grande quantité de données disponibles a posé des défis significatifs, notamment en termes de temps d'entraînement pour certains modèles sophistiqués, comme les forêts aléatoires (*Random Forests*) ou les arbres boostés (*Gradient Boosted Trees*), même en réduisant l'échantillon à un pourcentage aléatoire des données.

Face à ces contraintes, nous avons opté pour un modèle de régression multinomiale, qui s'est avéré bien plus rapide à entraîner tout en offrant des performances acceptables sur un sous-échantillon de 1 % des données. Cela a toutefois nécessité une adaptation spécifique de la préparation des données. En parallèle, pour mieux comprendre les relations entre les variables et le résultat, nous avons complété l'analyse par des mesures de corrélation, de covariance et des tableaux de contingence (avec le test du chi2), afin de capturer des liens potentiellement explicatifs ou prédictifs entre les

caractéristiques comme les erreurs (*mistakes*, *blunders*, etc.), la différence d'ELO entre les joueurs, et d'autres variables pertinentes.

Hypothèse : L'hypothèse principale est que la différence d'ELO est un facteur significatif pour prédire l'issue de la partie, conformément à la théorie sous-jacente à son calcul (probabilité basée sur une distribution normale).

Cependant, étant donné la richesse des données, d'autres facteurs pourraient également jouer un rôle dans la prédiction de l'issue de la partie. Ces facteurs incluent :

- Nombre d'erreurs (blunders, mistakes, inaccuracies, ts_blunders) : des erreurs fréquentes devraient augmenter les chances de défaite.
- Nombre de coups totaux (Total_moves) : des parties plus longues peuvent refléter un jeu plus équilibré ou stratégique.
- Niveau de jeu (type de partie : Blitz, Classique, etc.) : les parties rapides pourraient amplifier l'effet des erreurs.
- Autres statistiques liées au jeu : comme les renversements de partie (Game_flips), etc.

Préparation des données

[80]:
```
df_spark_plus.show(5)
```

```
+----+--------+--------------+----------+---+----------------+----------------
---+------+--------------------+-----------+----------+------------------+---
-----+---------------+--------------------+--------------------+-------------+--
-------+---------+-----------+--------------+--------------+--------------+-----
---------+-----------------+-----------------+--------------------+-----------
---------+--------------+--------------+-----------------+-----------------+----
------------+----------------+---------------+----------------+--------------
-----+--------------------+----------+-------------+--------------------+-------
----+-----------+
|GAME|BlackElo|BlackRatingDiff|      Date|ECO|           Event|
Opening|Result|                Site| Termination|TimeControl|
UTCTime|WhiteElo|WhiteRatingDiff|  Black_ELO_category|  White_ELO_category|start
ing_time|increment|Game_type|Total_moves|Black_blunders|White_blunders|Black_mis
takes|White_mistakes|Black_inaccuracies|White_inaccuracies|Black_inferior_moves|
White_inferior_moves|Black_ts_moves|White_ts_moves|Black_ts_blunders|White_ts_bl
unders|Black_ts_mistakes|White_ts_mistake|Black_long_moves|White_long_moves|Blac
k_bad_long_moves|White_bad_long_moves|Game_flips|Game_flips_ts|
Avg_ELO_category|white_moves|black_moves|
+----+--------+--------------+----------+---+----------------+----------------
---+------+--------------------+-----------+----------+------------------+---
-----+---------------+--------------------+--------------------+-------------+--
-------+---------+-----------+--------------+--------------+--------------+-----
---------+-----------------+-----------------+--------------------+-----------
---------+--------------+--------------+-----------------+-----------------+----
------------+----------------+---------------+----------------+--------------
-----+--------------------+----------+-------------+--------------------+-------
----+-----------+
|  11|    1143|             6|2020.09.01|A02|Rated Blitz game|        Bird
```

```
Opening|   0-1|https://lichess.o…|Time forfeit|      300+0|2025-01-07
00:00:00|    1180|             -7|   other lower bound|   other lower bound|
300|        0|    Blitz|         66|             4|             2|
0|             3|                 3|                 1|                   7|
6|             8|             8|                0|                0|
0|               0|              2|               1|                  1|
1|         8|            0|   other lower bound|       33.0|       33.0|
|  14|    1504|           NULL|2020.09.01|A04|Rated Blitz game|        Réti
Opening|   0-1|https://lichess.o…|      Normal|      300+0|2025-01-07
00:00:00|    1381|           NULL|    good club player|   occasional player|
300|        0|    Blitz|         64|             2|             1|
1|             1|                 7|                 5|                  10|
7|             0|             0|                0|                0|
0|               0|              0|               1|                  0|
0|         6|            0|   occasional player|       32.0|       32.0|
|  29|    1933|              1|2020.09.01|C41|Rated Blitz game|    Philidor
Defense|   0-1|https://lichess.o…|Time forfeit|      300+2|2025-01-07
00:00:00|    1485|             -1|very good club pl…|   occasional player|
300|        2|    Blitz|         70|             0|             1|
1|             2|                 8|                 8|                   9|
11|             0|             2|                0|                0|
0|               0|              1|               1|                  1|
0|         5|            0|    good club player|       35.0|       35.0|
|  40|    1710|             10|2020.09.01|B23|Rated Blitz game|Sicilian
Defense:…|   0-1|https://lichess.o…|      Normal|      180+2|2025-01-07
00:00:00|    2040|            -11|    good club player|national and inte…|
180|        2|    Blitz|         86|             4|             2|
1|             5|                 3|                 4|                   8|
11|            18|             0|                4|                0|
0|               0|              3|               1|                  1|
0|         8|            1|very good club pl…|       43.0|       43.0|
|  55|    1598|             -1|2020.09.01|B03|Rated Rapid game|    Alekhine
Defense|   1-0|https://lichess.o…|      Normal|      600+0|2025-01-07
00:00:00|    2163|              0|    good club player|national and inte…|
600|        0|    Rapid|         71|             1|             0|
1|             1|                 6|                 2|                   8|
3|             0|             0|                0|                0|
0|               0|              0|               0|                  0|
0|         2|            0|very good club pl…|       36.0|       35.0|
+----+--------+---------------+----------+---+----------------+----------------
---+------+-------------------+------------+-----------+------------------+---
-----+---------------+--------------------+--------------------+-------------+--
-------+---------+-----------+--------------+--------------+--------------+-----
---------+-----------------+-----------------+--------------------+-----------
---------+-------------+--------------+-----------------+-----------------+----
------------+----------------+---------------+----------------+--------------
-----+--------------------+----------+-------------+--------------------+-------
----+-----------+
```

```
only showing top 5 rows
```

[81]:
```
# Ajouter la colonne de différence d'ELO
df_spark_plus = df_spark_plus.withColumn("ELO_diff", col("WhiteELO") -␣
 ↪col("BlackELO"))
```

Suppression des colonnes non nécessaires Nous allons supprimer certaines colonnes du jeu de données qui ne sont pas pertinentes pour la prédiction de la colonne `Result` et/ou qui pourrait introduire un biais.

Suppression des colonnes étant des conséquences du résultat:

- `BlackRatingDiff` : Variation du classement ELO du joueur noir après la partie
- `WhiteRatingDiff` : Variation du classement ELO du joueur blanc après la partie

Ces colonnes reflètent directement l'issue de la partie et ne peuvent donc pas être utilisées comme des variables explicatives pour prédire le résultat. Cela nous évite aussi de devoir gérer les valeurs NULL présentes dans ces colonnes.

Suppression des colonnes n'apportant pas d'informations pertinentes :

- `GAME` : Identifiant unique de la partie
- `Date`: Date à laquelle la partie a été jouée
- `Site`: URL de la partie

- `TimeControl` : Temps de jeu en secondes (temps initial + incrément)
- `UTCTime` : Heure à laquelle la partie a été jouée
- `Event` : Evenement où la partie a été jouée

En plus d'être non pertinentes, certaines de ces données ne sont pas standardisées, cela nous évite donc des traitements supplémentaires non nécessaires.

Supprimer les colonnes que nous avons calculées (informations redondantes) : - `Black_ELO_category` - `White_ELO_category` - `Avg_ELO_category`

[82]:
```
# Combien d'instances pour Game_type
df_spark_plus.groupBy("Game_type").count().show()
```

```
+--------------+-------+
|     Game_type|  count|
+--------------+-------+
|        Bullet| 806199|
|         Blitz|1812120|
|     Classical| 144677|
|         Rapid| 966569|
|Correspondence|  10344|
+--------------+-------+
```

[83]:
```
# Supprimer Game_type = Correspondance pour pouvoir garder starting_time et␣
↪increment
# (où les données peuvent être manquantes)
df_spark_plus = df_spark_plus.filter(col("Game_type") != "Correspondence")
```

[84]:
```
# Suppression des colonnes
df_preparation = df_spark_plus.drop("BlackRatingDiff", "WhiteRatingDiff",␣
↪"GAME", "Date", "Site", "TimeControl", "UTCTime", "Event", ␣
↪"Black_ELO_category", "White_ELO_category", "Avg_ELO_category")
```

[85]:
```
df_preparation.show(5)
```

```
+--------+---+--------------------+------+-----------+--------+-------------+--
-------+---------+----------+-------------+-------------+--------------+-----
---------+-----------------+-----------------+--------------------+----------
---------+-------------+-------------+-----------------+-----------------+----
------------+----------------+---------------+----------------+--------------
-----+--------------------+---------+-------------+----------+----------+----
----+
|BlackElo|ECO|             Opening|Result| Termination|WhiteElo|starting_time|in
crement|Game_type|Total_moves|Black_blunders|White_blunders|Black_mistakes|White
_mistakes|Black_inaccuracies|White_inaccuracies|Black_inferior_moves|White_infer
ior_moves|Black_ts_moves|White_ts_moves|Black_ts_blunders|White_ts_blunders|Blac
k_ts_mistakes|White_ts_mistake|Black_long_moves|White_long_moves|Black_bad_long_
moves|White_bad_long_moves|Game_flips|Game_flips_ts|white_moves|black_moves|ELO_
diff|
+--------+---+--------------------+------+-----------+--------+-------------+--
-------+---------+----------+-------------+-------------+--------------+-----
---------+-----------------+-----------------+--------------------+----------
---------+-------------+-------------+-----------------+-----------------+----
------------+----------------+---------------+----------------+--------------
-----+--------------------+---------+-------------+----------+----------+----
----+
|    1143|A02|        Bird Opening|   0-1|Time forfeit|    1180|          300|
0|    Blitz|         66|             4|              2|             0|
3|                3|                 1|                   7|
6|             8|             8|                0|                0|
0|               0|               2|               1|                   1|
1|         8|            0|       33.0|       33.0|      37|
|    1504|A04|        Réti Opening|   0-1|      Normal|    1381|          300|
0|    Blitz|         64|             2|              1|             1|
1|                7|                 5|                  10|
7|             0|             0|                0|                0|
0|               0|               0|               1|                   0|
0|         6|            0|       32.0|       32.0|    -123|
|    1933|C41|    Philidor Defense|   0-1|Time forfeit|    1485|          300|
2|    Blitz|         70|             0|              1|             1|
2|                8|                 8|                   9|
```

```
11|             0|             2|                0|                0|
0|              0|               1|               1|                1|
0|         5|            0|       35.0|       35.0|    -448|
|    1710|B23|Sicilian Defense:…|   0-1|      Normal|    2040|          180|
2|    Blitz|         86|             4|             2|             1|
5|                3|                4|                8|
11|           18|             0|                4|                0|
0|              0|               3|               1|                1|
0|         8|            1|       43.0|       43.0|     330|
|    1598|B03|    Alekhine Defense|   1-0|      Normal|    2163|          600|
0|    Rapid|         71|             1|             0|             1|
1|                6|                2|                8|
3|             0|             0|                0|                0|
0|              0|               0|               0|                0|
0|         2|            0|       36.0|       35.0|     565|
+--------+---+--------------------+------+------------+--------+-------------+--
-------+---------+-----------+-------------+-------------+-------------+-----
---------+-----------------+-----------------+--------------------+-----------
---------+-------------+-------------+-----------------+-----------------+----
------------+----------------+---------------+----------------+--------------
-----+--------------------+----------+-------------+-----------+-----------+----
----+
only showing top 5 rows
```

[86]:
```
# Combien de valeurs nulles par colonne ?
df_preparation.select([count(when(col(c).isNull(), c)).alias(c) for c in␣
  ↪df_preparation.columns]).show()
```

```
+--------+---+-------+------+-----------+--------+-------------+---------+------
---+-----------+-------------+-------------+-------------+-------------+----
-------------+------------------+--------------------+--------------------+----
---------+-------------+-----------------+-----------------+-----------------
+---------------+---------------+---------------+--------------------+-------
------------+----------+------------+-----------+-----------+--------+
|BlackElo|ECO|Opening|Result|Termination|WhiteElo|starting_time|increment|Game_t
ype|Total_moves|Black_blunders|White_blunders|Black_mistakes|White_mistakes|Blac
k_inaccuracies|White_inaccuracies|Black_inferior_moves|White_inferior_moves|Blac
k_ts_moves|White_ts_moves|Black_ts_blunders|White_ts_blunders|Black_ts_mistakes|
White_ts_mistake|Black_long_moves|White_long_moves|Black_bad_long_moves|White_ba
d_long_moves|Game_flips|Game_flips_ts|white_moves|black_moves|ELO_diff|
+--------+---+-------+------+-----------+--------+-------------+---------+------
---+-----------+-------------+-------------+-------------+-------------+----
-------------+------------------+--------------------+--------------------+----
---------+-------------+-----------------+-----------------+-----------------
+---------------+---------------+---------------+--------------------+-------
------------+----------+------------+-----------+-----------+--------+
|       0|  0|      0|     0|          0|       0|            0|        0|
```

```
0|          0|             0|             0|             0|             0|
0|                 0|                  0|                   0|             0|
0|                0|                 0|                0|              0|
0|               0|                   0|                  0|         0|
0|          0|          0|       0|
+--------+---+-------+------+----------+--------+------------+---------+------
---+----------+--------------+--------------+--------------+--------------+----
-------------+------------------+--------------------+--------------------+----
----------+--------------+-----------------+-----------------+-----------------
+----------------+----------------+----------------+--------------------+-------
------------+----------+------------+----------+----------+--------+
```

[87]:
```
# Pas de valeur null, c'est parfait
```

Nous réalisons l'analyse de corrélation, de covariance et des tableaux de contingence avant de finaliser la préparation des données, car la normalisation, la standardisation et l'encodage des données catégorielles peuvent influencer l'interprétation des relations entre les variables. En effet, ces techniques de prétraitement modifient l'échelle ou la représentation des données, ce qui peut fausser les résultats des analyses de corrélation ou de covariance si elles sont effectuées après ces transformations.

Pourquoi cela est important :

Corrélation et Covariance : - La corrélation mesure la force et la direction d'une relation linéaire entre deux variables. Elle peut être affectée par la mise à l'échelle des données. Si ces transformations sont faites après l'analyse de corrélation, il devient plus difficile d'interpréter les relations d'origine. - La covariance, bien que similaire à la corrélation, n'est pas dimensionnée. Elle peut être influencée par les unités de mesure des variables, ce qui peut fausser l'interprétation si les données ne sont pas préparées correctement. - **NB** : Nous avons tout de même besoin d'encoder Result en Result_index avant l'analyse de corrélation et de covariance.

Encodage des données catégorielles : - Lorsque nous avons des colonnes catégorielles, il est nécessaire de les encoder. L'encodage peut introduire des relations implicites ou artificielles entre les variables. Cela peut affecter la façon dont les relations entre les catégories sont perçues dans les analyses. Il est donc important d'encodé correctement ces variables avant d'exécuter l'analyse, pour éviter d'introduire de fausses relations ou de perdre des informations importantes.

[88]:
```
# Nombre de valeurs par Result
df_preparation.groupBy("Result").count().show()
```

```
+-------+-------+
| Result|  count|
+-------+-------+
|      *|     72|
|1/2-1/2| 108898|
|    1-0|1858840|
|    0-1|1761755|
+-------+-------+
```

Il y a seulement 72 valeurs indéfinies dans la colonnes Result, nous allons supprimer ces parties d'échec (ces lignes), car elles n'ont pas d'intérêts et sont en trop faible nombre pour apporter un réel résultat à notre analyse.

[89]:
```
df_preparation = df_preparation.filter(col("Result") != "*")
```

Table de contingence

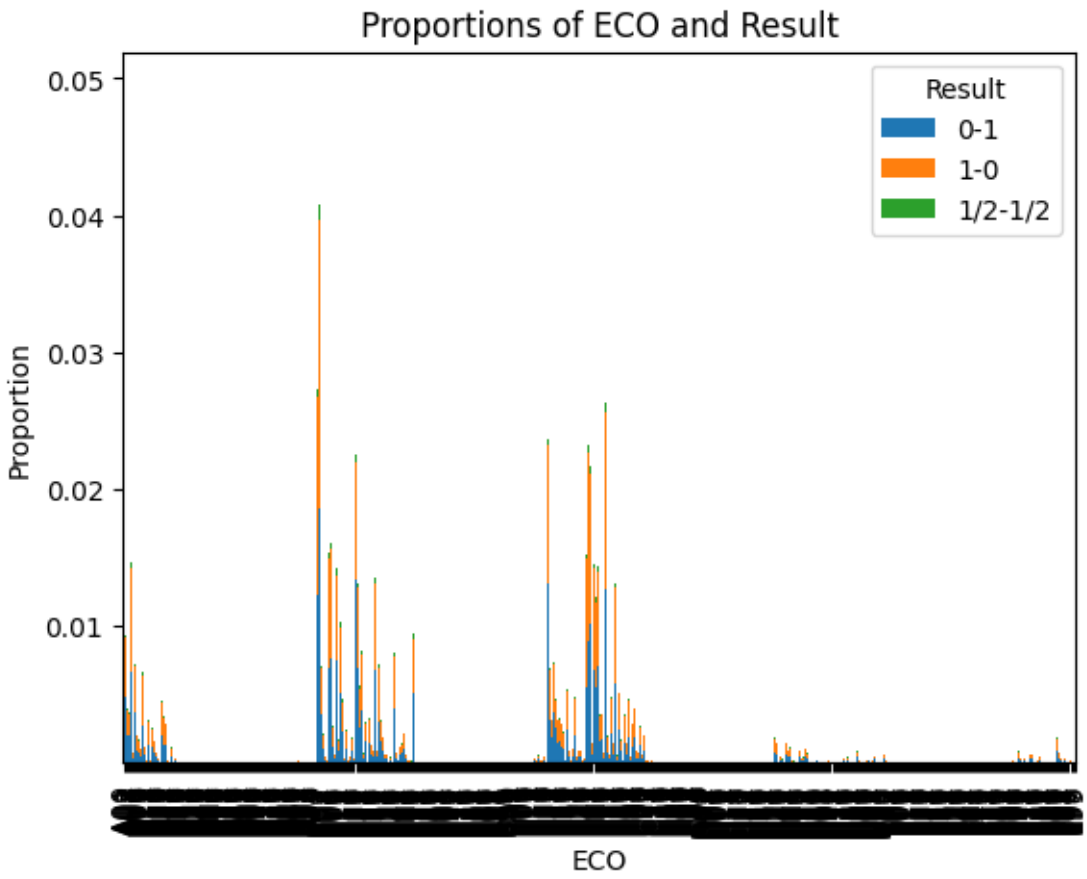
[90]:
```
# Sélectionner toutes les colonnes catégorielles dans le DataFrame
cat_columns = [col for col, dtype in df_preparation.dtypes if dtype == 'string']
cat_columns
```

[90]:
```
['ECO', 'Opening', 'Result', 'Termination', 'Game_type']
```

[91]:
```
# Fonction pour créer la table de contingence
def create_contingency_table(data, col1, col2):
    return data.groupBy(col1, col2).agg(F.count('*').alias('count'))

# Exclure 'Result' de cat_columns
cat_columns = [col for col in cat_columns if col != 'Result']

# Liste pour stocker les résultats
contingency_tables = []

# Boucle pour générer la table de contingence pour chaque paire de colonnes␣
 ↪catégorielles
for col in cat_columns:
    # Créer la table de contingence
    contingency_table = create_contingency_table(df_preparation, col, 'Result')
    contingency_table_ord = contingency_table.orderBy(col, 'Result')
    print(f"Contingency Table for {col} and Result :")
    contingency_table_ord.show()

    # Convertir la table de contingence en DataFrame Pandas pour l'analyse
    contingency_df = contingency_table.toPandas()

    # Calculer les proportions de chaque combinaison par rapport au total
    contingency_df['Proportion'] = contingency_df['count'] /␣
 ↪contingency_df['count'].sum()

    # Ajouter la table de contingence à la liste
    contingency_tables.append({
        "Variable": col,
        "Contingency Table": contingency_df
    })

    # S'assurer que chaque combinaison existe, remplir les valeurs manquantes␣
 ↪avec 0
```

```
    contingency_df = contingency_df.pivot_table(index=col, columns='Result',␣
 ↪values='Proportion', aggfunc='sum', fill_value=0)

    # Visualisation de la table de contingence
    plt.figure(figsize=(10, 6))
    contingency_df.plot(kind='bar', stacked=True)
    plt.title(f"Proportions of {col} and Result")
    plt.xlabel(col)
    plt.ylabel("Proportion")
    plt.legend(title="Result")
    plt.show()

# Résumé des tables de contingence
for result in contingency_tables:
    print(f"Summary for {result['Variable']}:")
    print(result['Contingency Table'].head())  # Affiche les premières lignes␣
 ↪pour une vue d'ensemble
    print("\n")
```

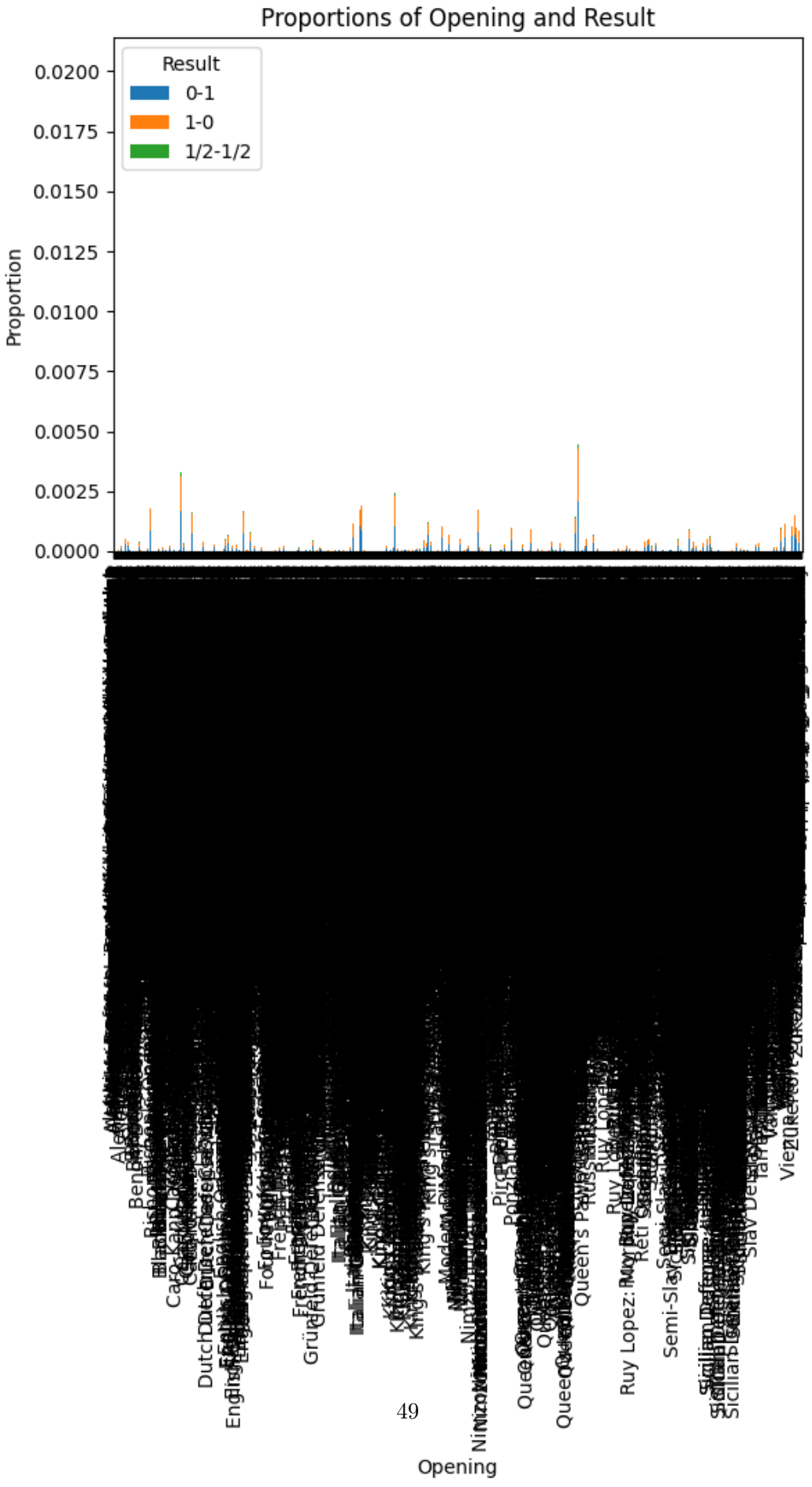
```
Contingency Table for ECO and Result :
+---+-------+-----+
|ECO| Result|count|
+---+-------+-----+
|A00|    0-1|99980|
|A00|    1-0|81207|
|A00|1/2-1/2| 3257|
|A01|    0-1|17722|
|A01|    1-0|16264|
|A01|1/2-1/2|  917|
|A02|    0-1| 7415|
|A02|    1-0| 6979|
|A02|1/2-1/2|  347|
|A03|    0-1| 7211|
|A03|    1-0| 6088|
|A03|1/2-1/2|  390|
|A04|    0-1|24630|
|A04|    1-0|28234|
|A04|1/2-1/2| 1534|
|A05|    0-1| 1282|
|A05|    1-0| 1389|
|A05|1/2-1/2|  177|
|A06|    0-1|13728|
|A06|    1-0|12632|
+---+-------+-----+
only showing top 20 rows


<Figure size 1000x600 with 0 Axes>
```



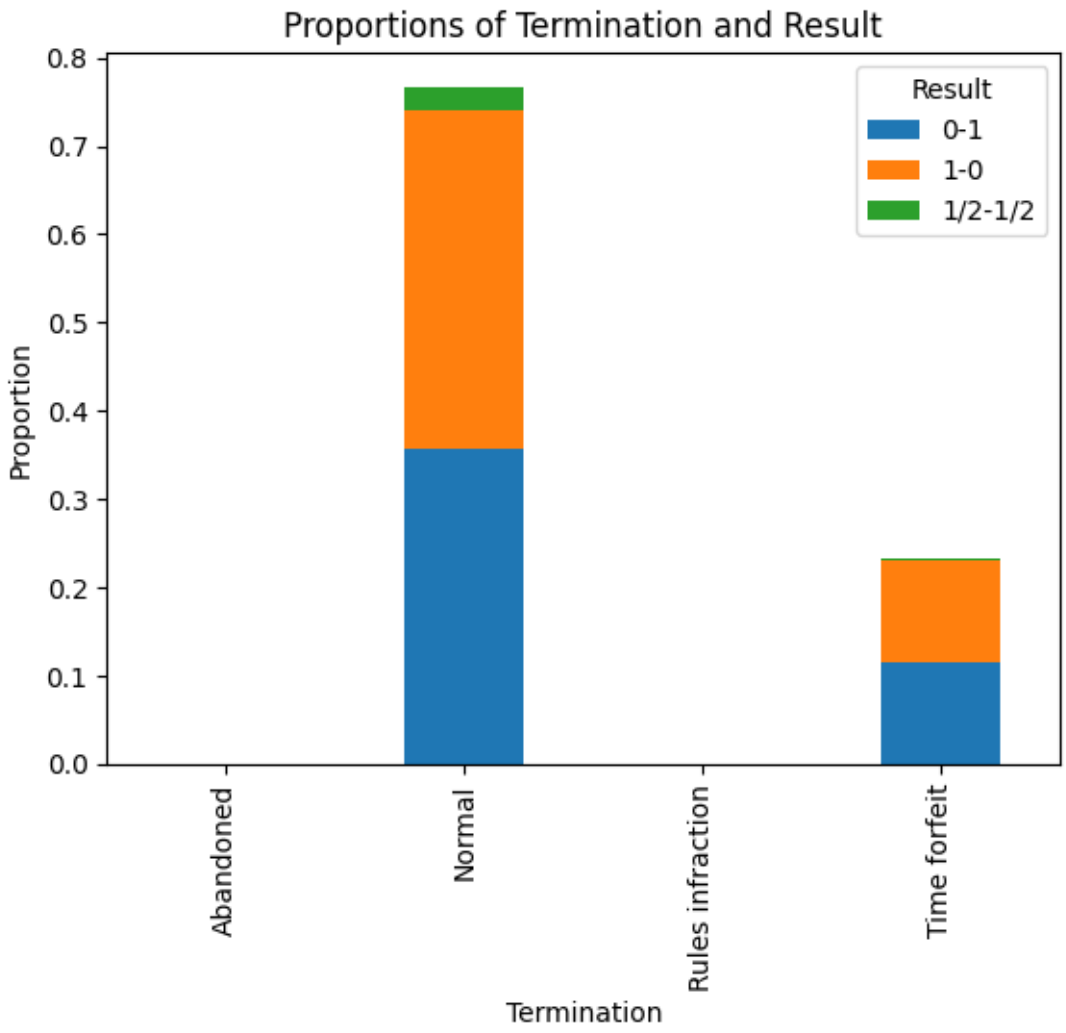
Contingency Table for Opening and Result :

Opening	Result	count
Alekhine Defense	0-1	4572
Alekhine Defense	1-0	4888
Alekhine Defense	1/2-1/2	197
Alekhine Defense:…	0-1	216
Alekhine Defense:…	1-0	195
Alekhine Defense:…	1/2-1/2	19
Alekhine Defense:…	0-1	349
Alekhine Defense:…	1-0	502
Alekhine Defense:…	1/2-1/2	11
Alekhine Defense:…	0-1	2
Alekhine Defense:…	1-0	7
Alekhine Defense:…	1-0	1
Alekhine Defense:…	0-1	1154
Alekhine Defense:…	1-0	1170
Alekhine Defense:…	1/2-1/2	102

```
|Alekhine Defense:…|    0-1|    1|
|Alekhine Defense:…|    1-0|    1|
|Alekhine Defense:…|    0-1|  257|
|Alekhine Defense:…|    1-0|  294|
|Alekhine Defense:…|1/2-1/2|   13|
+--------------------+-------+-----+
only showing top 20 rows

<Figure size 1000x600 with 0 Axes>
```


Proportions of Opening and Result
Result
0-1
1-0
1/2-1/2
Proportion
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
Opening

```
Contingency Table for Termination and Result :
+---------------+-------+-------+
|     Termination| Result|  count|
+---------------+-------+-------+
|       Abandoned|    1-0|      2|
|          Normal|    0-1|1331420|
|          Normal|    1-0|1426411|
|          Normal|1/2-1/2| 101932|
|Rules infraction|    0-1|    168|
|Rules infraction|    1-0|    159|
|    Time forfeit|    0-1| 430167|
|    Time forfeit|    1-0| 432268|
|    Time forfeit|1/2-1/2|   6966|
+---------------+-------+-------+


<Figure size 1000x600 with 0 Axes>
```

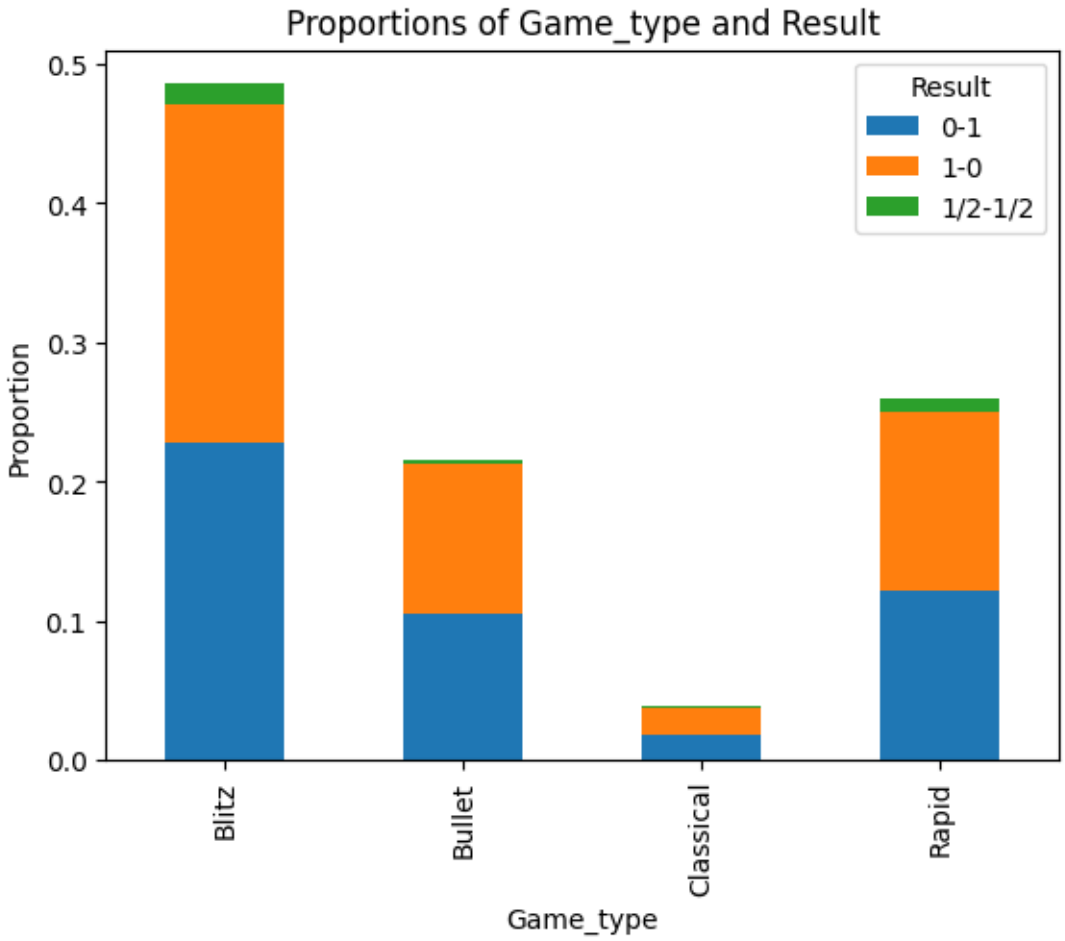


Contingency Table for Game_type and Result :

Game_type	Result	count
Blitz	0-1	850624
Blitz	1-0	903697
Blitz	1/2-1/2	57762
Bullet	0-1	389947
Bullet	1-0	404537
Bullet	1/2-1/2	11707
Classical	0-1	67652
Classical	1-0	71064
Classical	1/2-1/2	5956
Rapid	0-1	453532
Rapid	1-0	479542

```
|    Rapid|1/2-1/2| 33473|
+---------+-------+------+
```

```
<Figure size 1000x600 with 0 Axes>
```




```
Summary for ECO:
   ECO   Result  count  Proportion
0  C61      1-0   3052    0.000818
1  B07  1/2-1/2   1628    0.000437
2  E40      0-1    273    0.000073
3  B26      1-0    135    0.000036
4  C40  1/2-1/2    970    0.000260
```

```
Summary for Opening:
                                    Opening Result  count  Proportion
0      Sicilian Defense: Accelerated Dragon    1-0   1433    0.000384
1                           Budapest Defense    0-1    453    0.000121
```

```
2                           Blackmar Gambit    0-1   1980    0.000531
3         Italian Game: Two Knights Defense    1-0   3434    0.000921
4  King's Knight Opening: Konstantinopolsky    0-1   1080    0.000290


Summary for Termination:
        Termination   Result    count  Proportion
0      Time forfeit      1-0   432268    0.115905
1            Normal  1/2-1/2   101932    0.027331
2            Normal      1-0  1426411    0.382468
3      Time forfeit  1/2-1/2     6966    0.001868
4  Rules infraction      0-1      168    0.000045


Summary for Game_type:
   Game_type   Result   count  Proportion
0      Rapid      1-0  479542    0.128581
1      Blitz  1/2-1/2   57762    0.015488
2      Blitz      0-1  850624    0.228080
3  Classical      0-1   67652    0.018140
4     Bullet      1-0  404537    0.108470
```

Les analyses des tables de contingence pour les relations entre ECO et Result, Opening et Result, Termination et Result, ainsi que Game_type et Result ne révèlent pas de différences significatives en termes de proportions, avec une distribution globalement équilibrée entre les victoires des Blancs, des Noirs et un nombre notablement plus faible de matchs nuls.

[92]:
```python
# Sélectionner toutes les colonnes catégorielles dans le DataFrame qui ne sont␣
 ↪pas Result
cat_columns = [col for col, dtype in df_preparation.dtypes if dtype == 'string'␣
 ↪and col != 'Result']

# Indexation des colonnes catégorielles
indexers = [
    StringIndexer(inputCol=col_name, outputCol=col_name + "_index")
    for col_name in cat_columns + ['Result']
]

# Créer un VectorAssembler pour assembler les colonnes en un vecteur
assembler = VectorAssembler(
    inputCols=[col_name + "_index" for col_name in cat_columns],
    outputCol="features"
)

# Créer le pipeline
pipeline = Pipeline(stages=indexers + [assembler])
```

```python
# Appliquer le pipeline pour transformer les données
df_transformed = pipeline.fit(df_preparation).transform(df_preparation)

# Liste pour stocker les résultats
chi_results = []

# Boucle pour générer la table de contingence et effectuer le test chi-deux␣
 ↪pour chaque paire de colonnes catégorielles
for col in cat_columns:
    # Créer un DataFrame avec les colonnes à tester (en utilisant "features"␣
 ↪comme colonne d'entrée)
    df_chi = df_transformed.select("features", col + "_index")

    # Effectuer le test de chi-deux
    chi_result = ChiSquareTest.test(df_chi, "features", col + "_index").head()

    # Afficher le résultat du test de chi-deux
    print(f"Chi-Square Test for {col} and Result:")
    print(f"Chi-Square Statistic: {chi_result[0]}, p-value: {chi_result[1]}")

    # Ajouter les résultats dans la liste
    chi_results.append({
        "Variable": col,
        "Chi-Square Statistic": chi_result[0],
        "p-value": chi_result[1]
    })

# Créer un DataFrame pour les résultats
chi_results_df = spark.createDataFrame(chi_results)

# Afficher les résultats
chi_results_df.show()
```

```
Chi-Square Test for ECO and Result:
Chi-Square Statistic: [0.0,0.0,0.0,0.0], p-value: [241081, 1368908, 1473, 1473]
Chi-Square Test for Opening and Result:
Chi-Square Statistic: [0.0,0.0,0.0,0.0], p-value: [1368908, 7772944, 8364, 8364]
Chi-Square Test for Termination and Result:
Chi-Square Statistic: [0.0,0.0,0.0,0.0], p-value: [1473, 8364, 9, 9]
Chi-Square Test for Game_type and Result:
Chi-Square Statistic: [0.0,0.0,0.0,0.0], p-value: [1473, 8364, 9, 9]
+--------------------+-----------+--------------------+
|Chi-Square Statistic|   Variable|             p-value|
+--------------------+-----------+--------------------+
|   [0.0,0.0,0.0,0.0]|        ECO|[241081, 1368908,…|
|   [0.0,0.0,0.0,0.0]|    Opening|[1368908, 7772944…|
```

```
|   [0.0,0.0,0.0,0.0]|Termination|  [1473, 8364, 9, 9]|
|   [0.0,0.0,0.0,0.0]|  Game_type|  [1473, 8364, 9, 9]|
+--------------------+-----------+--------------------+
```

Les résultats des tests du chi-deux obtenus montrent les valeurs statistiques et les p-values pour les différentes variables catégorielles par rapport à la variable cible Result :

1. **ECO et Result**

Chi-Square Statistic : [0.0, 0.0, 0.0, 0.0]
p-value : [241081, 1368908, 1473, 1473]
L'absence de valeurs supérieures à zéro dans les statistiques du chi-deux (valeurs égales à 0) signifie qu'il n'y a pas d'association entre les catégories de la variable ECO et la variable cible Result. Cela suggère que les différentes catégories de ECO sont indépendantes de Result, du moins selon ce test.

Les p-values élevées (241081, 1368908, etc.) confirment l'absence de relation statistique significative entre les deux variables, car une p-value élevée (généralement > 0.05) indique une faible probabilité que l'association observée soit due au hasard.

2. **Opening et Result**

Chi-Square Statistic : [0.0, 0.0, 0.0, 0.0]
p-value : [1368908, 7772944, 8364, 8364]
Comme pour ECO, les statistiques du chi-deux de Opening sont égales à 0, ce qui indique qu'il n'y a aucune association significative entre cette variable et Result. Les p-values très élevées renforcent cette conclusion : la probabilité que l'absence d'association soit due au hasard est très faible.

3. **Termination et Result**

Chi-Square Statistic : [0.0, 0.0, 0.0, 0.0]
p-value : [1473, 8364, 9, 9]
Ici encore, la statistique du chi-deux est nulle, ce qui suggère qu'il n'y a pas d'association significative entre Termination et Result. Cependant, les p-values sont légèrement plus petites (9 et 8364), ce qui peut indiquer des zones où une association pourrait potentiellement être présente, mais elles restent suffisamment élevées pour indiquer qu'il n'y a pas de relation forte.

4. **Game_type et Result**

Chi-Square Statistic : [0.0, 0.0, 0.0, 0.0]
p-value : [1473, 8364, 9, 9]
Comme les résultats pour Termination, la statistique du chi-deux est de 0, ce qui ne montre aucune association significative entre Game_type et Result. Les p-values sont similaires à celles de Termination, ce qui confirme qu'il n'y a pas de lien important entre ces variables.

Conclusion générale :

Les résultats montrent que pour ces variables (ECO, Opening, Termination, Game_type), les tests de chi-deux ne révèlent aucune relation statistiquement significative avec la variable Result. En effet, les statistiques du chi-deux sont toutes nulles (0.0), ce qui suggère que la distribution des catégories de ces variables est indépendante de la variable cible Result. Les p-values élevées corroborent ce constat, car une p-value élevée indique que l'on ne peut pas rejeter l'hypothèse nulle (indépendance entre les variables).

Cela signifie que aucune des variables testées (ECO, Opening, Termination, Game_type) n'a de lien évident avec Result dans notre jeu de données, du moins selon le test statistique du chi-deux effectué.

On encode Result en donnée numérique.

[93]:
```
# Result : colonne cible, 3 valeurs possibles
from pyspark.sql.functions import when, col # très capricieux, laissé l'import

df_preparation = df_preparation.withColumn(
    "Result_index",
    when(col("Result") == "1-0", 0)
    .when(col("Result") == "0-1", 2)
    .when(col("Result") == "1/2-1/2", 1) # nul "moyenne" des deux autres␣
 ↪scénarios
)

# Vérification
df_preparation.select("Result", "Result_index").distinct().show()
```

```
+-------+------------+
| Result|Result_index|
+-------+------------+
|    0-1|           2|
|    1-0|           0|
|1/2-1/2|           1|
+-------+------------+
```

Corrélation La corrélation mesure la force et la direction de la relation linéaire entre deux variables.

- Une corrélation proche de +1 ou -1 indique une forte relation linéaire.
- Une corrélation proche de 0 indique peu ou pas de relation linéaire.

[94]:
```
# Quelles sont les colonnes numériques ?
numeric_cols = [col[0] for col in df_preparation.dtypes if col[1] in ["int",␣
 ↪"double"]]
print(f'Les colonnes numériques sont : {numeric_cols}')
```

```
Les colonnes numériques sont : ['BlackElo', 'WhiteElo', 'starting_time',
'increment', 'Total_moves', 'Black_blunders', 'White_blunders',
'Black_mistakes', 'White_mistakes', 'Black_inaccuracies', 'White_inaccuracies',
'Black_inferior_moves', 'White_inferior_moves', 'Black_ts_moves',
'White_ts_moves', 'Black_ts_blunders', 'White_ts_blunders', 'Black_ts_mistakes',
'White_ts_mistake', 'Black_long_moves', 'White_long_moves',
'Black_bad_long_moves', 'White_bad_long_moves', 'Game_flips', 'Game_flips_ts',
'white_moves', 'black_moves', 'ELO_diff', 'Result_index']
```

[95]:
```
# Assembler les colonnes numériques en un seul vecteur
assembler = VectorAssembler(inputCols=numeric_cols,␣
 ↪outputCol="numeric_features")
assembled_data = assembler.transform(df_preparation)

# Calculer la matrice de corrélation
correlation_matrix = Correlation.corr(assembled_data, "numeric_features").
 ↪head()[0]

# Convertir en DataFrame pour un affichage clair
correlation_array = np.array(correlation_matrix.toArray())
correlation_df = pd.DataFrame(correlation_array, columns=numeric_cols,␣
 ↪index=numeric_cols)
print(correlation_df)
```

```
                     BlackElo  WhiteElo  starting_time  increment  \
BlackElo             1.000000  0.910094       0.008177  -0.007045
WhiteElo             0.910094  1.000000       0.007926  -0.006531
starting_time        0.008177  0.007926       1.000000   0.437907
increment           -0.007045 -0.006531       0.437907   1.000000
Total_moves          0.190490  0.187216       0.031237   0.019891
Black_blunders      -0.194336 -0.175368      -0.034803  -0.033971
White_blunders      -0.179486 -0.197494      -0.035166  -0.033524
Black_mistakes      -0.015877  0.000017      -0.003054  -0.010270
White_mistakes       0.005894 -0.009455      -0.005331  -0.010534
Black_inaccuracies   0.070694  0.084842      -0.013919  -0.011425
White_inaccuracies   0.079578  0.059822      -0.009121  -0.008592
Black_inferior_moves -0.063166 -0.039434      -0.023877  -0.026209
White_inferior_moves -0.040511 -0.065927      -0.022844  -0.024637
Black_ts_moves       0.081393  0.086050      -0.071706  -0.010079
White_ts_moves       0.086832  0.080402      -0.073448  -0.011350
Black_ts_blunders    0.023101  0.028246      -0.070892  -0.013347
White_ts_blunders    0.027351  0.021430      -0.071984  -0.014609
Black_ts_mistakes    0.057611  0.061778      -0.047617   0.006496
White_ts_mistake     0.063184  0.057863      -0.048793   0.005337
Black_long_moves     0.048672  0.071725      -0.150171   0.065143
White_long_moves     0.067875  0.044156      -0.148309   0.067522
Black_bad_long_moves 0.014075  0.031548      -0.115505   0.037204
White_bad_long_moves 0.028254  0.010393      -0.113197   0.039445
Game_flips          -0.065991 -0.066276      -0.028105  -0.025942
Game_flips_ts        0.046089  0.045568      -0.071213  -0.007951
white_moves          0.190072  0.187992       0.031236   0.019881
black_moves          0.190840  0.186376       0.031228   0.019894
ELO_diff            -0.211949  0.212093      -0.000592   0.001212
Result_index         0.029181 -0.041489      -0.001041  -0.000142

                     Total_moves  Black_blunders  White_blunders  \
```

BlackElo	0.190490	-0.194336	-0.179486
WhiteElo	0.187216	-0.175368	-0.197494
starting_time	0.031237	-0.034803	-0.035166
increment	0.019891	-0.033971	-0.033524
Total_moves	1.000000	0.274766	0.288349
Black_blunders	0.274766	1.000000	0.735059
White_blunders	0.288349	0.735059	1.000000
Black_mistakes	0.391566	0.289265	0.298925
White_mistakes	0.404630	0.286528	0.292921
Black_inaccuracies	0.410706	0.102398	0.038222
White_inaccuracies	0.418230	0.033583	0.112278
Black_inferior_moves	0.526847	0.654935	0.510427
White_inferior_moves	0.538865	0.498097	0.656823
Black_ts_moves	0.476070	0.185094	0.178056
White_ts_moves	0.475140	0.173649	0.191203
Black_ts_blunders	0.247385	0.352122	0.287828
White_ts_blunders	0.247062	0.287561	0.358143
Black_ts_mistakes	0.227842	0.125080	0.118882
White_ts_mistake	0.227539	0.114293	0.128213
Black_long_moves	0.088465	0.069776	0.029657
White_long_moves	0.087621	0.029140	0.072889
Black_bad_long_moves	0.042163	0.130267	0.072034
White_bad_long_moves	0.047552	0.072779	0.133722
Game_flips	0.370487	0.531165	0.542604
Game_flips_ts	0.291641	0.286551	0.290900
white_moves	0.999825	0.278331	0.284596
black_moves	0.999825	0.271107	0.291997
ELO_diff	-0.007705	0.044717	-0.042483
Result_index	0.027271	-0.216912	0.237662

	Black_mistakes	White_mistakes	Black_inaccuracies	…	\
BlackElo	-0.015877	0.005894	0.070694	…	
WhiteElo	0.000017	-0.009455	0.084842	…	
starting_time	-0.003054	-0.005331	-0.013919	…	
increment	-0.010270	-0.010534	-0.011425	…	
Total_moves	0.391566	0.404630	0.410706	…	
Black_blunders	0.289265	0.286528	0.102398	…	
White_blunders	0.298925	0.292921	0.038222	…	
Black_mistakes	1.000000	0.611193	0.207226	…	
White_mistakes	0.611193	1.000000	0.257391	…	
Black_inaccuracies	0.207226	0.257391	1.000000	…	
White_inaccuracies	0.246896	0.228736	0.472489	…	
Black_inferior_moves	0.764966	0.578336	0.626031	…	
White_inferior_moves	0.574370	0.770570	0.373284	…	
Black_ts_moves	0.210446	0.209167	0.188991	…	
White_ts_moves	0.206612	0.217828	0.192859	…	
Black_ts_blunders	0.193081	0.195548	0.109663	…	
White_ts_blunders	0.195087	0.197563	0.103111	…	

Black_ts_mistakes	0.286422	0.214583	0.117544	…
White_ts_mistake	0.212573	0.291011	0.119315	…
Black_long_moves	0.073363	0.049057	0.049675	…
White_long_moves	0.049856	0.078121	0.032980	…
Black_bad_long_moves	0.147201	0.090132	0.086487	…
White_bad_long_moves	0.094445	0.154390	0.040730	…
Game_flips	0.571288	0.581062	0.338290	…
Game_flips_ts	0.251762	0.255255	0.160356	…
white_moves	0.393357	0.402564	0.412662	…
black_moves	0.389641	0.406553	0.408609	…
ELO_diff	0.037480	-0.036199	0.033372	…
Result_index	-0.104876	0.135857	-0.111273	…

	Black_long_moves	White_long_moves	\
BlackElo	0.048672	0.067875	
WhiteElo	0.071725	0.044156	
starting_time	-0.150171	-0.148309	
increment	0.065143	0.067522	
Total_moves	0.088465	0.087621	
Black_blunders	0.069776	0.029140	
White_blunders	0.029657	0.072889	
Black_mistakes	0.073363	0.049856	
White_mistakes	0.049057	0.078121	
Black_inaccuracies	0.049675	0.032980	
White_inaccuracies	0.031588	0.048082	
Black_inferior_moves	0.094194	0.055395	
White_inferior_moves	0.054096	0.096629	
Black_ts_moves	0.307970	0.108168	
White_ts_moves	0.108816	0.309173	
Black_ts_blunders	0.250722	0.087567	
White_ts_blunders	0.086812	0.251344	
Black_ts_mistakes	0.240817	0.090345	
White_ts_mistake	0.090689	0.244087	
Black_long_moves	1.000000	0.330405	
White_long_moves	0.330405	1.000000	
Black_bad_long_moves	0.717295	0.225367	
White_bad_long_moves	0.224468	0.718602	
Game_flips	0.066110	0.070436	
Game_flips_ts	0.188967	0.191944	
white_moves	0.090575	0.085495	
black_moves	0.086325	0.089713	
ELO_diff	0.054368	-0.055933	
Result_index	-0.120649	0.126939	

	Black_bad_long_moves	White_bad_long_moves	Game_flips	\
BlackElo	0.014075	0.028254	-0.065991	
WhiteElo	0.031548	0.010393	-0.066276	
starting_time	-0.115505	-0.113197	-0.028105	

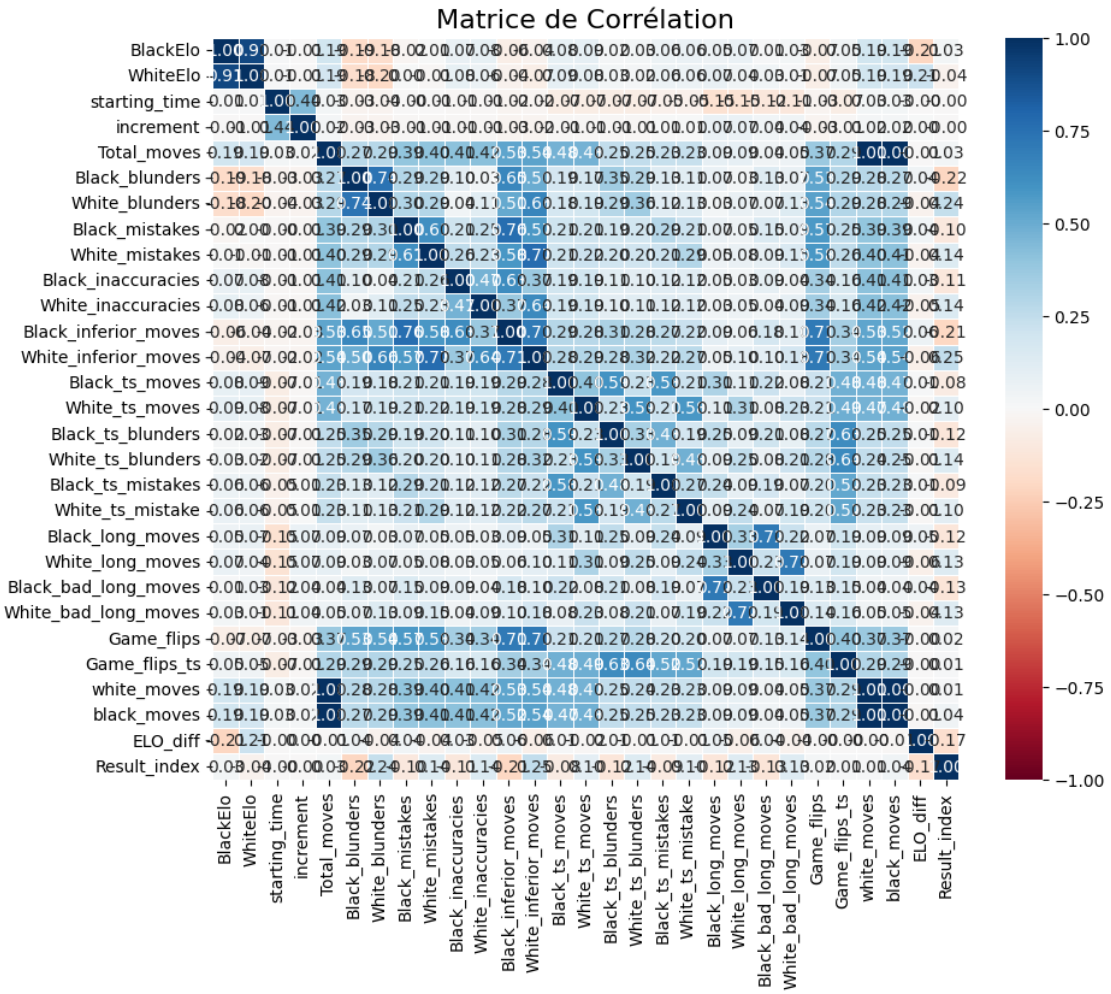
increment	0.037204	0.039445	-0.025942
Total_moves	0.042163	0.047552	0.370487
Black_blunders	0.130267	0.072779	0.531165
White_blunders	0.072034	0.133722	0.542604
Black_mistakes	0.147201	0.094445	0.571288
White_mistakes	0.090132	0.154390	0.581062
Black_inaccuracies	0.086487	0.040730	0.338290
White_inaccuracies	0.037340	0.089067	0.344633
Black_inferior_moves	0.178425	0.102686	0.705260
White_inferior_moves	0.097672	0.183573	0.712905
Black_ts_moves	0.218540	0.077923	0.212047
White_ts_moves	0.075222	0.225314	0.214930
Black_ts_blunders	0.206523	0.080438	0.271425
White_ts_blunders	0.077716	0.210979	0.275155
Black_ts_mistakes	0.185874	0.073020	0.199709
White_ts_mistake	0.070792	0.193023	0.200311
Black_long_moves	0.717295	0.224468	0.066110
White_long_moves	0.225367	0.718602	0.070436
Black_bad_long_moves	1.000000	0.191109	0.132226
White_bad_long_moves	0.191109	1.000000	0.140760
Game_flips	0.132226	0.140760	1.000000
Game_flips_ts	0.154021	0.159943	0.396613
white_moves	0.044320	0.045360	0.370238
black_moves	0.039993	0.049726	0.370606
ELO_diff	0.041208	-0.042119	-0.000676
Result_index	-0.126236	0.131903	0.021913

	Game_flips_ts	white_moves	black_moves	ELO_diff	\
BlackElo	0.046089	0.190072	0.190840	-0.211949	
WhiteElo	0.045568	0.187992	0.186376	0.212093	
starting_time	-0.071213	0.031236	0.031228	-0.000592	
increment	-0.007951	0.019881	0.019894	0.001212	
Total_moves	0.291641	0.999825	0.999825	-0.007705	
Black_blunders	0.286551	0.278331	0.271107	0.044717	
White_blunders	0.290900	0.284596	0.291997	-0.042483	
Black_mistakes	0.251762	0.393357	0.389641	0.037480	
White_mistakes	0.255255	0.402564	0.406553	-0.036199	
Black_inaccuracies	0.160356	0.412662	0.408609	0.033372	
White_inaccuracies	0.162740	0.415985	0.420327	-0.046584	
Black_inferior_moves	0.339777	0.530335	0.523177	0.055962	
White_inferior_moves	0.342000	0.535040	0.542499	-0.059941	
Black_ts_moves	0.483665	0.477577	0.474397	0.010988	
White_ts_moves	0.488007	0.473547	0.476566	-0.015157	
Black_ts_blunders	0.629218	0.249531	0.245155	0.012135	
White_ts_blunders	0.636299	0.244794	0.249242	-0.013962	
Black_ts_mistakes	0.517179	0.229456	0.226148	0.009831	
White_ts_mistake	0.519071	0.225806	0.229191	-0.012545	
Black_long_moves	0.188967	0.090575	0.086325	0.054368	

```
White_long_moves           0.191944     0.085495     0.089713 -0.055933
Black_bad_long_moves       0.154021     0.044320     0.039993  0.041208
White_bad_long_moves       0.159943     0.045360     0.049726 -0.042119
Game_flips                 0.396613     0.370238     0.370606 -0.000676
Game_flips_ts              1.000000     0.291562     0.291618 -0.001226
white_moves                0.291562     1.000000     0.999300 -0.004892
black_moves                0.291618     0.999300     1.000000 -0.010513
ELO_diff                  -0.001226    -0.004892    -0.010513  1.000000
Result_index               0.009755     0.010808     0.043709 -0.166659

                      Result_index
BlackElo                  0.029181
WhiteElo                 -0.041489
starting_time            -0.001041
increment                -0.000142
Total_moves               0.027271
Black_blunders           -0.216912
White_blunders            0.237662
Black_mistakes           -0.104876
White_mistakes            0.135857
Black_inaccuracies       -0.111273
White_inaccuracies        0.143083
Black_inferior_moves     -0.206421
White_inferior_moves      0.245347
Black_ts_moves           -0.078596
White_ts_moves            0.099583
Black_ts_blunders        -0.119808
White_ts_blunders         0.135608
Black_ts_mistakes        -0.089050
White_ts_mistake          0.103767
Black_long_moves         -0.120649
White_long_moves          0.126939
Black_bad_long_moves     -0.126236
White_bad_long_moves      0.131903
Game_flips                0.021913
Game_flips_ts             0.009755
white_moves               0.010808
black_moves               0.043709
ELO_diff                 -0.166659
Result_index              1.000000

[29 rows x 29 columns]
```

[96]:
```
# Créer une heatmap avec seaborn pour une meilleure visualisation
plt.figure(figsize=(10, 8))

# Utiliser une palette de couleurs allant de -1 à 1 (avec 0 en blanc)
```

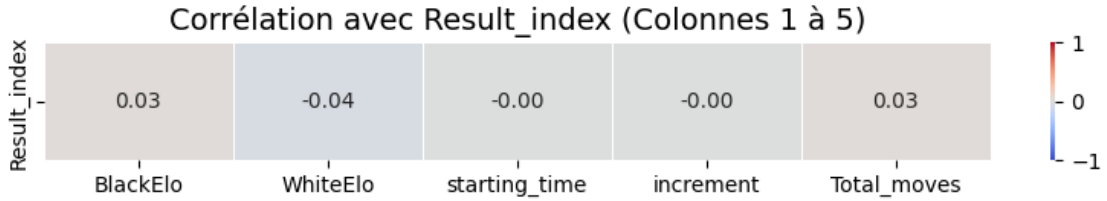
```
sns.heatmap(correlation_df, annot=True, cmap='RdBu', center=0, vmin=-1, vmax=1,␣
 ↪fmt=".2f", linewidths=0.5)

# Ajouter un titre
plt.title("Matrice de Corrélation", fontsize=16)

# Afficher la heatmap
plt.show()
```



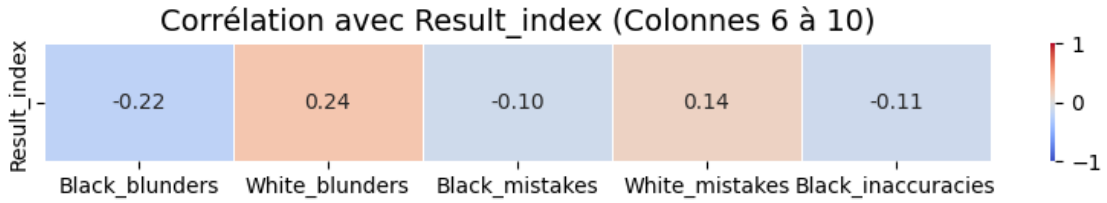
[97]:
```
# Sélectionner uniquement la dernière ligne (ici 'Result_index')
result_corr = correlation_df.loc['Result_index']

# Diviser en groupes de 5 colonnes
columns = correlation_df.columns
step = 5
```

```python
for i in range(0, len(columns), step):
    # Sélectionner un sous-ensemble de colonnes
    subset_columns = columns[i:i + step]
    result_corr_subset = result_corr[subset_columns].to_frame().T

    # Afficher les colonnes sélectionnées
    print(result_corr_subset)

    # Créer une heatmap pour le sous-ensemble
    plt.figure(figsize=(10, 1))  # Ajuster la taille pour le sous-ensemble
    sns.heatmap(result_corr_subset, annot=True, cmap='coolwarm', center=0,␣
 ↪vmin=-1, vmax=1, fmt=".2f", linewidths=0.5)
    plt.title(f"Corrélation avec Result_index (Colonnes {i+1} à␣
 ↪{i+len(subset_columns)})", fontsize=14)
    plt.show()
```

```
              BlackElo  WhiteElo  starting_time  increment  Total_moves
Result_index  0.029181 -0.041489      -0.001041  -0.000142     0.027271
```

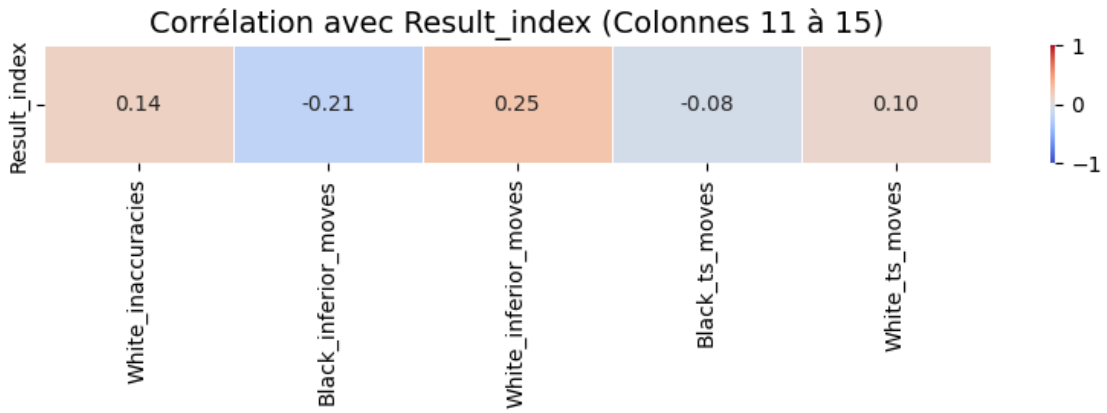



```
              Black_blunders  White_blunders  Black_mistakes  White_mistakes  \
Result_index       -0.216912        0.237662       -0.104876        0.135857

              Black_inaccuracies
Result_index           -0.111273
```

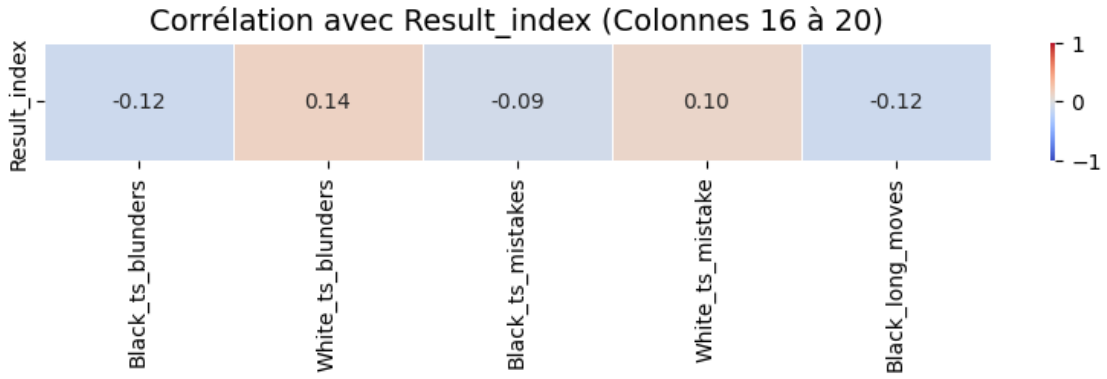



```
              White_inaccuracies  Black_inferior_moves  White_inferior_moves  \
Result_index            0.143083             -0.206421              0.245347

              Black_ts_moves  White_ts_moves
```

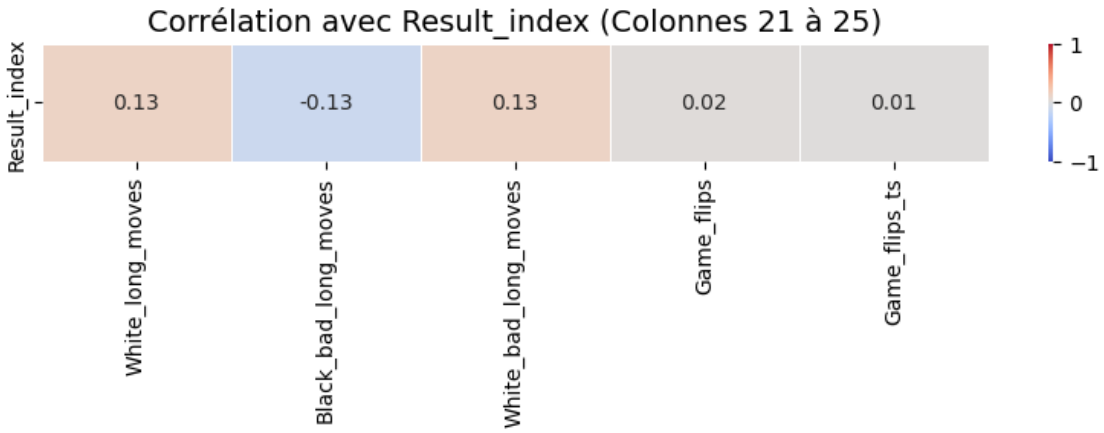
```
Result_index       -0.078596         0.099583
```



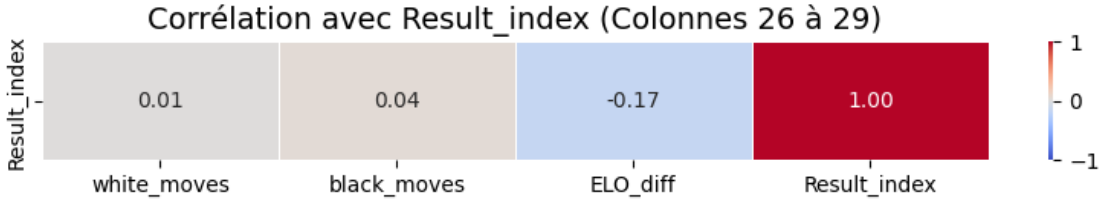

```
              Black_ts_blunders  White_ts_blunders  Black_ts_mistakes  \
Result_index          -0.119808           0.135608           -0.08905

              White_ts_mistake  Black_long_moves
Result_index          0.103767         -0.120649
```




```
              White_long_moves  Black_bad_long_moves  White_bad_long_moves  \
Result_index          0.126939             -0.126236              0.131903

              Game_flips  Game_flips_ts
Result_index    0.021913       0.009755
```



	white_moves	black_moves	ELO_diff	Result_index
Result_index	0.010808	0.043709	-0.166659	1.0



Variables liées à l'ELO

- `BlackElo` (0.029) : Une faible corrélation positive indique que lorsque l'ELO du joueur noir augmente, il y a une légère tendance à une victoire pour Black (classe 2), mais cet effet est négligeable. (Peut-être pas une vistoire de noir, mais on tend alors vers 1 ou 2, match nul ou victoire de noir.)
- `WhiteElo` (-0.041) : Une faible corrélation négative suggère qu'un ELO élevé pour White favorise légèrement la victoire de White (classe 0), mais l'effet reste marginal.
- `ELO_diff` (-0.167) : Une corrélation négative modérée montre que lorsque la différence d'ELO augmente (en faveur de White), la probabilité de victoire pour White (classe 0) augmente, ce qui est intuitif.

Variables temporelles

- `starting_time` (-0.001) et `increment` (-0.000) : Les corrélations quasi nulles montrent que ni le temps de départ ni l'incrément n'ont d'influence significative sur le résultat de la partie.

Variables sur le nombre de coups

- `Total_moves` (0.027) : Une faible corrélation positive indique que les parties avec plus de coups sont légèrement associées à des résultats favorisant les matchs nuls (classe 1) ou les

victoires de noir (classe 2), mais l'effet est très faible.

Blunders

- `Black_blunders` (-0.217) : Une corrélation négative modérée montre que plus le joueur noir commet des blunders, moins il est probable qu'il gagne (classe 2), ce qui est attendu.
- `White_blunders` (0.238) : Une corrélation positive modérée montre que plus le joueur blanc fait de blunders, plus il est probable que White perde, ce qui favorise les victoires de Black (classe 2).

Mistakes et inaccuracies

- `Black_mistakes` (-0.105) et `Black_inaccuracies` (-0.111) : Des corrélations négatives faibles montrent que les erreurs mineures des noirs réduisent leurs chances de victoire, mais pas de manière aussi significative que les blunders.
- `White_mistakes` (0.136) et `White_inaccuracies` (0.143) : Des corrélations positives faibles montrent que les erreurs des blancs augmentent la probabilité de victoire pour Black.

Inferior moves (mauvais coups globaux)

- `Black_inferior_moves` (-0.206) : Une corrélation négative modérée montre que des mauvais coups fréquents chez les noirs réduisent leurs chances de victoire (classe 2).
- `White_inferior_moves` (0.245) : Une corrélation positive modérée montre que des mauvais coups fréquents chez les blancs augmentent les chances pour Black de gagner (classe 2).

Time-sensitive moves (erreurs sous pression de temps)

- `Black_ts_blunders` (-0.120) et `Black_ts_mistakes` (-0.089) : Ces corrélations montrent que les erreurs des noirs sous pression temporelle réduisent leurs chances de victoire, mais l'effet est modéré.
- `White_ts_blunders` (0.136) et `White_ts_mistakes` (0.104) : Ces corrélations positives montrent que les erreurs des blancs sous pression temporelle augmentent les chances pour Black de gagner (classe 2).

Long moves et bad long moves

- `Black_long_moves` (-0.121) et `Black_bad_long_moves` (-0.126) : Ces corrélations négatives faibles indiquent que des mauvais coups longs chez les noirs réduisent légèrement leurs chances de victoire.
- `White_long_moves` (0.127) et `White_bad_long_moves` (0.132) : Ces corrélations positives faibles montrent que les mauvais coups longs chez les blancs augmentent légèrement les chances pour Black de gagner.

Game flips (changement de dynamique)

- `Game_flips` (0.022) et `Game_flips_ts` (0.010) : Ces corrélations très faibles montrent que les retournements de situation dans la partie n'ont presque aucun impact sur le résultat final.

Synthèse des principaux facteurs :

- Différence d'ELO (`ELO_diff`) a une influence notable, avec une tendance claire : un ELO plus élevé favorise la victoire du joueur plus fort.
- Blunders (`Black_blunders` et `White_blunders`) ont l'effet le plus significatif sur le résultat. Les blunders des noirs diminuent leurs chances, tandis que ceux des blancs augmentent les chances de victoire pour Black.
- Inferior moves (Black et White) suivent une tendance similaire aux blunders, bien que leur impact soit légèrement plus faible.
- Mistakes et inaccuracies ont un impact moindre, mais leur tendance est cohérente avec celle des blunders.
- Facteurs temporels et coups longs ont peu d'effet significatif sur le résultat.

Conclusion Les résultats confirment que la performance est fortement liée aux erreurs majeures et à l'ELO des joueurs, tandis que les autres variables comme le temps ou les retournements de jeu ont une influence marginale.

Covariance

- Une covariance positive indique que les deux variables augmentent ensemble.
- Une covariance négative indique qu'une variable augmente tandis que l'autre diminue.

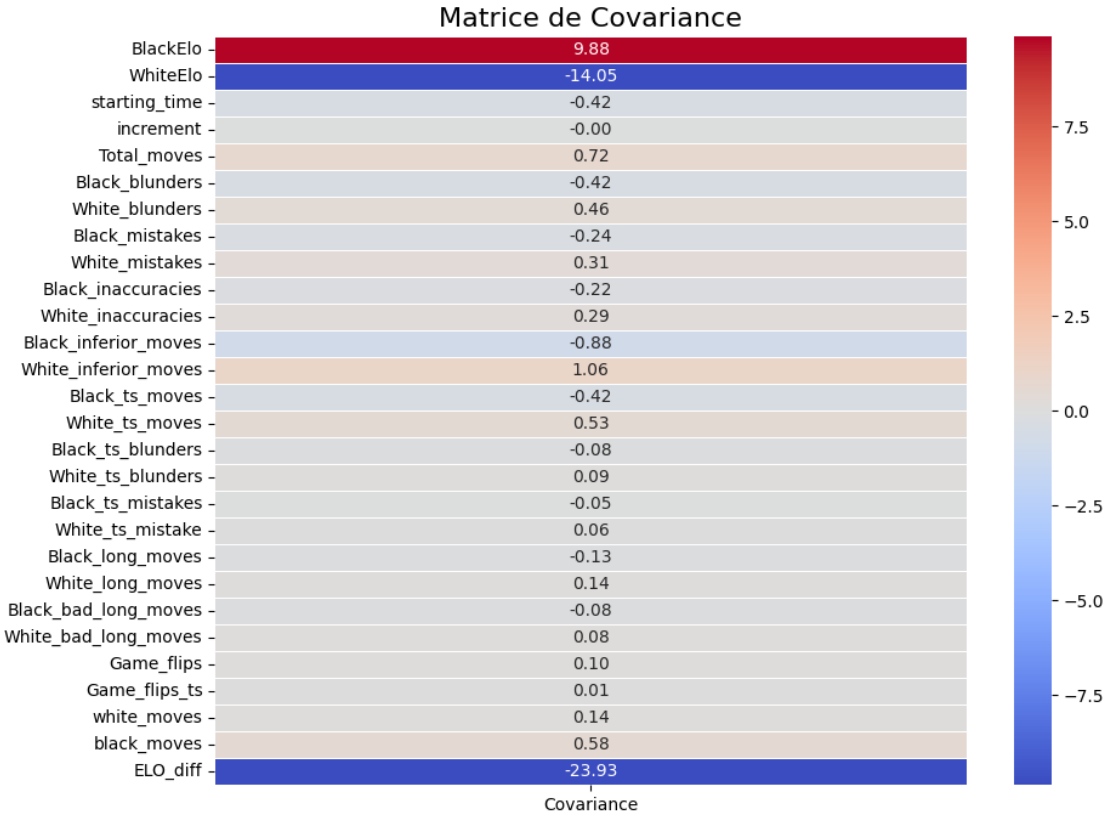
[98]:
```
# Liste des colonnes numériques sauf "Result"
numeric_cols_without_result = [col for col in numeric_cols if col !=␣
 ↪'Result_index']

data = df_preparation

# Calculer la covariance entre 'Result' et chaque autre colonne numérique
cov_result = []
for col in numeric_cols_without_result:
    print(f"Calcul de {col}")
    covariance = data.stat.cov('Result_index', col)
    cov_result.append(covariance)

# Créer un DataFrame pour afficher les résultats
cov_result_df = pd.DataFrame(cov_result, columns=['Covariance'],␣
 ↪index=numeric_cols_without_result)

# Afficher les covariances
print(cov_result_df)
```

```
Calcul de BlackElo
Calcul de WhiteElo
Calcul de starting_time
Calcul de increment
Calcul de Total_moves
Calcul de Black_blunders
Calcul de White_blunders
```

```
Calcul de Black_mistakes
Calcul de White_mistakes
Calcul de Black_inaccuracies
Calcul de White_inaccuracies
Calcul de Black_inferior_moves
Calcul de White_inferior_moves
Calcul de Black_ts_moves
Calcul de White_ts_moves
Calcul de Black_ts_blunders
Calcul de White_ts_blunders
Calcul de Black_ts_mistakes
Calcul de White_ts_mistake
Calcul de Black_long_moves
Calcul de White_long_moves
Calcul de Black_bad_long_moves
Calcul de White_bad_long_moves
Calcul de Game_flips
Calcul de Game_flips_ts
Calcul de white_moves
Calcul de black_moves
Calcul de ELO_diff
                      Covariance
BlackElo                9.880810
WhiteElo              -14.049130
starting_time          -0.416678
increment              -0.000598
Total_moves             0.717455
Black_blunders         -0.419101
White_blunders          0.462271
Black_mistakes         -0.241101
White_mistakes          0.314062
Black_inaccuracies     -0.223042
White_inaccuracies      0.287405
Black_inferior_moves   -0.883244
White_inferior_moves    1.063739
Black_ts_moves         -0.416011
White_ts_moves          0.530404
Black_ts_blunders      -0.078282
White_ts_blunders       0.089726
Black_ts_mistakes      -0.049384
White_ts_mistake        0.058309
Black_long_moves       -0.131760
White_long_moves        0.138799
Black_bad_long_moves   -0.079347
White_bad_long_moves    0.083056
Game_flips              0.102300
Game_flips_ts           0.010070
white_moves             0.142137
```

```
black_moves              0.575318
ELO_diff               -23.929939
```

[99]:
```python
# Créer une heatmap pour la matrice de covariance
plt.figure(figsize=(10, 8))

# Utiliser une palette de couleurs pour les covariances
sns.heatmap(cov_result_df, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.
 ↪5, center=0, vmin=-cov_result_df.max().max(), vmax=cov_result_df.max().max())

# Ajouter un titre
plt.title("Matrice de Covariance", fontsize=16)

# Afficher la heatmap
plt.show()
```



Covariances importantes avec Result_index :

- `BlackElo` (9.88) : Une covariance positive suggère que l'ELO plus élevé du joueur noir est associé à une plus grande probabilité de victoire pour Black (classe 2). Cela est logique, car un joueur avec un ELO plus élevé est plus susceptible de gagner, mais l'effet n'est pas très fort.

- `WhiteElo` (-14.05) : La covariance négative indique que l'ELO plus élevé du joueur blanc est associé à une probabilité plus faible de match nul ou de victoire pour Black. Cela montre une relation inverse : un joueur blanc plus fort (plus élevé en ELO) augmente la probabilité de victoire pour White (classe 0).
- `Total_moves` (0.72) : La covariance positive montre que plus le nombre de coups est élevé, plus la probabilité de match nul (classe 1) augmente. Cela pourrait indiquer que les matchs avec plus de coups sont souvent plus équilibrés et ont une probabilité plus élevée de se terminer par un match nul. (Ceci dit, la probabilité de victoire de Black augmente par la même occasion.)
- `Black_blunders` (-0.42) et `White_blunders` (0.46) : Les blunders des noirs et des blancs ont une covariance relativement forte avec le résultat, suggérant que des erreurs (blunders) des noirs diminuent la probabilité de victoire pour Black (classe 2), tandis que les blunders des blancs augmentent la probabilité de victoire pour White (classe 0). Les blunders sont donc un facteur significatif dans le résultat de la partie.
- `Black_inferior_moves` (-0.88) et `White_inferior_moves` (1.06) : Les mauvais coups (moves inférieurs) des noirs sont associés négativement avec le résultat pour Black, tandis que les mauvais coups des blancs augmentent la probabilité de victoire de White. Cela montre que les erreurs stratégiques jouent un rôle important dans l'issue du match.
- `ELO_diff` (-23.93) : Une forte covariance négative indique que la différence d'ELO a un impact important sur le résultat. Lorsque la différence d'ELO entre les joueurs est importante, cela favorise la victoire de White (classe 0) ou de Black (classe 2), selon qui a le plus grand ELO. Cela est cohérent avec l'idée que des différences d'ELO plus élevées augmentent la probabilité que le joueur avec l'ELO le plus élevé gagne.

Covariances faibles ou peu significatives :

- `starting_time` (-0.42) et `increment` (-0.0006) : Ces valeurs faibles indiquent que le temps de départ et l'incrément n'ont qu'un faible impact sur le résultat du match. Cela soutient l'idée que les caractéristiques temporelles, comme le temps de départ ou les incréments, influencent peu le résultat final.
- `Black_mistakes` (-0.24) et `White_mistakes` (0.31) : Les erreurs sont légèrement corrélées avec le résultat, mais l'impact est modéré. Les erreurs des noirs réduisent légèrement la probabilité de victoire de Black, tandis que les erreurs des blancs ont un effet inverse.
- `Black_ts_moves` (-0.42) et `White_ts_moves` (0.53) : Les mouvements temporels (moves dans la time control) ont une corrélation faible, mais légèrement plus marquée du côté des blancs, où des mouvements dans le cadre du contrôle du temps pourraient favoriser leur victoire.
- `Black_long_moves` (-0.13) et `White_long_moves` (0.14) : Les mouvements longs n'ont qu'une faible covariance avec le résultat, bien que les mouvements longs des blancs semblent légèrement augmenter la probabilité de leur victoire.

Autres variables et facteurs supplémentaires :

- `Black_bad_long_moves` (-0.08) et `White_bad_long_moves` (0.08) : Bien que faibles, ces covariances suggèrent que les mouvements longs "mauvais" des noirs sont légèrement corrélés avec une diminution de leur probabilité de victoire, tandis que ceux des blancs semblent avoir l'effet inverse, mais de manière marginale.
- `Game_flips` (0.10) et `Game_flips_ts` (0.01) : Ces valeurs indiquent que les flips du jeu n'ont qu'un très faible impact sur le résultat du match.

Conclusion générale :

- L'ELO et la différence d'ELO ont un impact fort sur le résultat du match. Un ELO plus élevé pour l'un des joueurs diminue la probabilité de match nul et favorise la victoire de ce joueur.
- Les blunders et les erreurs stratégiques (inferior) jouent un rôle majeur dans le résultat. Les blunders des noirs diminuent leur probabilité de victoire, tandis que les blunders des blancs augmentent les chances de victoire pour White.
- Les autres variables temporelles et les mouvements longs n'ont qu'un faible impact sur le résultat, à l'exception de quelques cas où les coups longs peuvent influencer légèrement le résultat.

Encodage des colonnes non numériques Un véritable défi dans la préparation des données a été l'encodage des colonnes catégorielles, notamment celles avec un très grand nombre de catégories distinctes. Dans de tels cas, utiliser un encodage classique comme le one-hot encoding aurait été impraticable. Cela aurait non seulement multiplié la taille des données de manière exponentielle, mais également introduit des matrices très clairsemées et difficiles à exploiter efficacement dans les modèles.

De même, l'encodage par label (Label Encoding), qui associe un entier unique à chaque catégorie, aurait pu introduire un biais implicite d'ordre dans certaines configurations de modèles (notamment les modèles linéaires), ce qui n'était pas souhaitable.

Pour surmonter ces limitations, nous avons opté pour le target encoding, une approche plus adaptée. Cette méthode consiste à remplacer chaque catégorie par une valeur numérique calculée en fonction de la cible (par exemple, le taux moyen de victoire pour chaque catégorie). Cela nous a permis de conserver l'information tout en réduisant significativement la complexité des données et en améliorant leur utilisabilité dans les modèles d'apprentissage.

[100]:
```
# Sauver état de df_preparation
df_preparation_original = df_preparation
```

[101]:
```
# Observation des types des colonnes pour savoir comment les traiter
schema = df_preparation.schema
columns_by_type = defaultdict(list)

for field in schema:
    columns_by_type[str(field.dataType)].append(field.name)

for data_type, columns in columns_by_type.items():
    print(f"Type: {data_type}")
    print(f"Columns: {columns}\n")
```

```
Type: IntegerType()
Columns: ['BlackElo', 'WhiteElo', 'starting_time', 'increment', 'Total_moves',
'Black_blunders', 'White_blunders', 'Black_mistakes', 'White_mistakes',
'Black_inaccuracies', 'White_inaccuracies', 'Black_inferior_moves',
'White_inferior_moves', 'Black_ts_moves', 'White_ts_moves', 'Black_ts_blunders',
'White_ts_blunders', 'Black_ts_mistakes', 'White_ts_mistake',
'Black_long_moves', 'White_long_moves', 'Black_bad_long_moves',
'White_bad_long_moves', 'Game_flips', 'Game_flips_ts', 'ELO_diff',
```

```
'Result_index']

Type: StringType()
Columns: ['ECO', 'Opening', 'Result', 'Termination', 'Game_type']

Type: DoubleType()
Columns: ['white_moves', 'black_moves']
```

Il y a 5 colonnes de type chaine de caractères, nous allons les encoder afin de pouvoir les utiliser dans nos prédictions.

[102]:
```
# Combien de données différentes contiennent chacune de ces colonnes ?
for column in columns_by_type["StringType()"]:
  distinct_count = df_preparation.select(column).distinct().count()
  print(f"Nombre de valeurs distinctes pour la colonne '{column}':␣
  ↪{distinct_count}")
```

```
Nombre de valeurs distinctes pour la colonne 'ECO': 492
Nombre de valeurs distinctes pour la colonne 'Opening': 2789
Nombre de valeurs distinctes pour la colonne 'Result': 3
Nombre de valeurs distinctes pour la colonne 'Termination': 4
Nombre de valeurs distinctes pour la colonne 'Game_type': 4
```

Nous allons gérer ces colonnes de façons différentes, en fonction de leur nombre de valeurs possibles et de leur type.

- `Result` : Colonne à prédire (déjà encodée avant l'analyse par corrélation)
- `ECO`, `Opening` : Beaucoup de valeurs possibles
- `Termination`, `Game_type` : Peu de valeurs

[103]:
```
# Supprimer alors Result
df_preparation = df_preparation.drop("Result")
```

[104]:
```
# Game_type : 4 valeurs, pseudo relation d'ordre
from pyspark.sql.functions import col # important de laisser l'import de nouveau

df_preparation = df_preparation.withColumn(
    "Game_type_encoded",
    when(col("Game_type") == "Bullet", 1)
    .when(col("Game_type") == "Blitz", 2)
    .when(col("Game_type") == "Rapid", 3)
    .when(col("Game_type") == "Classical", 4)
)

# vérifier
df_preparation.select("Game_type", "Game_type_encoded").distinct().show()
```

```
+---------+-----------------+
|Game_type|Game_type_encoded|
```

```
+---------+----------------+
|   Bullet|               1|
|    Blitz|               2|
|Classical|               4|
|    Rapid|               3|
+---------+----------------+
```

[105]:
```
# Supprimer Game_type
df_preparation = df_preparation.drop("Game_type")
```

[106]:
```
# copy du df pour pas tout perdre
data = df_preparation
```

[107]:
```
from pyspark.sql.functions import mean
# Target encoding pour "Opening"
avg_opening_result = data.groupBy("Opening").agg(mean("Result_index").
 ↪alias("opening_score"))
data = data.join(avg_opening_result, on="Opening", how="left")

# Target encoding pour "Eco"
avg_eco_result = data.groupBy("Eco").agg(mean("Result_index").
 ↪alias("eco_score"))
data = data.join(avg_eco_result, on="Eco", how="left")

# Vérification des colonnes ajoutées
data.select("Opening", "opening_score", "Eco", "eco_score").show(10)
```

```
+--------------------+-----------------+---+-----------------+
|             Opening|     opening_score|Eco|         eco_score|
+--------------------+-----------------+---+-----------------+
|King's Indian Def…|0.9078947368421053|E73|0.9171314741035856|
|Queen's Pawn Game…|0.8923074899371235|D00|0.9834913183473263|
|   Queen's Pawn Game|1.0926208031198585|D00|0.9834913183473263|
|French Defense: K…|0.9988988021664382|C00|0.9748934957282736|
|Nimzo-Indian Defe…|1.0746268656716418|E32|0.9526813880126183|
|Sicilian Defense:…|1.0036543675586467|B23|0.9675707741488976|
|Four Knights Game…|1.0389320779261002|C46|1.0021847107592337|
|    Alekhine Defense|0.9672776224500362|B03| 0.987273185483871|
|        Réti Opening| 1.015946773682543|A04|0.9337475642486857|
|        Bird Opening|1.0258256378149946|A02|1.0295773692422494|
+--------------------+-----------------+---+-----------------+
only showing top 10 rows
```

[108]:
```
# Supprimer Opening et Eco
data = data.drop("Opening", "Eco")
```

[109]:
```
# Quelles sont les valeurs de Termination ?
data.select("Termination").distinct().show()
```

```
+----------------+
|     Termination|
+----------------+
|       Abandoned|
|Rules infraction|
|    Time forfeit|
|          Normal|
+----------------+
```

[110]:
```
# Termination : 4 valeurs possibles
# One hot encoding pour éviter relations d'ordre implicite

# Créer Termination_Abandoned, Termination_Rules_infraction,␣
 ↪Termination_Time_forfeit, Termination_Normal

data = data.withColumn(
    "Termination_Abandoned",
    when(col("Termination") == "Abandoned", 1)
    .otherwise(0)
)

data = data.withColumn(
    "Termination_Rules_infraction",
    when(col("Termination") == "Rules infraction", 1)
    .otherwise(0)
)

data = data.withColumn(
    "Termination_Time_forfeit",
    when(col("Termination") == "Time forfeit", 1)
    .otherwise(0)
)

data = data.withColumn(
    "Termination_Normal",
    when(col("Termination") == "Normal", 1)
    .otherwise(0)
)

# Afficher Termination et les 4 colonnes créées
data.select("Termination", "Termination_Abandoned",␣
 ↪"Termination_Rules_infraction", "Termination_Time_forfeit",␣
 ↪"Termination_Normal").show(10)
```

```
+------------+--------------------+--------------------------+--------------
---------+-----------------+
| Termination|Termination_Abandoned|Termination_Rules_infraction|Termination_Tim
e_forfeit|Termination_Normal|
+------------+--------------------+--------------------------+--------------
---------+-----------------+
|Time forfeit|                   0|                         0|
1|                0|
|      Normal|                   0|                         0|
0|                1|
|Time forfeit|                   0|                         0|
1|                0|
|      Normal|                   0|                         0|
0|                1|
|      Normal|                   0|                         0|
0|                1|
|      Normal|                   0|                         0|
0|                1|
|      Normal|                   0|                         0|
0|                1|
|Time forfeit|                   0|                         0|
1|                0|
|      Normal|                   0|                         0|
0|                1|
|Time forfeit|                   0|                         0|
1|                0|
+------------+--------------------+--------------------------+--------------
---------+-----------------+
only showing top 10 rows
```

[111]:
```
# Supprimer Termination
data = data.drop("Termination")
```

[112]:
```
# Vérifier les colonnes
data.printSchema()
```

```
root
 |-- BlackElo: integer (nullable = true)
 |-- WhiteElo: integer (nullable = true)
 |-- starting_time: integer (nullable = true)
 |-- increment: integer (nullable = true)
 |-- Total_moves: integer (nullable = true)
 |-- Black_blunders: integer (nullable = true)
 |-- White_blunders: integer (nullable = true)
 |-- Black_mistakes: integer (nullable = true)
 |-- White_mistakes: integer (nullable = true)
 |-- Black_inaccuracies: integer (nullable = true)
```

```
 |-- White_inaccuracies: integer (nullable = true)
 |-- Black_inferior_moves: integer (nullable = true)
 |-- White_inferior_moves: integer (nullable = true)
 |-- Black_ts_moves: integer (nullable = true)
 |-- White_ts_moves: integer (nullable = true)
 |-- Black_ts_blunders: integer (nullable = true)
 |-- White_ts_blunders: integer (nullable = true)
 |-- Black_ts_mistakes: integer (nullable = true)
 |-- White_ts_mistake: integer (nullable = true)
 |-- Black_long_moves: integer (nullable = true)
 |-- White_long_moves: integer (nullable = true)
 |-- Black_bad_long_moves: integer (nullable = true)
 |-- White_bad_long_moves: integer (nullable = true)
 |-- Game_flips: integer (nullable = true)
 |-- Game_flips_ts: integer (nullable = true)
 |-- white_moves: double (nullable = true)
 |-- black_moves: double (nullable = true)
 |-- ELO_diff: integer (nullable = true)
 |-- Result_index: integer (nullable = true)
 |-- Game_type_encoded: integer (nullable = true)
 |-- opening_score: double (nullable = true)
 |-- eco_score: double (nullable = true)
 |-- Termination_Abandoned: integer (nullable = false)
 |-- Termination_Rules_infraction: integer (nullable = false)
 |-- Termination_Time_forfeit: integer (nullable = false)
 |-- Termination_Normal: integer (nullable = false)
```

Normalisation des données

[113]:
```python
# Quelles sont les colonnes numériques ? (normalement toutes)
numeric_cols = [col[0] for col in data.dtypes if col[1] in ["int", "double"]]
print(f'Les colonnes numériques sont : {numeric_cols}')

# Quelles sont les colonnes non numériques ?
non_numeric_cols = [col[0] for col in data.dtypes if col[1] not in ["int",␣
  ↪"double"]]
print(f'Les colonnes non numériques sont : {non_numeric_cols}')
```

```
Les colonnes numériques sont : ['BlackElo', 'WhiteElo', 'starting_time',
'increment', 'Total_moves', 'Black_blunders', 'White_blunders',
'Black_mistakes', 'White_mistakes', 'Black_inaccuracies', 'White_inaccuracies',
'Black_inferior_moves', 'White_inferior_moves', 'Black_ts_moves',
'White_ts_moves', 'Black_ts_blunders', 'White_ts_blunders', 'Black_ts_mistakes',
'White_ts_mistake', 'Black_long_moves', 'White_long_moves',
'Black_bad_long_moves', 'White_bad_long_moves', 'Game_flips', 'Game_flips_ts',
'white_moves', 'black_moves', 'ELO_diff', 'Result_index', 'Game_type_encoded',
'opening_score', 'eco_score', 'Termination_Abandoned',
'Termination_Rules_infraction', 'Termination_Time_forfeit',
```

```
'Termination_Normal']
Les colonnes non numériques sont : []
```

[114]:
```python
# Retirer Result_index
numeric_cols.remove("Result_index")

# Combiner les colonnes numériques dans un seul vecteur pour la standardisation
assembler = VectorAssembler(inputCols=numeric_cols,␣
 ↪outputCol="numeric_features")
data = assembler.transform(data)

# Standardiser les données
scaler = StandardScaler(inputCol="numeric_features",␣
 ↪outputCol="scaled_features", withMean=True, withStd=True)
scaler_model = scaler.fit(data)
data = scaler_model.transform(data)

# Transformer le vecteur en colonnes séparées
data = data.withColumn("scaled_features_array",␣
 ↪vector_to_array(col("scaled_features")))

# Réattribuer chaque colonne standardisée à son nom d'origine
for i, col_name in enumerate(numeric_cols):
    data = data.withColumn(f"{col_name}_scaled",␣
 ↪col("scaled_features_array")[i])

# Supprimer les colonnes intermédiaires si nécessaire
data = data.drop("numeric_features", "scaled_features", "scaled_features_array")

# Afficher un aperçu des données
data.select(*[f"{col}_scaled" for col in numeric_cols]).show(5)
```

```
+-----------------+-----------------+-------------------+-------------------
+-------------------+--------------------+--------------------+-------------
-------+--------------------+------------------------+-----------------------
-+---------------------------+---------------------------+--------------------
+---------------------+-----------------------+-----------------------+-------
----------------+----------------------+----------------------+-------------
---------+---------------------------+---------------------------+-------------
------+-------------------+-------------------+-------------------+----------
---------+-----------------------+-------------------+-------------------+----
-----------------------+-----------------------------------+------------------
-----------+-------------------------+
|   BlackElo_scaled|   WhiteElo_scaled|starting_time_scaled|
increment_scaled|  Total_moves_scaled|Black_blunders_scaled|White_blunders_scale
d|Black_mistakes_scaled|White_mistakes_scaled|Black_inaccuracies_scaled|White_in
accuracies_scaled|Black_inferior_moves_scaled|White_inferior_moves_scaled|Black_
ts_moves_scaled|White_ts_moves_scaled|Black_ts_blunders_scaled|White_ts_blunders
```

```
_scaled|Black_ts_mistakes_scaled|White_ts_mistake_scaled|Black_long_moves_scaled
|White_long_moves_scaled|Black_bad_long_moves_scaled|White_bad_long_moves_scaled
|   Game_flips_scaled|Game_flips_ts_scaled|  white_moves_scaled|
black_moves_scaled|
ELO_diff_scaled|Game_type_encoded_scaled|opening_score_scaled|   eco_score_scale
d|Termination_Abandoned_scaled|Termination_Rules_infraction_scaled|Termination_T
ime_forfeit_scaled|Termination_Normal_scaled|
+------------------+------------------+--------------------+--------------------
+--------------------+--------------------+--------------------+-------------
-------+--------------------+------------------------+-----------------------
-+---------------------------+---------------------------+--------------------
+--------------------+-----------------------+-----------------------+-------
----------------+-----------------------+-----------------------+-------------
---------+---------------------------+---------------------------+-------------
------+--------------------+--------------------+--------------------+----------
---------+------------------------+--------------------+--------------------+----
------------------------+-----------------------------------+------------------
-----------+-------------------------+
| 2.821507437134427| 2.608464914980534|-0.47872223681890813|
0.07552217113453705| -1.0869759843797009|   -1.041748755998445|
-1.0243111693967832|  -1.3207574061539318|  -0.8827759160551474|
0.5433572237938805|     -0.39012390587828777|        -0.9254707404238817|
-1.1109560827469414|  -0.3542082211881719|  -0.3561006447979297|
-0.26632160174511016|    -0.26880918763882045|    -0.22198307725312016|
-0.22446918232884344|     1.1253200980749132|     -0.675338115546342|
-0.4978591708336349|       -0.49697629003981186|
-1.1088364527771444|-0.28488330519902905|  -1.068957454927526|
-1.1045985732279289|-0.5022041103911916|    -0.15363790997085056|
0.7418524488196845|-0.6122626917830395|        -7.32301894843026…|
-0.00936414133821…|            -0.5513406423096711|       0.5514766714508069|
| 2.815689823491864|2.6142823420151733|-0.47872223681890813|
0.07552217113453705|   1.084433501836981|  -0.5319731413848415|
-0.5179304672424869| -0.03543011304704095|  -0.8827759160551474|
0.051976682914644017|       0.5905802678148901|        -0.2349003034642157|
-0.42943192685693715|  -0.3542082211881719|    3.157434500174238|
-0.26632160174511016|      1.2197959696296632|    -0.22198307725312016|
1.528352174673588|     0.2234245714290354|     2.0270217707121096|
-0.4978591708336349|         1.0672531949834678| 0.15703499382135386|
1.62337116797626|    1.10302037186513|  1.0654835300919008|-0.4747656687167645|
-0.15363790997085056|  0.7418524488196845|-0.6122626917830395|
-7.32301894843026…|               -0.00936414133821…|
-0.5513406423096711|       0.5514766714508069|
|0.6253582870667844|1.5787803298493885| -0.7740817464211995|
-0.1578919409693436|-0.18846171422107386|  -0.5319731413848415|
-1.0243111693967832|   0.3930123179885893| -0.03063762225753736|
2.0174988464315904|       0.1002281809683012|         0.9160504248018942|
-0.42943192685693715|  0.20404239141333624|  -0.3561006447979297|
1.2411044750967586|    -0.26880918763882045|      1.5540782198688878|
```

```
-0.22446918232884344|     1.1253200980749132|    0.22544851320647524|
1.069128016211841|          1.0672531949834678| -0.6868793039109783|-
0.28488330519902905|-0.17020800935815125|-0.20663356495765453|
2.2484996674701305|     -1.4278018187334303|
0.7418524488196845|-0.6122626917830395|        -7.32301894843026…|
-0.00936414133821…|             1.8137602330164697|      -1.8133128446508393|
|1.2594781741061831| 1.590415183918667|-0.18336272721661676|
0.3089362832384177|  1.6460049206861231|   -1.041748755998445|
-1.0243111693967832| -0.03543011304704095|  0.39543152464126763|
-0.4394038579645925|       2.5519886152012465|        -0.6952805947706597|
0.9336163849230715|  -0.3542082211881719|    2.602665793073369|
-0.26632160174511016|    -0.26880918763882045|    -0.22198307725312016|
-0.22446918232884344|    -0.6784709552168422|     2.0270217707121096|
-0.4978591708336349|
1.0672531949834678|-0.05394358061172…|-0.28488330519902905|
1.6272908817805984|  1.6641268689387503| 0.7805430378882778|
-0.15363790997085056|  0.7418524488196845|-0.6122626917830395|
-7.32301894843026…|               -0.00936414133821…|
-0.5513406423096711|       0.5514766714508069|
|1.1576699353613256|1.0639380372838159|
1.2934348207948403|-0.39130605307322425|-0.26333790340095947|
-1.041748755998445|  -1.0243111693967832|   0.8214547490242197|
-0.8827759160551474|     0.051976682914644017|       0.5905802678148901|
-0.00471015781099372|        -0.6566066454869386|  -0.3542082211881719|
-0.3561006447979297|    -0.26632160174511016|    -0.26880918763882045|
-0.22198307725312016|   -0.22446918232884344|     0.2234245714290354|
0.22544851320647524|        -0.4978591708336349|       -0.49697629003981186|
-0.6868793039109783|-0.28488330519902905| -0.2451037964889325|
-0.2814639823135107|-0.2209600832283133|      1.1205259987917293|
0.7418524488196845|-0.6122626917830395|        -7.32301894843026…|
-0.00936414133821…|             -0.5513406423096711|       0.5514766714508069|
+-----------------+-----------------+-------------------+--------------------
+--------------------+--------------------+---------------------+-------------
-------+--------------------+------------------------+-----------------------
-+--------------------------+-------------------------+--------------------
+--------------------+----------------------+----------------------+-------
----------------+----------------------+----------------------+-------------
---------+--------------------------+--------------------------+-------------
------+-------------------+-------------------+-------------------+----------
---------+-----------------------+-------------------+-------------------+----
-----------------------+-----------------------------------+------------------
-----------+------------------------+
only showing top 5 rows
```

[115]:
```
# Extraire toutes les colonnes scaled et Result_index
scaled_cols = [col_name for col_name in data.columns if "_scaled" in col_name]
```

```
df_scaled = data.select(*scaled_cols, "Result_index")

feature_columns = df_scaled.columns
feature_columns
```

[115]:
```
['BlackElo_scaled',
 'WhiteElo_scaled',
 'starting_time_scaled',
 'increment_scaled',
 'Total_moves_scaled',
 'Black_blunders_scaled',
 'White_blunders_scaled',
 'Black_mistakes_scaled',
 'White_mistakes_scaled',
 'Black_inaccuracies_scaled',
 'White_inaccuracies_scaled',
 'Black_inferior_moves_scaled',
 'White_inferior_moves_scaled',
 'Black_ts_moves_scaled',
 'White_ts_moves_scaled',
 'Black_ts_blunders_scaled',
 'White_ts_blunders_scaled',
 'Black_ts_mistakes_scaled',
 'White_ts_mistake_scaled',
 'Black_long_moves_scaled',
 'White_long_moves_scaled',
 'Black_bad_long_moves_scaled',
 'White_bad_long_moves_scaled',
 'Game_flips_scaled',
 'Game_flips_ts_scaled',
 'white_moves_scaled',
 'black_moves_scaled',
 'ELO_diff_scaled',
 'Game_type_encoded_scaled',
 'opening_score_scaled',
 'eco_score_scaled',
 'Termination_Abandoned_scaled',
 'Termination_Rules_infraction_scaled',
 'Termination_Time_forfeit_scaled',
 'Termination_Normal_scaled',
 'Result_index']
```

[116]:
```
feature_columns.remove("Result_index")
```

[117]:
```
# Combinaison des colonnes finales
assembler_final = VectorAssembler(inputCols=feature_columns,␣
 ↪outputCol="features")
```

```
df_final = assembler_final.transform(df_scaled)
```

[118]:
```
df_final.show(5)
```

```
+-----------------+-----------------+-------------------+--------------------
+--------------------+--------------------+--------------------+-------------
-------+--------------------+------------------------+----------------------
-+---------------------------+---------------------------+--------------------
+--------------------+------------------------+----------------------+-------
----------------+----------------------+----------------------+-------------
---------+---------------------------+---------------------------+-------------
------+--------------------+--------------------+--------------------+----------
---------+------------------------+--------------------+--------------------+----
----------------------+-----------------------------------+------------------
-----------+-------------------------+-----------+--------------------+
|   BlackElo_scaled|   WhiteElo_scaled|starting_time_scaled|
increment_scaled|  Total_moves_scaled|Black_blunders_scaled|White_blunders_scale
d|Black_mistakes_scaled|White_mistakes_scaled|Black_inaccuracies_scaled|White_in
accuracies_scaled|Black_inferior_moves_scaled|White_inferior_moves_scaled|Black_
ts_moves_scaled|White_ts_moves_scaled|Black_ts_blunders_scaled|White_ts_blunders
_scaled|Black_ts_mistakes_scaled|White_ts_mistake_scaled|Black_long_moves_scaled
|White_long_moves_scaled|Black_bad_long_moves_scaled|White_bad_long_moves_scaled
|   Game_flips_scaled|Game_flips_ts_scaled|  white_moves_scaled|
black_moves_scaled|
ELO_diff_scaled|Game_type_encoded_scaled|opening_score_scaled|   eco_score_scale
d|Termination_Abandoned_scaled|Termination_Rules_infraction_scaled|Termination_T
ime_forfeit_scaled|Termination_Normal_scaled|Result_index|            features|
+-----------------+-----------------+-------------------+--------------------
+--------------------+--------------------+--------------------+-------------
-------+--------------------+------------------------+----------------------
-+---------------------------+---------------------------+--------------------
+--------------------+------------------------+----------------------+-------
----------------+----------------------+----------------------+-------------
---------+---------------------------+---------------------------+-------------
------+--------------------+--------------------+--------------------+----------
---------+------------------------+--------------------+--------------------+----
----------------------+-----------------------------------+------------------
-----------+-------------------------+-----------+--------------------+
| 2.821507437134427| 2.608464914980534|-0.47872223681890813|
0.07552217113453705| -1.0869759843797009|   -1.041748755998445|
-1.0243111693967832|  -1.3207574061539318|  -0.8827759160551474|
0.5433572237938805|     -0.39012390587828777|        -0.9254707404238817|
-1.1109560827469414|  -0.3542082211881719|  -0.3561006447979297|
-0.26632160174511016|    -0.26880918763882045|    -0.22198307725312016|
-0.22446918232884344|     1.1253200980749132|     -0.675338115546342|
-0.4978591708336349|       -0.49697629003981186|
-1.1088364527771444|-0.28488330519902905|  -1.068957454927526|
-1.1045985732279289|-0.5022041103911916|    -0.15363790997085056|
```

```
0.7418524488196845|-0.6122626917830395|        -7.32301894843026…|
-0.00936414133821…|             -0.5513406423096711|       0.5514766714508069|
1|[2.82150743713442…|
| 2.815689823491864|2.6142823420151733|-0.47872223681890813|
0.07552217113453705|   1.084433501836981|  -0.5319731413848415|
-0.5179304672424869| -0.03543011304704095|  -0.8827759160551474|
0.051976682914644017|       0.5905802678148901|        -0.2349003034642157|
-0.42943192685693715|  -0.3542082211881719|    3.157434500174238|
-0.26632160174511016|      1.2197959696296632|    -0.22198307725312016|
1.528352174673588|     0.2234245714290354|     2.0270217707121096|
-0.4978591708336349|         1.0672531949834678| 0.15703499382135386|
1.62337116797626|    1.10302037186513|  1.0654835300919008|-0.4747656687167645|
-0.15363790997085056|  0.7418524488196845|-0.6122626917830395|
-7.32301894843026…|               -0.00936414133821…|
-0.5513406423096711|       0.5514766714508069|           1|[2.81568982349186…|
|0.6253582870667844|1.5787803298493885| -0.7740817464211995|
-0.1578919409693436|-0.18846171422107386|  -0.5319731413848415|
-1.0243111693967832|   0.3930123179885893| -0.03063762225753736|
2.0174988464315904|       0.1002281809683012|         0.9160504248018942|
-0.42943192685693715|  0.20404239141333624|  -0.3561006447979297|
1.2411044750967586|    -0.26880918763882045|      1.5540782198688878|
-0.22446918232884344|     1.1253200980749132|    0.22544851320647524|
1.069128016211841|         1.0672531949834678| -0.6868793039109783|-
0.28488330519902905|-0.17020800935815125|-0.20663356495765453|
2.2484996674701305|     -1.4278018187334303|
0.7418524488196845|-0.6122626917830395|        -7.32301894843026…|
-0.00936414133821…|             1.8137602330164697|      -1.8133128446508393|
0|[0.62535828706678…|
|1.2594781741061831| 1.590415183918667|-0.18336272721661676|
0.3089362832384177|  1.6460049206861231|   -1.041748755998445|
-1.0243111693967832| -0.03543011304704095|  0.39543152464126763|
-0.4394038579645925|       2.5519886152012465|        -0.6952805947706597|
0.9336163849230715|  -0.3542082211881719|    2.602665793073369|
-0.26632160174511016|    -0.26880918763882045|    -0.22198307725312016|
-0.22446918232884344|    -0.6784709552168422|     2.0270217707121096|
-0.4978591708336349|
1.0672531949834678|-0.05394358061172…|-0.28488330519902905|
1.6272908817805984|  1.6641268689387503| 0.7805430378882778|
-0.15363790997085056|  0.7418524488196845|-0.6122626917830395|
-7.32301894843026…|               -0.00936414133821…|
-0.5513406423096711|       0.5514766714508069|           2|[1.25947817410618…|
|1.1576699353613256|1.0639380372838159|
1.2934348207948403|-0.39130605307322425|-0.26333790340095947|
-1.041748755998445|  -1.0243111693967832|   0.8214547490242197|
-0.8827759160551474|     0.051976682914644017|       0.5905802678148901|
-0.00471015781099372|        -0.6566066454869386|  -0.3542082211881719|
-0.3561006447979297|    -0.26632160174511016|    -0.26880918763882045|
-0.22198307725312016|   -0.22446918232884344|     0.2234245714290354|
```

```
0.22544851320647524|         -0.4978591708336349|        -0.49697629003981186|
-0.6868793039109783|-0.28488330519902905| -0.2451037964889325|
-0.2814639823135107|-0.2209600832283133|       1.1205259987917293|
0.7418524488196845|-0.6122626917830395|         -7.32301894843026…|
-0.00936414133821…|              -0.5513406423096711|        0.5514766714508069|
0|[1.15766993536132…|
+------------------+------------------+--------------------+--------------------
+--------------------+--------------------+--------------------+-------------
-------+--------------------+------------------------+------------------------
-+--------------------------+--------------------------+--------------------
+--------------------+------------------------+------------------------+-------
-----------------+------------------------+------------------------+-------------
---------+--------------------------+--------------------------+-------------
------+--------------------+--------------------+--------------------+----------
---------+------------------------+--------------------+-------------------+----
------------------------+-----------------------------------+------------------
------------+--------------------------+------------+--------------------+
only showing top 5 rows
```

[119]:
```
df_features = df_final.select("features", "Result_index")
```

Train test split

[120]:
```
# Diviser les données en 80% pour l'entrainement et 20% pour le test
train_data, test_data = df_features.randomSplit([0.8, 0.2], seed=42)
```

[121]:
```
# Utiliser un sous-ensemble des données (1% des données d'entraînement)
small_train_data = train_data.sample(withReplacement=False, fraction=0.01,␣
 ↪seed=1234)
```

Régression multinomiale Pour entraîner notre modèle de régression multinomiale, nous avons utilisé un sous-échantillon représentant 1 % des données d'entraînement afin de réduire le temps de calcul et d'optimiser les performances. Le modèle a été évalué sur un ensemble de test distinct.

[122]:
```
# Environ 5 minutes avec un sous sample à 1%
lr = LogisticRegression(featuresCol="features", labelCol="Result_index",␣
 ↪family="multinomial")
model = lr.fit(small_train_data)
```

[123]:
```
predictions = model.transform(test_data)
evaluator = MulticlassClassificationEvaluator(labelCol="Result_index",␣
 ↪predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Accuracy: {accuracy:.4f}")
```

```
Accuracy: 0.9211
```

Le modèle a obtenu une précision (accuracy) de 92.11 %, ce qui signifie que 92 % des prédictions

effectuées étaient correctes. Cela indique que le modèle est capable de bien prédire les résultats des parties (victoire des Blancs, victoire des Noirs ou match nul) à partir des caractéristiques utilisées.

Etant donné que les classes dans les données cibles (Result_index) sont relativement équilibrées, la précision est une métrique appropriée pour évaluer la performance du modèle.

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}}$$

Ce modèle de régression multinomiale s'est révélé efficace pour prédire le résultat des parties avec des performances élevées, même sur un sous-échantillon réduit des données. Cela valide l'approche choisie et la pertinence des caractéristiques sélectionnées pour la prédiction.

[124]:
```
# Récupérer les noms des caractéristiques utilisées dans le modèle
feature_names = assembler.getInputCols()
```

[125]:
```
# Intercept pour chaque classe
intercepts = model.interceptVector
print("Intercepts pour chaque classe :")
for i, intercept in enumerate(intercepts):
    print(f"Classe {i}: {intercept}")
```

```
Intercepts pour chaque classe :
Classe 0: 0.8111877863580168
Classe 1: -1.4895177403734967
Classe 2: 0.6783299540154799
```

[126]:
```
# Coefficient matrix : chaque ligne représente les coefficients pour une classe
coeff_matrix = model.coefficientMatrix
coeff_matrix
```

[126]:
```
DenseMatrix(3, 35, [-0.0774, -0.027, -0.0711, 0.0089, -0.1388, 1.2698, -1.204,
0.2717, …, -0.0912, 0.0184, 0.0686, -0.0023, 0.0, 0.269, 0.1557, -0.158], 1)
```

[127]:
```
# Afficher les coefficients pour chaque classe
for i in range(coeff_matrix.numRows):
    print(f"Classe {i}:")
    for j in range(coeff_matrix.numCols):
        print(f"  {feature_names[j]}: {coeff_matrix[i, j]}")
    print(f"Intercept: {intercepts[i]}")
    print("\n")
```

```
Classe 0:
  BlackElo: -0.07736915560356267
  WhiteElo: -0.02702302123939665
  starting_time: -0.07112350797772558
  increment: 0.008915071843590893
  Total_moves: -0.13882360926608137
  Black_blunders: 1.2697597601522257
```

```
  White_blunders: -1.204008308590287
  Black_mistakes: 0.27169003165395667
  White_mistakes: -0.2540091921978337
  Black_inaccuracies: -0.03817016569134492
  White_inaccuracies: 0.033365104894202045
  Black_inferior_moves: 0.7092695105419098
  White_inferior_moves: -0.6639070976017397
  Black_ts_moves: 0.16966220913989644
  White_ts_moves: -0.12226459147938878
  Black_ts_blunders: 0.04592007969860644
  White_ts_blunders: -0.05662144063002851
  Black_ts_mistakes: -0.0029951156520558807
  White_ts_mistake: -0.010008083925987631
  Black_long_moves: 0.14595695388676158
  White_long_moves: -0.233136860695801
  Black_bad_long_moves: -0.04379094364467315
  White_bad_long_moves: 0.07188701915250867
  Game_flips: -0.1284645837750833
  Game_flips_ts: -0.024766198703859876
  white_moves: 40.2227332671894
  black_moves: -40.47650179773016
  ELO_diff: 0.11989343773483457
  Game_type_encoded: 0.06066074394380462
  opening_score: -0.017965199519032596
  eco_score: -0.012059604834978165
  Termination_Abandoned: 0.0
  Termination_Rules_infraction: -0.1995971950421267
  Termination_Time_forfeit: 0.16149919422110606
  Termination_Normal: -0.1598187764989659
Intercept: 0.8111877863580168


Classe 1:
  BlackElo: 0.08935909205420099
  WhiteElo: 0.07722609031551156
  starting_time: 0.0802776088620373
  increment: 0.01661544218365697
  Total_moves: 0.2245418887821591
  Black_blunders: -0.12768461655118174
  White_blunders: -0.0072385612354655925
  Black_mistakes: -0.048452986426376785
  White_mistakes: -0.02450857619093093
  Black_inaccuracies: 0.0029614367358083904
  White_inaccuracies: -0.019209759322526408
  Black_inferior_moves: -0.08303326071733941
  White_inferior_moves: -0.025131142727438226
  Black_ts_moves: -0.06582791407578678
  White_ts_moves: -0.027238510845652243
```

```
  Black_ts_blunders: 0.05504602303348835
  White_ts_blunders: -0.03769418082278843
  Black_ts_mistakes: -0.00782782073199424
  White_ts_mistake: -0.00463529231547349
  Black_long_moves: 0.02766658639755247
  White_long_moves: 0.059729494749241774
  Black_bad_long_moves: 0.007310149956690249
  White_bad_long_moves: -0.052073233399407316
  Game_flips: 0.25151938756785425
  Game_flips_ts: 0.08102719270174072
  white_moves: 0.635273049565436
  black_moves: -0.18602661396004472
  ELO_diff: -0.028646381902494647
  Game_type_encoded: -0.07904039120118686
  opening_score: -0.05065685411330523
  eco_score: 0.01437656370981958
  Termination_Abandoned: 0.0
  Termination_Rules_infraction: -0.06942653771145073
  Termination_Time_forfeit: -0.317185805924029
  Termination_Normal: 0.3177824246902544
Intercept: -1.4895177403734967


Classe 2:
  BlackElo: -0.011989936450638323
  WhiteElo: -0.0502030690761149
  starting_time: -0.009154100884311721
  increment: -0.02553051402724786
  Total_moves: -0.08571827951607776
  Black_blunders: -1.142075143601044
  White_blunders: 1.2112468698257528
  Black_mistakes: -0.22323704522757987
  White_mistakes: 0.2785177683887647
  Black_inaccuracies: 0.03520872895553653
  White_inaccuracies: -0.014155345571675635
  Black_inferior_moves: -0.6262362498245705
  White_inferior_moves: 0.6890382403291778
  Black_ts_moves: -0.10383429506410966
  White_ts_moves: 0.14950310232504102
  Black_ts_blunders: -0.10096610273209479
  White_ts_blunders: 0.09431562145281694
  Black_ts_mistakes: 0.010822936384050122
  White_ts_mistake: 0.014643376241461123
  Black_long_moves: -0.17362354028431404
  White_long_moves: 0.1734073659465592
  Black_bad_long_moves: 0.0364807936879829
  White_bad_long_moves: -0.01981378575310135
  Game_flips: -0.12305480379277096
```

```
  Game_flips_ts: -0.05626099399788085
  white_moves: -40.85800631675483
  black_moves: 40.662528411690204
  ELO_diff: -0.09124705583233991
  Game_type_encoded: 0.018379647257382236
  opening_score: 0.06862205363233782
  eco_score: -0.002316958874841416
  Termination_Abandoned: 0.0
  Termination_Rules_infraction: 0.26902373275357744
  Termination_Time_forfeit: 0.15568661170292297
  Termination_Normal: -0.1579636481912885
Intercept: 0.6783299540154799
```

[128]:
```python
# Boucle pour afficher les 5 facteurs les plus influents pour chaque classe
for i in range(coeff_matrix.numRows):
    print(f"Classe {i}:")

    # Extraire les coefficients pour la classe i
    class_coeffs = [coeff_matrix[i, j] for j in range(coeff_matrix.numCols)]  # Liste des coefficients

    # Créer une liste de tuples (coefficient, feature_name) et trier par valeur absolue des coefficients
    feature_coeffs = [(feature_names[j], class_coeffs[j]) for j in range(len(class_coeffs))]

    # Trier les caractéristiques par valeur absolue décroissante des coefficients
    feature_coeffs_sorted = sorted(feature_coeffs, key=lambda x: abs(x[1]), reverse=True)

    # Afficher les 5 facteurs les plus influents
    for feature, coeff in feature_coeffs_sorted[:5]:
        print(f"  {feature}: {coeff}")

    # Afficher l'intercept de la classe
    print(f"Intercept: {intercepts[i]}")
    print("\n")
```

```
Classe 0:
  black_moves: -40.47650179773016
  white_moves: 40.2227332671894
  Black_blunders: 1.2697597601522257
  White_blunders: -1.204008308590287
  Black_inferior_moves: 0.7092695105419098
Intercept: 0.8111877863580168
```

```
Classe 1:
  white_moves: 0.635273049565436
  Termination_Normal: 0.3177824246902544
  Termination_Time_forfeit: -0.317185805924029
  Game_flips: 0.25151938756785425
  Total_moves: 0.2245418887821591
Intercept: -1.4895177403734967


Classe 2:
  white_moves: -40.85800631675483
  black_moves: 40.662528411690204
  White_blunders: 1.2112468698257528
  Black_blunders: -1.142075143601044
  White_inferior_moves: 0.6890382403291778
Intercept: 0.6783299540154799
```

Classe 0 : White gagne :

- `black_moves` (-40.48) : Plus le joueur noir joue de coups, moins il est probable que White gagne.
- `white_moves` (40.22) : Plus le joueur blanc joue de coups, plus il est probable qu'il gagne.

  NB importante : Les variables `white_moves` et `black_moves` sont naturellement corrélées, car les joueurs alternent leurs coups tout au long de la partie. Un grand nombre de coups totaux implique donc nécessairement une contribution importante des deux joueurs. Cela reflète davantage la durée globale de la partie (longue ou courte) qu'une performance individuelle.
- `Black_blunders` (1.27) : Les erreurs majeures des noirs augmentent significativement la probabilité de victoire pour White.
- `White_blunders` (-1.20) : Les blunders des blancs réduisent la probabilité de victoire pour White.
- `Black_inferior_moves` (0.71) : Les mauvais coups globaux des noirs favorisent également la victoire de White.
- `Intercept` (0.811) : Cela indique une tendance de base légèrement positive pour White, en l'absence de contribution des autres variables.

Classe 1 : Match nul :

- `white_moves` (0.64) : Une légère augmentation des mouvements du joueur blanc favorise les matchs nuls. Cet indicateur est peu pertinent, comme déjà expliqué, le nombre de coups joué par white et black est similaire.
- `Termination_Normal` (0.32) : Les parties qui se terminent normalement (pas par abandon ou dépassement de temps) favorisent un match nul. Ce qui est logique.

- `Termination_Time_forfeit` (-0.32) : Un abandon dû au dépassement de temps réduit la probabilité d'un match nul. Ce qui est aussi logique, un forfait implique un gagnant.
- `Game_flips` (0.25) : Les retournements de situation augmentent la probabilité de match nul, ce qui reflète une dynamique équilibrée dans ces parties.
- `Total_moves` (0.22) : Un nombre total élevé de mouvements est associé aux matchs nuls.
- `Intercept` (-1.49) : Cela indique une tendance de base négative pour les matchs nuls, sauf si les variables les favorisent.

Classe 2 : Black gagne :

- `white_moves` (-40.86) : Plus le joueur blanc joue de coups, moins il est probable que Black gagne.
- `black_moves` (40.66) : Plus le joueur noir joue de coups, plus il est probable qu'il gagne.

  Comme dit précédemment, les variables `white_moves` et `black_moves` sont naturellement liées.
- `White_blunders` (1.21) : Les erreurs majeures des blancs augmentent significativement la probabilité de victoire pour Black.
- `Black_blunders` (-1.14) : Les blunders des noirs réduisent la probabilité de victoire pour eux.
- `White_inferior_moves` (0.69) : Les mauvais coups globaux des blancs favorisent également la victoire de Black.
- `Intercept` (0.678) : Cela indique une légère tendance de base favorable à Black.

Conclusion

Ces résultats mettent en évidence des dynamiques intéressantes dans les parties d'échecs : - Les blunders et les mauvais coups (inferior moves) sont des indicateurs clés pour prédire le résultat, avec des impacts inverses selon le joueur (Blanc ou Noir). - Les parties longues et équilibrées semblent favoriser les matchs nuls, tandis que l'activité d'un joueur, mesurée par le nombre de coups, est un facteur déterminant pour les victoires.

Conclusion question 3 Les différentes approches ont permis d'identifier les variables significativement liées au Result :

- Corrélation et covariance ont révélé les tendances générales, montrant notamment quels facteurs influencent les chances de victoire pour les Blancs ou les Noirs.
- L'approche par Machine Learning a permis d'aller plus loin, en étudiant également les facteurs associés à une issue de match nul, ce qui n'était pas accessible via des analyses statistiques plus simples.

Ainsi, ces analyses complémentaires offrent une vision plus globale et détaillée des variables influençant l'issue d'une partie.

Réponse à l'hypothèse :

- `ELO_diff` comme facteur prédominant : Les résultats confirment que la différence d'ELO est effectivement un facteur significatif pour prédire l'issue de la partie, en accord avec la théorie. Une grande différence d'ELO favorise fortement le joueur mieux classé, comme prévu par le modèle de probabilité basé sur une distribution normale.

- Cependant, les résultats montrent que les erreurs (blunders, mistakes) et le contexte (type de partie, dynamique des coups) ajoutent des nuances importantes à cette prédiction. Le facteur temporel lui n'est pas important. Cela suggère que, bien que l'ELO soit un bon indicateur global, l'issue d'une partie peut dépendre de nombreux autres facteurs comportementaux.

1.5 Questions supplémentaires

1.5.1 Distribution des parties nulles selon l'ouverture et le niveau

Étude des cas de parties nulles : - **Quelles ouvertures ont une probabilité plus élevée de mener à une partie nulle ? - Quelle est la distribution des parties nulles en fonction de l'ouverture et des catégories ELO des joueurs ?**

Hypothèses : - Certaines ouvertures symétriques ou solides pourraient favoriser les parties nulles en raison de leur tendance à se stabiliser rapidement. Des ouvertures plus complexes ou agressives tendent à mener à des résultats plus tranchés (gagnant/perdant), tandis que des ouvertures plus passives favorisent des parties plus équilibrées et donc plus de nulles. - Les joueurs avec un ELO plus élevé choisiraient probablement des ouvertures menant à des positions équilibrées, augmentant ainsi la probabilité de match nul. - À l'inverse, les joueurs avec un ELO plus bas seraient plus susceptibles de commettre des erreurs, rendant les parties plus dynamiques et donc moins susceptibles de se terminer par une nulle.

[129]:
```
df_spark_null = df_spark_plus
# ne garder que les données où le Result est 1/2-1/2
df_spark_null = df_spark_null.filter(col("Result") == "1/2-1/2")
```

[130]:
```
# Distribution des parties nulles selon l'ouverture
distribution_null = df_spark_null.groupBy("Opening") \
                                        .agg(count("*").alias("num_draws"),
                                             (count("*") / df_spark.count()).
 ↪alias("draw_rate")) \
                                        .orderBy(col("draw_rate").desc())
```

[131]:
```
distribution_null.show(truncate=False)
```

```
+-----------------------------------------------+---------+----------------------+
|Opening                                        |num_draws|draw_rate             |
+-----------------------------------------------+---------+----------------------+
|Queen's Pawn Game: Mason Attack                |2445     |6.537592224837557E-4  |
|Indian Game                                    |1843     |4.927927390746673E-4  |
|Sicilian Defense                               |1620     |4.3316561980518776E-4|
|Caro-Kann Defense                              |1545     |4.13111655925318E-4   |
|Philidor Defense                               |1528     |4.0856609077921413E-4|
|Scandinavian Defense: Mieses-Kotroc Variation|1434     |3.834317893831107E-4  |
|French Defense: Knight Variation               |1236     |3.304893247402544E-4  |
|Pirc Defense                                   |1166     |3.1177229178570924E-4|
|Sicilian Defense: Old Sicilian                 |1143     |3.056224095292158E-4  |
|Scotch Game                                    |1131     |3.0241377530843667E-4|
|Modern Defense                                 |1126     |3.01076844383112E-4   |
```

```
|Queen's Pawn Game                              |1121     |2.9973991345778737E-4|
|Queen's Pawn Game: Chigorin Variation          |1106     |2.957291206818134E-4 |
|Horwitz Defense                                |1067     |2.8530105946428106E-4|
|Sicilian Defense: Bowdler Attack               |1003     |2.681883436201255E-4 |
|Four Knights Game: Italian Variation           |862      |2.3048689152597028E-4|
|French Defense: Exchange Variation             |845      |2.2594132637986645E-4|
|Van't Kruijs Opening                           |824      |2.203262164935029E-4 |
|Queen's Pawn Game: Zukertort Variation         |761      |2.0348088683441227E-4|
|Queen's Pawn                                   |745      |1.9920270787337338E-4|
+-----------------------------------------------+---------+---------------------+
only showing top 20 rows
```

Les résultats montrent que certaines ouvertures spécifiques ont une probabilité plus élevée de mener à des parties nulles.

- *Queen's Pawn Game: Mason Attack*, une approche calme et solide, a le taux de nulles le plus élevé, avec un `draw_rate` de 0.065 %.
- Les ouvertures classiques et solides comme *Indian Game* (0.050 %), *Sicilian Defense* (0.044 %), et *Caro-Kann Defense* (0.041 %) figurent également parmi les ouvertures avec les taux de nulles les plus élevés, ce qui confirme leur nature équilibrée et stable.

Ces résultats sont cohérents avec l'hypothèse selon laquelle des ouvertures symétriques ou solides tendent à stabiliser les positions, augmentant ainsi la probabilité d'un match nul.

En revanche, des ouvertures plus agressives ou tactiques, comme celles non mentionnées ici, pourraient entraîner davantage de victoires ou de défaites, réduisant leur `draw_rate`.

[132]:
```
# Distribution des parties nulles en fonction de la catégorie ELO des joueurs
distribution_null_by_elo = distribution_null_by_elo = df_spark_null.
 ↪groupBy("Black_ELO_category", "White_ELO_category") \
                                    .agg(count("*").alias("num_draws"),
                                         (count("*") / df_spark.count()).
 ↪alias("draw_rate")) \
                                    .orderBy(col("draw_rate").desc())
```

[133]:
```
distribution_null_by_elo.show(truncate = False)
```

```
+--------------------------------+--------------------------------+---------+---
------------------+
|Black_ELO_category              |White_ELO_category
|num_draws|draw_rate            |
+--------------------------------+--------------------------------+---------+---
------------------+
|good club player                |good club player                |25871
|0.0069175479938148226|
|national and international level|national and international level|18465
|0.004937285907223946 |
|occasional player               |occasional player               |16214
|0.0043353996046427866|
```

```
|very good club player           |very good club player           |15072
|0.004030044581298636 |
|other lower bound               |other lower bound               |6486
|0.0017342667963311407|
|national and international level|very good club player           |3462
|9.256909726947901E-4 |
|very good club player           |good club player                |3362
|8.989523541882971E-4 |
|very good club player           |national and international level|3317
|8.869199758603753E-4 |
|good club player                |very good club player           |3180
|8.502880685064797E-4 |
|occasional player               |good club player                |2187
|5.847735867370035E-4 |
|good club player                |occasional player               |2186
|5.845062005519386E-4 |
|GMI, World Champions            |GMI, World Champions            |2060
|5.508155412337573E-4 |
|GMI, World Champions            |national and international level|1201
|3.211308082629818E-4 |
|national and international level|GMI, World Champions            |1111
|2.97066051607138E-4  |
|occasional player               |other lower bound               |1090
|2.914509417207745E-4 |
|other lower bound               |occasional player               |1052
|2.8129026668830714E-4|
|national and international level|good club player                |672
|1.7968351636363345E-4|
|good club player                |national and international level|561
|1.5000364982142614E-4|
|very good club player           |occasional player               |242
|6.470745678571323E-5 |
|occasional player               |very good club player           |187
|5.0001216607142045E-5|
+--------------------------------+--------------------------------+---------+---
------------------+
only showing top 20 rows
```

- On remarque que les catégories “good club player” pour les deux joueurs (noir et blanc) ont le taux de nulles le plus élevé (0.0069). Cela pourrait suggérer que les joueurs de niveau similaire tendent à jouer des parties plus équilibrées, augmentant la probabilité de match nul. En effet, les 5 premiers résultats sont des parties entre joueurs de même niveau.
- Pour la catégorie “GMI, World Champions”, les taux de nulles sont relativement faibles. Cela pourrait signifier que ces joueurs de très haut niveau sont moins enclins à se laisser entraîner dans des positions égalisées ou à commettre des erreurs permettant de créer une partie nulle. Leur expertise doit leur permettre de forcer une victoire.
- Les joueurs occasionnels ou ceux de catégorie “other lower bound” ont des taux de nulles

beaucoup plus faibles, notamment dans les comparaisons avec des joueurs de niveaux plus élevés. Cela peut être expliqué par un manque de stratégie ou des erreurs qui conduisent plus souvent à des victoires pour l'adversaire plutôt qu'à des parties équilibrées.

- Les joueurs avec des différences d'ELO marquées, comme "national and international level" vs "very good club player", ou "GMI, World Champions" contre "national and international level", ont des taux de match nul relativement faibles, ce qui reflète probablement la différence de compétence et la tendance des joueurs à ne pas se stabiliser dans une position nulle.

La distribution des parties nulles en fonction des catégories ELO montre que les matchs entre joueurs de niveaux similaires ont plus de chances de se terminer par une nulle. En revanche, les grandes différences de niveau (comme entre les joueurs occasionnels et les experts) entraînent généralement des parties avec des résultats plus tranchés (victoire/défaite).

[134]:
```
# Croiser les données des ouvertures et des catégories ELO des joueurs pour␣
 ↪analyser les probabilités de nulles
distribution_null_by_opening_elo = df_spark_null.groupBy("Opening",␣
 ↪"Black_ELO_category", "White_ELO_category") \
                                          .agg(count("*").alias("num_draws"),
                                               (count("*") / df_spark.count()).
 ↪alias("draw_rate")) \
                                          .orderBy(col("draw_rate").desc())
```

[135]:
```
distribution_null_by_opening_elo.show(truncate = False)
```

```
+-----------------------------------------------+--------------------------------
+--------------------------------+---------+----------------------+
|Opening                                        |Black_ELO_category
|White_ELO_category              |num_draws|draw_rate             |
+-----------------------------------------------+--------------------------------
+--------------------------------+---------+----------------------+
|Queen's Pawn Game: Mason Attack                |good club player
|good club player                |699      |1.8690294336038657E-4|
|Queen's Pawn Game: Mason Attack                |occasional player
|occasional player               |545      |1.4572547086038725E-4|
|Philidor Defense                               |good club player
|good club player                |516      |1.3797127149350424E-4|
|Sicilian Defense                               |good club player
|good club player                |483      |1.2914752738636155E-4|
|Indian Game                                    |good club player
|good club player                |438      |1.1711514905843966E-4|
|Philidor Defense                               |occasional player
|occasional player               |405      |1.0829140495129694E-4|
|Scandinavian Defense: Mieses-Kotroc Variation|good club player
|good club player                |405      |1.0829140495129694E-4|
|Caro-Kann Defense                              |good club player
|good club player                |404      |1.0802401876623201E-4|
|Sicilian Defense: Bowdler Attack               |good club player
|good club player                |398      |1.0641970165584242E-4|
```

```
|French Defense: Knight Variation             |good club player
|good club player                |388      |1.0374583980519312E-4|
|Indian Game                                  |national and international
level|national and international level|387      |1.034784536201282E-4 |
|Sicilian Defense: Old Sicilian               |good club player
|good club player                |341      |9.117868910714137E-5 |
|Scotch Game                                  |good club player
|good club player                |339      |9.064391673701152E-5 |
|Scotch Game                                  |occasional player
|occasional player               |309      |8.26223311850636E-5  |
|Queen's Pawn Game                            |good club player
|good club player                |304      |8.128540025973895E-5 |
|Horwitz Defense                              |good club player
|good club player                |296      |7.914631077921949E-5 |
|Sicilian Defense                             |occasional player
|occasional player               |291      |7.780937985389484E-5 |
|Modern Defense                               |good club player
|good club player                |288      |7.700722129870004E-5 |
|Indian Game                                  |very good club player
|very good club player           |283      |7.56702903733754E-5  |
|Queen's Pawn Game: Mason Attack              |very good club player
|very good club player           |282      |7.540290418831047E-5 |
+---------------------------------------------+---------------------------------
+--------------------------------+---------+---------------------+
only showing top 20 rows
```

- Des ouvertures comme "*Queen's Pawn Game: Mason Attack*", "*Philidor Defense*", et "*Sicilian Defense*" sont populaires, notamment parmi les joueurs de niveau "good club player". Cependant, les taux de nulles restent relativement faibles, malgré un nombre important de parties.
- "*Queen's Pawn Game: Mason Attack*" et "*Philidor Defense*" montrent des taux de nulles qui sont plutôt faibles (autour de 0.00018 et 0.00014 respectivement), malgré un nombre significatif de parties, ce qui suggère que bien que populaires, ces ouvertures ne sont pas particulièrement associées à des parties nulles.

Hypothèses et analyse des résultats :

- Les ouvertures symétriques ou solides favorisent les parties nulles :
  - Dans les résultats de la distribution des parties nulles par ouverture, on observe que des ouvertures telles que "*Queen's Pawn Game: Mason Attack*" et "*Philidor Defense*" sont bien présentes, avec des taux de nulles relativement élevés. Cela correspond à l'hypothèse que certaines ouvertures, plus symétriques et solides, favorisent la stabilisation des positions et augmentent la probabilité de match nul. Ces ouvertures sont en effet classiques, connues pour leur tendance à conduire à des positions équilibrées où les deux joueurs ont des chances similaires, ce qui peut mener à plus de parties nulles.
- Les joueurs avec un ELO plus élevé choisissent des ouvertures équilibrées :
  - Les catégories good club player et national and international level sont fréquemment

associées à des ouvertures solides comme "*Sicilian Defense*", "*Queen's Pawn Game*", et "*Philidor Defense*", qui sont également les plus fréquentes dans les résultats de parties nulles.
  - Les joueurs de ces catégories, ayant plus de maîtrise et de stratégie, choisissent probablement des ouvertures qui minimisent les risques de pertes, ce qui pourrait augmenter la probabilité de nulles. Ces ouvertures sont souvent caractérisées par leur stabilité, menant à moins d'erreurs fatales.
  - Cela semble corroborer l'idée que des joueurs avec un ELO élevé tendent à choisir des ouvertures plus équilibrées, ce qui augmente les chances de match nul.
- Les joueurs avec un ELO plus bas sont plus susceptibles de commettre des erreurs et de rendre les parties moins susceptibles de finir par une nulle :
  - Les joueurs d'ELO plus bas (comme les occasional players ou other lower bound) ont tendance à avoir des résultats moins équilibrés, et cela se reflète dans les statistiques des ouvertures. Par exemple, des ouvertures comme "*Scotch Game*" et "*Sicilian Defense*" apparaissent fréquemment parmi les résultats des joueurs avec un ELO plus bas.
  - Ces ouvertures peuvent conduire à plus de dynamisme dans la partie, avec des erreurs plus fréquentes de la part des joueurs moins expérimentés, ce qui rend les matchs plus susceptibles de se terminer par une victoire pour l'un des joueurs, plutôt que par un match nul.

Conclusion Les résultats montrent que des ouvertures solides et symétriques, ainsi que des joueurs avec un ELO plus élevé, semblent favoriser les matchs nuls. En revanche, les joueurs avec un ELO plus bas, avec des ouvertures plus dynamiques ou agressives, ont tendance à avoir moins de matchs nuls. Ces observations confirment en grande partie les hypothèses formulées.

1.5.2 Impact de la différence d'ELO sur la durée d'une partie (en nombre de coups)

Hypothèse à tester : Les parties où la différence d'ELO entre les joueurs est grande ont tendance à durer moins longtemps en nombre de coups. Cela pourrait être dû au fait qu'un joueur plus fort (avec un ELO plus élevé) termine rapidement la partie avec un coup décisif ou un "mat", tandis qu'une partie entre des joueurs avec des ELO similaires pourrait durer plus longtemps en raison d'une lutte plus équilibrée.

Pour tester notre hypothèse, nous allons calculer la corrélation entre la différence d'ELO et la durée de la partie.

[136]:
```
df_spark_diff_ELO = df_spark_plus

from pyspark.sql.functions import col, abs

# Ajouter la colonne de différence d'ELO avec valeur absolue
df_spark_diff_ELO = df_spark_plus.withColumn("ELO_diff", abs(col("WhiteELO") -␣
 ↪col("BlackELO")))
```

[137]:
```
# Créer un VectorAssembler pour combiner les colonnes en un vecteur
assembler = VectorAssembler(inputCols=['ELO_diff', 'Total_moves'],␣
 ↪outputCol='features')
```

```
# Appliquer le VectorAssembler pour transformer les données
assembled_data = assembler.transform(df_spark_diff_ELO)

# Calcul de la corrélation
correlation_matrix = Correlation.corr(assembled_data, 'features',␣
 ↪method='pearson')

# Affichage de la matrice de corrélation
print(f"Correlation Matrix:\n{correlation_matrix.head()}")
```

```
Correlation Matrix:
Row(pearson(features)=DenseMatrix(2, 2, [1.0, -0.0077, -0.0077, 1.0], False))
```

Corrélation très faible : La valeur de la corrélation -0.0077 indique une très faible relation entre la différence d'ELO et le nombre de coups dans la partie. La différence d'ELO entre les joueurs ne semble pas avoir un impact significatif sur la durée de la partie (en termes de nombre de coups).

La corrélation, ayant une valeur proche de 0, signifie qu'il n'y a pas de lien linéaire évident entre ces deux variables. Et la corrélation, étant négative, indique que si un lien existe, il est inverse, c'est-à-dire que des différences d'ELO plus grandes pourraient être associées à des parties plus courtes, mais dans ce cas, l'effet est tellement faible que cela ne constitue pas une relation significative.

Réflexion sur l'hypothèse :
L'hypothèse selon laquelle les parties avec une grande différence d'ELO durent moins longtemps n'est pas confirmée par ces résultats. En effet, la corrélation très faible suggère que d'autres facteurs peuvent jouer un rôle plus important dans la durée de la partie que la différence d'ELO entre les joueurs.

2 Conclusion

Ce projet a permis d'explorer en profondeur un grand ensemble de données issues des parties d'échecs jouées sur Lichess.

Grâce aux résultats apportés aux questions proposées et aux problématiques supplémentaires, nous avons pu mettre en lumière plusieurs aspects du jeu d'échecs et offrir des perspectives intéressantes pour mieux comprendre les dynamiques du jeu.

Nous avons constaté que le **taux d'erreurs** (blunders, erreurs, imprécisions) varie significativement selon la catégorie ELO des joueurs, avec des taux plus faibles pour les joueurs de haut niveau.

Nous avons également vu que certaines **ouvertures** offrent des **probabilités de victoire** plus élevées pour les Blancs ou les Noirs en fonction des niveaux de joueurs et des types de parties.

Les modèles prédictifs, en particulier la régression multinomiale, nous ont permis d'atteindre une précision (accuracy) de 92 % dans la **prédiction des résultats** des parties. Bien que ce soit un excellent résultat, il est important de noter que certaines variables, comme la différence d'ELO ou les erreurs commises, montrent une influence limitée lorsqu'elles sont prises isolément. Les interactions complexes entre variables jouent un rôle crucial, et les parties d'échecs restent influencées par des dynamiques stratégiques difficiles à modéliser intégralement.

Ensuite, nous avons analysé la **probabilité de parties nulles** en fonction des ouvertures et des catégories ELO. Et enfin, la **corrélation entre la différence d'ELO et la durée des parties** s'est révélée insignifiante, invalidant l'hypothèse d'une relation directe.