# BigData | English Premier League Analysis

## Introducere

Folosind un set de date care acoperă meciuri care au avut loc între anii 1993 și ianuarie 2025, acest proiect își propune să studieze și să analizeze campionatul de fotbal English Premier League (EPL) din Anglia. Setul de date conține informații detaliate despre fiecare joc. Acestea includ rezultatele finale și finale, numărul de goluri marcate, performanța echipelor, informații despre arbitri (care includ faulturi, cornere, cartonașe galbene sau roșii) și alte statistici relevante. Un aspect interesant al proiectului este faptul că sezonul intermediar 2024/2025 este în curs de desfășurare. Acest lucru permite previzionarea clasamentului final pe baza datelor care sunt disponibile până în ianuarie 2025. Analiza va urmări performanța echipelor și randamentul arbitrilor, încercând să găsească favorizări sau pattern-uri interesante, precum și alte idei care pot fi extrase din analiza datelor.

### a. Prezentarea succintă a setului de date

Setul de date utilizat conține, pentru fiecare meci din EPL, următoarele informații principale:

- Data și sezonul meciului
- Echipa gazdă și echipa oaspete
- Numărul de goluri marcate de fiecare echipă (final și la pauză)
- Rezultatul final și la pauză
- Numele arbitrului
- Statistici detaliate: șuturi, șuturi pe poartă, faulturi, cornere, cartonașe galbene și roșii, etc.
- Cote de pariuri și alte metadate

Exemplu de structură a datelor (coloane):

```
MatchID,Season,MatchWeek,Date,Time,HomeTeam,AwayTeam,FullTimeHomeTeamGoals,FullTimeAwayTeamGoals,FullTimeResult,HalfTimeHomeTeamGoals,HalfTimeAwayTeamGoals,HalfTimeResult,Referee,HomeTeamShots,AwayTeamShots,HomeTeamShotsOnTarget,AwayTeamShotsOnTarget,HomeTeamCorners,AwayTeamCorners,HomeTeamFouls,AwayTeamFouls,HomeTeamYellowCards,AwayTeamYellowCards,HomeTeamRedCards,AwayTeamRedCards,B365HomeTeam,B365Draw,B365AwayTeam,B365Over2.5Goals,B365Under2.5Goals,MarketMaxHomeTeam,MarketMaxDraw,MarketMaxAwayTeam,MarketAvgHomeTeam,MarketAvgDraw,MarketAvgAwayTeam,MarketMaxOver2.5Goals,MarketMaxUnder2.5Goals,MarketAvgOver2.5Goals,MarketAvgUnder2.5Goals,HomeTeamPoints,AwayTeamPoints
```

Datele provin din surse publice și pot fi accesate la următorul link: Premier League Data

### b. Enunțarea obiectivelor

Obiectivele principale ale proiectului sunt:

1. Analiza performanței echipelor și a evoluției acestora de-a lungul sezoanelor EPL.
2. Studiul numărului de goluri marcate, identificarea tendințelor și a factorilor care influențează rezultatele.
3. Analiza randamentului arbitrilor: distribuția cartonașelor, faulturilor și identificarea eventualelor favorizări ale unor echipe de către anumiți arbitri.

Un alt scop principal al studiului este de a vedea si corelatia intre inceputul ciclului de victorii a echipei Manchester City si inceputul crizei a echipei Manchester United (ele fiind echipe din acelasi oras).

Proiectul va fi dezvoltat în notebook-ul `EPL_Analysis.ipynb`, folosind instrumente moderne de analiză a datelor și machine learning.

# 1. SPARK

## a. Inițializarea Spark și încărcarea datelor

În această etapă, inițializăm o sesiune Spark și încărcăm setul de date EPL într-un DataFrame Spark pentru a putea lucra cu volume mari de date.

```
from pyspark.sql import SparkSession

# Inițializăm sesiunea Spark
spark = SparkSession.builder.appName("EPL Analysis").getOrCreate()

# Încărcăm datele EPL într-un DataFrame Spark
df = spark.read.option("header", True).option(
    "inferSchema", True).csv("../data/PremierLeague_mid_season.csv")

# Afișăm primele 5 rânduri pentru a verifica încărcarea corectă a
datelor
df.show(5)
```

```
+--------------------+---------+---------+----------+----+-----------
+--------------+---------------------+---------------------
+--------------+---------------------+---------------------
+--------------+-------+-------------+-------------
+---------------------+---------------------+---------------
+---------------+-------------+-------------+-------------------
+-------------------+----------------+----------------+------------
+--------+------------+----------------+-----------------
+-----------------+-------------+-----------------+-----------------
+-------------+-----------------+---------------------
+---------------------+---------------------+----------------------
+--------------+--------------+
|             MatchID|   Season|MatchWeek|      Date|Time|   HomeTeam|
AwayTeam|FullTimeHomeTeamGoals|FullTimeAwayTeamGoals|FullTimeResult|
HalfTimeHomeTeamGoals|HalfTimeAwayTeamGoals|HalfTimeResult|Referee|
HomeTeamShots|AwayTeamShots|HomeTeamShotsOnTarget|
AwayTeamShotsOnTarget|HomeTeamCorners|AwayTeamCorners|HomeTeamFouls|
```

```
AwayTeamFouls|HomeTeamYellowCards|AwayTeamYellowCards|
HomeTeamRedCards|AwayTeamRedCards|B365HomeTeam|B365Draw|B365AwayTeam|
B365Over2.5Goals|B365Under2.5Goals|MarketMaxHomeTeam|MarketMaxDraw|
MarketMaxAwayTeam|MarketAvgHomeTeam|MarketAvgDraw|MarketAvgAwayTeam|
MarketMaxOver2.5Goals|MarketMaxUnder2.5Goals|MarketAvgOver2.5Goals|
MarketAvgUnder2.5Goals|HomeTeamPoints|AwayTeamPoints|
+--------------------+---------+---------+----------+----+-----------
+--------------+---------------------+---------------------
+--------------+---------------------+---------------------
+--------------+-------+-------------+-------------
+---------------------+---------------------+--------------
+--------------+-------------+-------------+------------------
+--------------------+----------------+----------------+------------
+--------+------------+----------------+-----------------
+-----------------+-------------+-----------------+-----------------
+-------------+-----------------+---------------------
+----------------------+---------------------+----------------------
+--------------+--------------+
|1993-1994_Arsenal...|1993-1994|        1|1993-08-14|NULL|    Arsenal|
Coventry|                    0|                    3|             A|
NULL|                  NULL|           NULL|   NULL|          NULL|
NULL|                  NULL|                   NULL|           NULL|
NULL|          NULL|          NULL|                NULL|
NULL|             NULL|             NULL|         NULL|    NULL|
NULL|             NULL|              NULL|             NULL|
NULL|              NULL|              NULL|          NULL|
NULL|                  NULL|                    NULL|
NULL|                   NULL|             0|             3|
|1993-1994_Aston V...|1993-1994|        1|1993-08-14|NULL|Aston Villa|
QPR|                     4|                     1|             H|
NULL|                  NULL|           NULL|   NULL|          NULL|
NULL|                  NULL|                   NULL|           NULL|
NULL|          NULL|          NULL|                NULL|
NULL|             NULL|             NULL|         NULL|    NULL|
NULL|             NULL|              NULL|             NULL|
NULL|              NULL|              NULL|          NULL|
NULL|                  NULL|                    NULL|
NULL|                   NULL|             3|             0|
|1993-1994_Chelsea...|1993-1994|        1|1993-08-14|NULL|    Chelsea|
Blackburn|                    1|                    2|             A|
NULL|                  NULL|           NULL|   NULL|          NULL|
NULL|                  NULL|                   NULL|           NULL|
NULL|          NULL|          NULL|                NULL|
NULL|             NULL|             NULL|         NULL|    NULL|
NULL|             NULL|              NULL|             NULL|
NULL|              NULL|              NULL|          NULL|
NULL|                  NULL|                    NULL|
NULL|                   NULL|             0|             3|
|1993-1994_Liverpo...|1993-1994|        1|1993-08-14|NULL|  Liverpool|
```

```
Sheffield Weds|                     2|                     0|
H|                  NULL|                  NULL|          NULL|   NULL|
NULL|          NULL|                   NULL|                    NULL|
NULL|            NULL|          NULL|          NULL|                 NULL|
NULL|             NULL|             NULL|         NULL|    NULL|
NULL|             NULL|              NULL|             NULL|
NULL|              NULL|              NULL|          NULL|
NULL|                  NULL|                   NULL|
NULL|                   NULL|             3|             0|
|1993-1994_Man Cit...|1993-1994|        1|1993-08-14|NULL|   Man City|
Leeds|                     1|                     1|             D|
NULL|                  NULL|           NULL|   NULL|          NULL|
NULL|                  NULL|                   NULL|            NULL|
NULL|          NULL|          NULL|                 NULL|
NULL|             NULL|             NULL|         NULL|    NULL|
NULL|             NULL|              NULL|             NULL|
NULL|              NULL|              NULL|          NULL|
NULL|                  NULL|                   NULL|
NULL|                   NULL|             1|             1|
+--------------------+---------+---------+----------+----+-----------
+--------------+--------------------+--------------------
+--------------+--------------------+--------------------
+--------------+-------+-------------+-------------
+---------------------+---------------------+--------------
+---------------+-------------+-------------+------------------
+-------------------+----------------+----------------+-----------
+--------+------------+----------------+-----------------
+-----------------+-------------+-----------------+-----------------
+-------------+-----------------+--------------------
+----------------------+---------------------+---------------------
+--------------+--------------+
only showing top 5 rows
```

## b. Explorarea structurii și a valorilor lipsă

În această secțiune, vom analiza structura setului de date și vom identifica valorile lipsă. Acest lucru ne ajută să înțelegem calitatea datelor și să decidem ce preprocesări sunt necesare înainte de analiză. Vom afișa schema DataFrame-ului și vom calcula numărul de valori lipsă pentru fiecare coloană.

```
# Afișăm schema DataFrame-ului pentru a vedea tipurile de date și coloanele disponibile
from pyspark.sql.functions import col, sum as spark_sum
df.printSchema()

# Redenumim coloanele care conțin caractere speciale (ex: punctul) cu underscore
for col_name in df.columns:
```

```
    if '.' in col_name:
        df = df.withColumnRenamed(col_name, col_name.replace('.',
'_'))

# Verificăm noile nume de coloane
print(df.columns)

# Calculăm numărul de valori lipsă (NULL) pentru fiecare coloană

missing_counts = df.select(
    [spark_sum(col(c).isNull().cast("int")).alias(c) for c in
df.columns])
missing_counts.show()
```

```
root
 |-- MatchID: string (nullable = true)
 |-- Season: string (nullable = true)
 |-- MatchWeek: integer (nullable = true)
 |-- Date: date (nullable = true)
 |-- Time: timestamp (nullable = true)
 |-- HomeTeam: string (nullable = true)
 |-- AwayTeam: string (nullable = true)
 |-- FullTimeHomeTeamGoals: integer (nullable = true)
 |-- FullTimeAwayTeamGoals: integer (nullable = true)
 |-- FullTimeResult: string (nullable = true)
 |-- HalfTimeHomeTeamGoals: integer (nullable = true)
 |-- HalfTimeAwayTeamGoals: integer (nullable = true)
 |-- HalfTimeResult: string (nullable = true)
 |-- Referee: string (nullable = true)
 |-- HomeTeamShots: integer (nullable = true)
 |-- AwayTeamShots: integer (nullable = true)
 |-- HomeTeamShotsOnTarget: integer (nullable = true)
 |-- AwayTeamShotsOnTarget: integer (nullable = true)
 |-- HomeTeamCorners: integer (nullable = true)
 |-- AwayTeamCorners: integer (nullable = true)
 |-- HomeTeamFouls: integer (nullable = true)
 |-- AwayTeamFouls: integer (nullable = true)
 |-- HomeTeamYellowCards: integer (nullable = true)
 |-- AwayTeamYellowCards: integer (nullable = true)
 |-- HomeTeamRedCards: integer (nullable = true)
 |-- AwayTeamRedCards: integer (nullable = true)
 |-- B365HomeTeam: double (nullable = true)
 |-- B365Draw: double (nullable = true)
 |-- B365AwayTeam: double (nullable = true)
 |-- B365Over2.5Goals: double (nullable = true)
 |-- B365Under2.5Goals: double (nullable = true)
 |-- MarketMaxHomeTeam: double (nullable = true)
 |-- MarketMaxDraw: double (nullable = true)
 |-- MarketMaxAwayTeam: double (nullable = true)
 |-- MarketAvgHomeTeam: double (nullable = true)
```

```
 |-- MarketAvgDraw: double (nullable = true)
 |-- MarketAvgAwayTeam: double (nullable = true)
 |-- MarketMaxOver2.5Goals: double (nullable = true)
 |-- MarketMaxUnder2.5Goals: double (nullable = true)
 |-- MarketAvgOver2.5Goals: double (nullable = true)
 |-- MarketAvgUnder2.5Goals: double (nullable = true)
 |-- HomeTeamPoints: integer (nullable = true)
 |-- AwayTeamPoints: integer (nullable = true)

['MatchID', 'Season', 'MatchWeek', 'Date', 'Time', 'HomeTeam',
'AwayTeam', 'FullTimeHomeTeamGoals', 'FullTimeAwayTeamGoals',
'FullTimeResult', 'HalfTimeHomeTeamGoals', 'HalfTimeAwayTeamGoals',
'HalfTimeResult', 'Referee', 'HomeTeamShots', 'AwayTeamShots',
'HomeTeamShotsOnTarget', 'AwayTeamShotsOnTarget', 'HomeTeamCorners',
'AwayTeamCorners', 'HomeTeamFouls', 'AwayTeamFouls',
'HomeTeamYellowCards', 'AwayTeamYellowCards', 'HomeTeamRedCards',
'AwayTeamRedCards', 'B365HomeTeam', 'B365Draw', 'B365AwayTeam',
'B365Over2_5Goals', 'B365Under2_5Goals', 'MarketMaxHomeTeam',
'MarketMaxDraw', 'MarketMaxAwayTeam', 'MarketAvgHomeTeam',
'MarketAvgDraw', 'MarketAvgAwayTeam', 'MarketMaxOver2_5Goals',
'MarketMaxUnder2_5Goals', 'MarketAvgOver2_5Goals',
'MarketAvgUnder2_5Goals', 'HomeTeamPoints', 'AwayTeamPoints']
+-------+------+---------+----+----+--------+--------
+---------------------+---------------------+--------------
+---------------------+---------------------+--------------+-------
+-------------+-------------+--------------------
+---------------------+---------------+---------------+-------------
+-------------+-------------------+-------------------
+----------------+----------------+------------+--------+------------
+----------------+-----------------+-----------------+-------------
+-----------------+-----------------+-------------+-----------------
+---------------------+----------------------+--------------------
+----------------------+--------------+--------------+
|MatchID|Season|MatchWeek|Date|Time|HomeTeam|AwayTeam|
FullTimeHomeTeamGoals|FullTimeAwayTeamGoals|FullTimeResult|
HalfTimeHomeTeamGoals|HalfTimeAwayTeamGoals|HalfTimeResult|Referee|
HomeTeamShots|AwayTeamShots|HomeTeamShotsOnTarget|
AwayTeamShotsOnTarget|HomeTeamCorners|AwayTeamCorners|HomeTeamFouls|
AwayTeamFouls|HomeTeamYellowCards|AwayTeamYellowCards|
HomeTeamRedCards|AwayTeamRedCards|B365HomeTeam|B365Draw|B365AwayTeam|
B365Over2_5Goals|B365Under2_5Goals|MarketMaxHomeTeam|MarketMaxDraw|
MarketMaxAwayTeam|MarketAvgHomeTeam|MarketAvgDraw|MarketAvgAwayTeam|
MarketMaxOver2_5Goals|MarketMaxUnder2_5Goals|MarketAvgOver2_5Goals|
MarketAvgUnder2_5Goals|HomeTeamPoints|AwayTeamPoints|
+-------+------+---------+----+----+--------+--------
+---------------------+---------------------+--------------
+---------------------+---------------------+--------------+-------
+-------------+-------------+--------------------
+---------------------+---------------+---------------+-------------
```

```
+-------------+-------------------+-------------------
+----------------+----------------+------------+--------+------------
+----------------+-----------------+-----------------+-------------
+-----------------+-----------------+-------------+-----------------
+----------------------+----------------------+---------------------
+----------------------+--------------+--------------+
|      0|     0|        0|   0|9880|       0|       0|
0|                    0|             0|                  760|
760|           760|   2660|         2660|         2660|
2660|                  2660|           2660|           2660|
2660|         2660|               2660|              2660|
2660|            2660|        3420|    3420|        3420|
8877|             8877|             9880|         9880|
9880|             9880|          9880|             9880|
9880|                  9880|                  9880|
9880|              0|              0|
+-------+------+---------+----+----+--------+--------
+----------------------+---------------------+--------------
+----------------------+---------------------+--------------+-------
+-------------+-------------+--------------------
+----------------------+---------------+---------------+-------------
+-------------+-------------------+-------------------
+----------------+----------------+------------+--------+------------
+----------------+-----------------+-----------------+-------------
+-----------------+-----------------+-------------+-----------------
+----------------------+----------------------+---------------------
+----------------------+--------------+--------------+
```

## c. Separarea datelor pentru analize generale vs. analize detaliate

Pentru a asigura acuratețea analizelor, vom separa datele în două subseturi:

- Un subset pentru statistici generale (clasamente, goluri, puncte), unde folosim toate meciurile disponibile, chiar dacă unele coloane au valori lipsă.
- Un subset pentru analize detaliate (șuturi, cornere, arbitri etc.), unde păstrăm doar meciurile din sezonul 2000-2001 încolo și doar acele rânduri care au date complete pe coloanele relevante.

```
from pyspark.sql.functions import col

# Definim lista cu numele coloanelor de eliminat
cols_to_drop = [
    "Time",
    "B365HomeTeam", "B365Draw", "B365AwayTeam", "B365Over2_5Goals",
"B365Under2_5Goals",
    "MarketMaxHomeTeam", "MarketMaxDraw", "MarketMaxAwayTeam",
    "MarketAvgHomeTeam", "MarketAvgDraw", "MarketAvgAwayTeam",
```

```
    "MarketMaxOver2_5Goals", "MarketMaxUnder2_5Goals",
    "MarketAvgOver2_5Goals", "MarketAvgUnder2_5Goals"
]

# Eliminăm coloanele doar dacă există în DataFrame
for col_name in cols_to_drop:
    if col_name in df.columns:
        df = df.drop(col_name)

general_cols = [
    "MatchID", "Season", "MatchWeek", "Date", "HomeTeam", "AwayTeam",
    "FullTimeHomeTeamGoals", "FullTimeAwayTeamGoals",
"FullTimeResult",
    "HomeTeamPoints", "AwayTeamPoints"
]

# 1. Subset pentru statistici generale (toate meciurile cu scor și
echipe)
df_general = df.select(general_cols).dropna()

# 2. Subset pentru analize detaliate (doar meciuri cu date complete,
din 2000-2001 încolo)
relevant_cols = [
    "HomeTeam", "AwayTeam", "FullTimeHomeTeamGoals",
"FullTimeAwayTeamGoals", "Date",
    "HalfTimeHomeTeamGoals", "HalfTimeAwayTeamGoals",
"HalfTimeResult", "Referee",
    "HomeTeamShots", "AwayTeamShots", "HomeTeamShotsOnTarget",
"AwayTeamShotsOnTarget",
    "HomeTeamCorners", "AwayTeamCorners", "HomeTeamFouls",
"AwayTeamFouls",
    "HomeTeamYellowCards", "AwayTeamYellowCards", "HomeTeamRedCards",
"AwayTeamRedCards"
]
df_detailed = (
    df
    .filter(col("Season") >= "2000-2001")
    .dropna(subset=relevant_cols)
)

# Verificăm câte meciuri avem în fiecare subset
print("Meciuri pentru statistici generale:", df_general.count())
print("Meciuri pentru analize detaliate:", df_detailed.count())
print("---------------------------------")

# Aratam primele 20 de rânduri din fiecare subset
df_general.show(20)
df_detailed.show(20)
```

```
Meciuri pentru statistici generale: 12001
Meciuri pentru analize detaliate: 9341
--------------------------------
+--------------------+---------+---------+----------+---------------
+----------------+--------------------+--------------------
+--------------+--------------+--------------+
|             MatchID|   Season|MatchWeek|      Date|        HomeTeam|
AwayTeam|FullTimeHomeTeamGoals|FullTimeAwayTeamGoals|FullTimeResult|
HomeTeamPoints|AwayTeamPoints|
+--------------------+---------+---------+----------+---------------
+----------------+--------------------+--------------------
+--------------+--------------+--------------+
|1993-1994_Arsenal...|1993-1994|        1|1993-08-14|         Arsenal|
Coventry|                   0|                   3|             A|
0|             3|
|1993-1994_Aston V...|1993-1994|        1|1993-08-14|     Aston Villa|
QPR|                   4|                   1|             H|
3|             0|
|1993-1994_Chelsea...|1993-1994|        1|1993-08-14|         Chelsea|
Blackburn|                   1|                   2|             A|
0|             3|
|1993-1994_Liverpo...|1993-1994|        1|1993-08-14|       Liverpool|
Sheffield Weds|                   2|                   0|
H|             3|             0|
|1993-1994_Man Cit...|1993-1994|        1|1993-08-14|        Man City|
Leeds|                   1|                   1|             D|
1|             1|
|1993-1994_Newcast...|1993-1994|        1|1993-08-14|       Newcastle|
Tottenham|                   0|                   1|             A|
0|             3|
|1993-1994_Oldham_...|1993-1994|        1|1993-08-14|          Oldham|
Ipswich|                   0|                   3|             A|
0|             3|
|1993-1994_Sheffie...|1993-1994|        1|1993-08-14|Sheffield United|
Swindon|                   3|                   1|             H|
3|             0|
|1993-1994_Southam...|1993-1994|        1|1993-08-14|     Southampton|
Everton|                   0|                   2|             A|
0|             3|
|1993-1994_West Ha...|1993-1994|        1|1993-08-14|        West Ham|
Wimbledon|                   0|                   2|             A|
0|             3|
|1993-1994_Norwich...|1993-1994|        1|1993-08-15|         Norwich|
Man United|                   0|                   2|             A|
0|             3|
|1993-1994_Tottenh...|1993-1994|        1|1993-08-16|       Tottenham|
Arsenal|                   0|                   1|             A|
0|             3|
|1993-1994_Everton...|1993-1994|        1|1993-08-17|         Everton|
Man City|                   1|                   0|             H|
```

```
3|             0|
|1993-1994_Ipswich...|1993-1994|        1|1993-08-17|         Ipswich|
Southampton|                     1|                     0|
H|             3|              0|
|1993-1994_Leeds_W...|1993-1994|        1|1993-08-17|           Leeds|
West Ham|                     1|                     0|             H|
3|             0|
|1993-1994_Wimbled...|1993-1994|        1|1993-08-17|       Wimbledon|
Chelsea|                     1|                     1|             D|
1|             1|
|1993-1994_Blackbu...|1993-1994|        1|1993-08-18|       Blackburn|
Norwich|                     2|                     3|             A|
0|             3|
|1993-1994_Coventr...|1993-1994|        1|1993-08-18|        Coventry|
Newcastle|                     2|                     1|             H|
3|             0|
|1993-1994_Man Uni...|1993-1994|        1|1993-08-18|      Man United|
Sheffield United|                     3|                     0|
H|             3|              0|
|1993-1994_QPR_Liv...|1993-1994|        1|1993-08-18|             QPR|
Liverpool|                     1|                     3|             A|
0|             3|
+--------------------+---------+---------+----------+----------------
+----------------+---------------------+--------------------
+--------------+--------------+--------------+
only showing top 20 rows

+--------------------+---------+---------+----------+-------------
+-------------+---------------------+--------------------
+--------------+---------------------+---------------------
+--------------+----------------+-------------+-------------
+---------------------+---------------------+--------------
+---------------+-------------+-------------+-------------------
+-------------------+----------------+----------------+--------------
+--------------+
|              MatchID|   Season|MatchWeek|      Date|     HomeTeam|
AwayTeam|FullTimeHomeTeamGoals|FullTimeAwayTeamGoals|FullTimeResult|
HalfTimeHomeTeamGoals|HalfTimeAwayTeamGoals|HalfTimeResult|
Referee|HomeTeamShots|AwayTeamShots|HomeTeamShotsOnTarget|
AwayTeamShotsOnTarget|HomeTeamCorners|AwayTeamCorners|HomeTeamFouls|
AwayTeamFouls|HomeTeamYellowCards|AwayTeamYellowCards|
HomeTeamRedCards|AwayTeamRedCards|HomeTeamPoints|AwayTeamPoints|
+--------------------+---------+---------+----------+-------------
+-------------+---------------------+--------------------
+--------------+---------------------+---------------------
+--------------+----------------+-------------+-------------
+---------------------+---------------------+--------------
+---------------+-------------+-------------+-------------------
+-------------------+----------------+----------------+--------------
+--------------+
```

```
|2000-2001_Charlto...|2000-2001|        1|2000-08-19|     Charlton|
Man City|                    4|                    0|             H|
2|                     0|             H|      Rob Harris|           17|
8|                    14|                    4|              6|
6|           13|           12|                   1|                   2|
0|               0|             3|              0|
|2000-2001_Chelsea...|2000-2001|        1|2000-08-19|      Chelsea|
West Ham|                    4|                    2|             H|
1|                     0|             H|   Graham Barber|           17|
12|                    10|                    5|              7|
7|           19|           14|                   1|                   2|
0|               0|             3|              0|
|2000-2001_Coventr...|2000-2001|        1|2000-08-19|     Coventry|
Middlesbrough|                    1|                    3|
A|                     1|                    1|             D|    Barry
Knight|            6|           16|                    3|
9|              8|              4|           15|           21|
5|                  3|                1|               0|
0|             3|
|2000-2001_Derby_S...|2000-2001|        1|2000-08-19|        Derby|
Southampton|                    2|                    2|
D|                     1|                    2|             A|     Andy
D'Urso|            6|           13|                    4|
6|              5|              8|           11|           13|
1|                  1|                0|               0|
1|             1|
|2000-2001_Leeds_E...|2000-2001|        1|2000-08-19|        Leeds|
Everton|                    2|                    0|            H|
2|                     0|             H|Dermot Gallagher|           17|
12|                    8|                    6|              6|
4|           21|           20|                   1|                   3|
0|               0|             3|              0|
|2000-2001_Leicest...|2000-2001|        1|2000-08-19|    Leicester|
Aston Villa|                    0|                    0|
D|                     0|                    0|             D|
Mike Riley|             5|            5|                    4|
3|              5|              4|           12|           12|
2|                  3|                0|               0|
1|             1|
|2000-2001_Liverpo...|2000-2001|        1|2000-08-19|    Liverpool|
Bradford|                    1|                    0|             H|
0|                     0|             D|     Paul Durkin|           16|
3|                    10|                    2|              6|
1|             8|             8|                   1|                   1|
0|               0|             3|              0|
|2000-2001_Sunderl...|2000-2001|        1|2000-08-19|   Sunderland|
Arsenal|                    1|                    0|             H|
0|                     0|             D|      Steve Dunn|            8|
14|                    2|                    7|              2|
9|           10|           21|                   3|                   1|
```

```
0|               1|             3|             0|
|2000-2001_Tottenh...|2000-2001|        1|2000-08-19|    Tottenham|
Ipswich|                    3|                  1|             H|
2|                   1|             H|      Alan Wiley|           20|
15|                   6|                  5|              3|
4|          14|           13|                 0|                  0|
0|              0|             3|            0|
|2000-2001_Man Uni...|2000-2001|        1|2000-08-20|   Man United|
Newcastle|                    2|                  0|             H|
1|                   0|             H|     Steve Lodge|           19|
9|                   9|                  6|              7|
1|            7|           13|                 0|                  1|
0|              0|             3|            0|
|2000-2001_Arsenal...|2000-2001|        1|2000-08-21|      Arsenal|
Liverpool|                    2|                  0|             H|
1|                   0|             H|     Graham Poll|           17|
7|                  12|                  4|             10|
11|           25|           20|                2|
4|               1|               2|             3|             0|
|2000-2001_Bradfor...|2000-2001|        1|2000-08-22|     Bradford|
Chelsea|                    2|                  0|             H|
1|                   0|             H|     Mark Halsey|           12|
14|                   3|                  6|              6|
4|          14|           16|                 0|                  1|
0|              0|             3|            0|
|2000-2001_Ipswich...|2000-2001|        1|2000-08-22|      Ipswich|
Man United|                    1|                  1|             D|
1|                   1|             D|     Jeff Winter|           13|
15|                   8|                  6|              4|
6|          10|            7|                 1|                  4|
0|              0|             1|            1|
|2000-2001_Middles...|2000-2001|        1|2000-08-22|Middlesbrough|
Tottenham|                    1|                  1|             D|
0|                   1|             A|     Peter Jones|           12|
11|                   6|                  4|              5|
5|            9|           18|                 2|                  1|
0|              0|             1|            1|
|2000-2001_Everton...|2000-2001|        1|2000-08-23|      Everton|
Charlton|                    3|                  0|             H|
0|                   0|             D|       Andy Hall|           13|
8|                   8|                  4|              3|
5|          17|           15|                 2|                  1|
0|              1|             3|            0|
|2000-2001_Man Cit...|2000-2001|        1|2000-08-23|     Man City|
Sunderland|                    4|                  2|             H|
2|                   0|             H|   David Ellaray|           15|
9|                  10|                  4|              7|
3|          24|           14|                 3|                  3|
0|              0|             3|            0|
|2000-2001_Newcast...|2000-2001|        1|2000-08-23|    Newcastle|
```

```
Derby|                    3|                    2|             H|
1|                   1|             D|Dermot Gallagher|            9|
10|                   4|                     5|              9|
6|           23|           11|                  0|                  3|
1|               0|             3|             0|
|2000-2001_Southam...|2000-2001|        1|2000-08-23|  Southampton|
Coventry|                    1|                    2|             A|
0|                   1|             A|        F Taylor|           12|
7|                   4|                     5|              6|
5|           18|           20|                  5|                  3|
0|               0|             0|             3|
|2000-2001_West Ha...|2000-2001|        1|2000-08-23|     West Ham|
Leicester|                    0|                    1|             A|
0|                   0|             D|      Rob Styles|           17|
4|                  12|                     2|              11|
5|           16|           14|                  3|                  3|
1|               0|             0|             3|
|2000-2001_Arsenal...|2000-2001|        2|2000-08-26|      Arsenal|
Charlton|                    5|                    3|             H|
1|                   2|             A|     Steve Lodge|           18|
7|                   9|                     4|              8|
3|           12|           15|                  0|                  1|
0|               0|             3|             0|
+--------------------+---------+---------+----------+-------------
+-------------+--------------------+--------------------
+--------------+--------------------+--------------------
+--------------+----------------+-------------+-------------
+---------------------+---------------------+---------------
+---------------+-------------+-------------+------------------
+-------------------+----------------+----------------+-------------
+--------------+
only showing top 20 rows
```

# d. Playground cu datele

## I. Clasament general - Cele mai multe puncte all time din EPL, 1990-2025

În această secțiune, am realizat o analiză detaliată a performanței echipelor din English Premier League (EPL) pe baza datelor istorice disponibile, acoperind perioada 1990–2025. Analiza a fost efectuată utilizând Apache Spark, ceea ce a permis procesarea eficientă a unui volum mare de date și realizarea de agregări complexe.

- a. Clasament pe fiecare sezon -- Pentru fiecare sezon din EPL, am calculat punctele obținute de fiecare echipă, atât în meciurile disputate pe teren propriu, cât și în deplasare. Punctajul total al fiecărei echipe într-un sezon a fost determinat prin însumarea punctelor obținute acasă și în deplasare. Ulterior, echipele au fost

ordonate descrescător în funcție de punctajul total, obținând astfel clasamentul final pentru fiecare sezon. Pentru o vizualizare sintetică, am extras și afișat primele trei echipe din fiecare sezon, evidențiind astfel echipele de top ale fiecărui an competițional.

- a. Determinarea câștigătoarei fiecărui sezon -- Pe baza clasamentului sezonier, am identificat câștigătoarea fiecărui sezon, adică echipa care a acumulat cele mai multe puncte la finalul sezonului respectiv. Acest demers permite o analiză longitudinală a performanței echipelor și evidențiază cluburile dominante de-a lungul istoriei EPL.

- a. Clasamentul general all-time -- Am calculat și clasamentul general all-time, adunând punctele obținute de fiecare echipă în toate sezoanele analizate, atât acasă, cât și în deplasare. Acest clasament oferă o perspectivă globală asupra performanței echipelor pe termen lung, permițând identificarea celor mai de succes cluburi din istoria Premier League.

- a. Numărul de campionate câștigate de fiecare echipă -- În final, am agregat numărul de titluri de campioană obținute de fiecare echipă, pe baza rezultatelor anuale. Acest indicator reflectă consistența și succesul echipelor de-a lungul timpului, oferind o imagine clară asupra palmaresului fiecărui club.

- a. Numărul de meciuri jucate de fiecare echipă -- Am calculat numărul total de meciuri jucate de fiecare echipă în toate sezoanele analizate, atât acasă, cât și în deplasare. Acest indicator oferă o perspectivă asupra performanței echipelor pe termen lung, permițând identificarea celor mai consistente cluburi din istoria Premier League.

```
from pyspark.sql import Window
from pyspark.sql.functions import row_number, col, coalesce, lit,
count, sum as spark_sum

# =========================
# 1. Clasament pe fiecare sezon
# =========================

print("\n=== Clasament pe fiecare sezon: Top 3 echipe/sezon ===")

# Calculăm punctele pe sezon pentru fiecare echipă (acasă + deplasare)
home_points_season = df_general.groupBy("Season", "HomeTeam").agg(
    spark_sum("HomeTeamPoints").alias("TotalHomePoints"))
away_points_season = df_general.groupBy("Season", "AwayTeam").agg(
    spark_sum("AwayTeamPoints").alias("TotalAwayPoints"))

# Redenumim pentru join
home_points_season = home_points_season.withColumnRenamed("HomeTeam",
"Team")
away_points_season = away_points_season.withColumnRenamed("AwayTeam",
"Team")
```

```
# Join și total
standings_season = home_points_season.join(
    away_points_season, on=["Season", "Team"], how="outer"
).withColumn(
    "TotalHomePoints", coalesce(home_points_season["TotalHomePoints"],
lit(0))
).withColumn(
    "TotalAwayPoints", coalesce(away_points_season["TotalAwayPoints"],
lit(0))
).withColumn(
    "TotalPoints", col("TotalHomePoints") + col("TotalAwayPoints")
)

# Pentru fiecare sezon, sortăm descrescător după puncte
window =
Window.partitionBy("Season").orderBy(col("TotalPoints").desc())
standings_season = standings_season.withColumn(
    "Rank", row_number().over(window)
)

# Afișăm top 3 echipe din fiecare sezon
standings_season.filter(col("Rank") <= 3).orderBy(
    "Season", "Rank").show(20, truncate=False)

# =========================
# 2. Câștigătoarea fiecărui sezon
# =========================

print("\n=== Câștigătoarea fiecărui sezon ===")
winners = standings_season.filter(col("Rank") == 1).select(
    "Season", "Team", "TotalPoints")
winners.show(truncate=False)

# =========================
# 3. Clasamentul general all-time (toate punctele adunate)
# =========================

print("\n=== Clasament general all-time (toate punctele adunate) ===")
home_points = df_general.groupBy("HomeTeam").agg(
    spark_sum("HomeTeamPoints").alias("TotalHomePoints"))
away_points = df_general.groupBy("AwayTeam").agg(
    spark_sum("AwayTeamPoints").alias("TotalAwayPoints"))

home_points = home_points.withColumnRenamed("HomeTeam", "Team")
away_points = away_points.withColumnRenamed("AwayTeam", "Team")

standings_alltime = home_points.join(away_points, on="Team",
how="outer") \
    .withColumn("TotalHomePoints",
```

```
coalesce(home_points["TotalHomePoints"], lit(0))) \
    .withColumn("TotalAwayPoints",
coalesce(away_points["TotalAwayPoints"], lit(0))) \
    .withColumn("TotalPoints", col("TotalHomePoints") +
col("TotalAwayPoints"))

standings_alltime =
standings_alltime.orderBy(col("TotalPoints").desc())
standings_alltime.show(20, truncate=False)

# ========================
# 4. Număr campionate câștigate de fiecare echipă
# ========================

print("\n=== Număr campionate câștigate de fiecare echipă ===")
winners_counter = winners.groupBy("Team").agg(count("Season").alias(
    "Championships")).orderBy(col("Championships").desc())
winners_counter.show(truncate=False)

# ========================
# 5. Numar total de meciuri jucate de fiecare echipa
# ========================

print("\n=== Număr total de meciuri jucate de fiecare echipă ===")

# Numărăm numărul total de meciuri jucate de fiecare echipă ordonat
descrescator dupa numarul de meciuri
total_matches_per_team = df_general.groupBy("HomeTeam").agg(
    count("HomeTeam").alias("TotalMatches")
).withColumnRenamed("HomeTeam",
"Team").orderBy(col("TotalMatches").desc())

# Afișăm rezultatele
total_matches_per_team.show(100, truncate=False)


=== Clasament pe fiecare sezon: Top 3 echipe/sezon ===
+---------+----------+---------------+---------------+-----------+----
+
|Season   |Team      |TotalHomePoints|TotalAwayPoints|TotalPoints|
Rank|
+---------+----------+---------------+---------------+-----------+----
+
|1993-1994|Man United|38             |38             |76         |1
|
|1993-1994|Blackburn |42             |31             |73         |2
|
|1993-1994|Newcastle |36             |28             |64         |3
|
|1994-1995|Blackburn |47             |32             |79         |1
```

```
|
|1994-1995|Man United|45              |29              |74          |2
|
|1994-1995|Newcastle |41              |23              |64          |3
|
|1995-1996|Man United|49              |33              |82          |1
|
|1995-1996|Newcastle |52              |26              |78          |2
|
|1995-1996|Liverpool |46              |25              |71          |3
|
|1996-1997|Man United|41              |34              |75          |1
|
|1996-1997|Arsenal   |35              |33              |68          |2
|
|1996-1997|Liverpool |36              |32              |68          |3
|
|1997-1998|Arsenal   |47              |31              |78          |1
|
|1997-1998|Man United|43              |34              |77          |2
|
|1997-1998|Liverpool |41              |24              |65          |3
|
|1998-1999|Man United|46              |33              |79          |1
|
|1998-1999|Arsenal   |47              |31              |78          |2
|
|1998-1999|Chelsea   |42              |33              |75          |3
|
|1999-2000|Man United|49              |42              |91          |1
|
|1999-2000|Arsenal   |45              |28              |73          |2
|
+---------+----------+----------------+----------------+------------+----
+
only showing top 20 rows


=== Câștigătoarea fiecărui sezon ===
+---------+----------+-----------+
|Season   |Team      |TotalPoints|
+---------+----------+-----------+
|1993-1994|Man United|76         |
|1994-1995|Blackburn |79         |
|1995-1996|Man United|82         |
|1996-1997|Man United|75         |
|1997-1998|Arsenal   |78         |
|1998-1999|Man United|79         |
|1999-2000|Man United|91         |
```

```
|2000-2001|Man United|80         |
|2001-2002|Arsenal   |87         |
|2002-2003|Man United|83         |
|2003-2004|Arsenal   |90         |
|2004-2005|Chelsea   |95         |
|2005-2006|Chelsea   |91         |
|2006-2007|Man United|89         |
|2007-2008|Man United|87         |
|2008-2009|Man United|90         |
|2009-2010|Chelsea   |86         |
|2010-2011|Man United|80         |
|2011-2012|Man City  |89         |
|2012-2013|Man United|89         |
+---------+----------+-----------+
only showing top 20 rows


=== Clasament general all-time (toate punctele adunate) ===
+--------------+---------------+---------------+-----------+
|Team          |TotalHomePoints|TotalAwayPoints|TotalPoints|
+--------------+---------------+---------------+-----------+
|Man United    |1349           |1064           |2413       |
|Arsenal       |1313           |972            |2285       |
|Liverpool     |1286           |940            |2226       |
|Chelsea       |1239           |970            |2209       |
|Tottenham     |1107           |752            |1859       |
|Man City      |1018           |755            |1773       |
|Everton       |980            |627            |1607       |
|Newcastle     |969            |589            |1558       |
|Aston Villa   |823            |613            |1436       |
|West Ham      |828            |521            |1349       |
|Southampton   |623            |396            |1019       |
|Blackburn     |538            |340            |878        |
|Leicester     |480            |347            |827        |
|Fulham        |498            |274            |772        |
|Leeds         |419            |316            |735        |
|Crystal Palace|330            |298            |628        |
|Middlesbrough |385            |235            |620        |
|Sunderland    |381            |237            |618        |
|Bolton        |354            |221            |575        |
|West Brom     |294            |196            |490        |
+--------------+---------------+---------------+-----------+
only showing top 20 rows


=== Număr campionate câștigate de fiecare echipă ===
+----------+-------------+
|Team      |Championships|
+----------+-------------+
|Man United|12           |
```

```
|Man City  |8           |
|Chelsea   |5           |
|Arsenal   |3           |
|Liverpool |2           |
|Blackburn |1           |
|Leicester |1           |
+----------+------------+


=== Număr total de meciuri jucate de fiecare echipă ===
+----------------+------------+
|Team            |TotalMatches|
+----------------+------------+
|Man United      |598         |
|Tottenham       |597         |
|Arsenal         |597         |
|Chelsea         |597         |
|Everton         |597         |
|Liverpool       |595         |
|Newcastle       |558         |
|Aston Villa     |541         |
|West Ham        |538         |
|Man City        |502         |
|Southampton     |444         |
|Fulham          |334         |
|Leicester       |333         |
|Blackburn       |321         |
|Sunderland      |304         |
|Crystal Palace  |275         |
|Middlesbrough   |266         |
|Leeds           |261         |
|Bolton          |247         |
|West Brom       |247         |
|Wolves          |200         |
|Stoke           |190         |
|Burnley         |171         |
|Norwich         |168         |
|Charlton        |152         |
|Watford         |152         |
|Wigan           |152         |
|Coventry        |150         |
|Brighton        |143         |
|Bournemouth     |143         |
|Birmingham      |133         |
|Derby           |133         |
|Swansea         |133         |
|Portsmouth      |133         |
|Sheffield Weds  |130         |
|Wimbledon       |129         |
```

```
|Nott'm Forest   |124         |
|QPR             |108         |
|Hull            |95          |
|Sheffield United|94          |
|Ipswich         |85          |
|Brentford       |69          |
|Reading         |57          |
|Bradford        |38          |
|Cardiff         |38          |
|Huddersfield    |38          |
|Barnsley        |19          |
|Luton           |19          |
|Blackpool       |19          |
|Swindon         |18          |
|Oldham          |16          |
+----------------+------------+
```

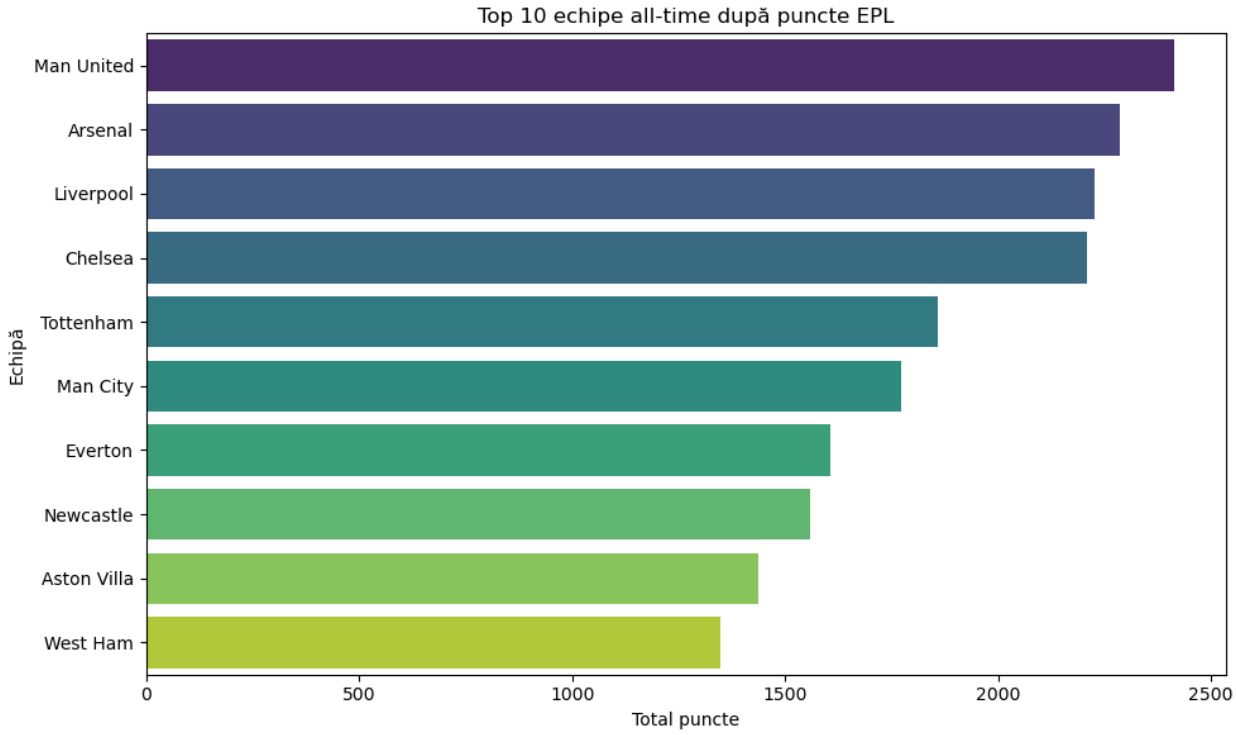
Voi crea acum niste vizualizari pentru a vedea cum se comporta echipele in timp.

### 1. Bar chart: Top 10 echipe all-time după puncte

Vizualizarea de mai jos prezintă, sub formă de bar chart, primele 10 echipe din istoria Premier League în funcție de totalul punctelor acumulate all-time. Graficul evidențiază cluburile cu cele mai bune performanțe pe termen lung, oferind o perspectivă rapidă asupra echipelor dominante din EPL.

Putem observa ca echipa de pe locul 1 este Manchester United, care a castigat si 12 titluri de campioana. Am putea spune ca este si "normal" ca Manchester United sa fie pe primul loc, deoarece a fost una dintre cele mai "bune" echipe din EPL in ultimii 30 de ani; Am putea studia echipa asta si Manchester City, oarecum inceperea dominiului Manchester City este concomitenta cu inceperea caderii Manchester United.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Convertim standings_alltime în Pandas pentru vizualizare
top10_alltime = standings_alltime.limit(10).toPandas()

plt.figure(figsize=(10, 6))
sns.barplot(data=top10_alltime, x="TotalPoints", y="Team",
            hue="Team", palette="viridis", dodge=False, legend=False)
plt.title("Top 10 echipe all-time după puncte EPL")
plt.xlabel("Total puncte")
plt.ylabel("Echipă")
plt.tight_layout()
plt.show()
```



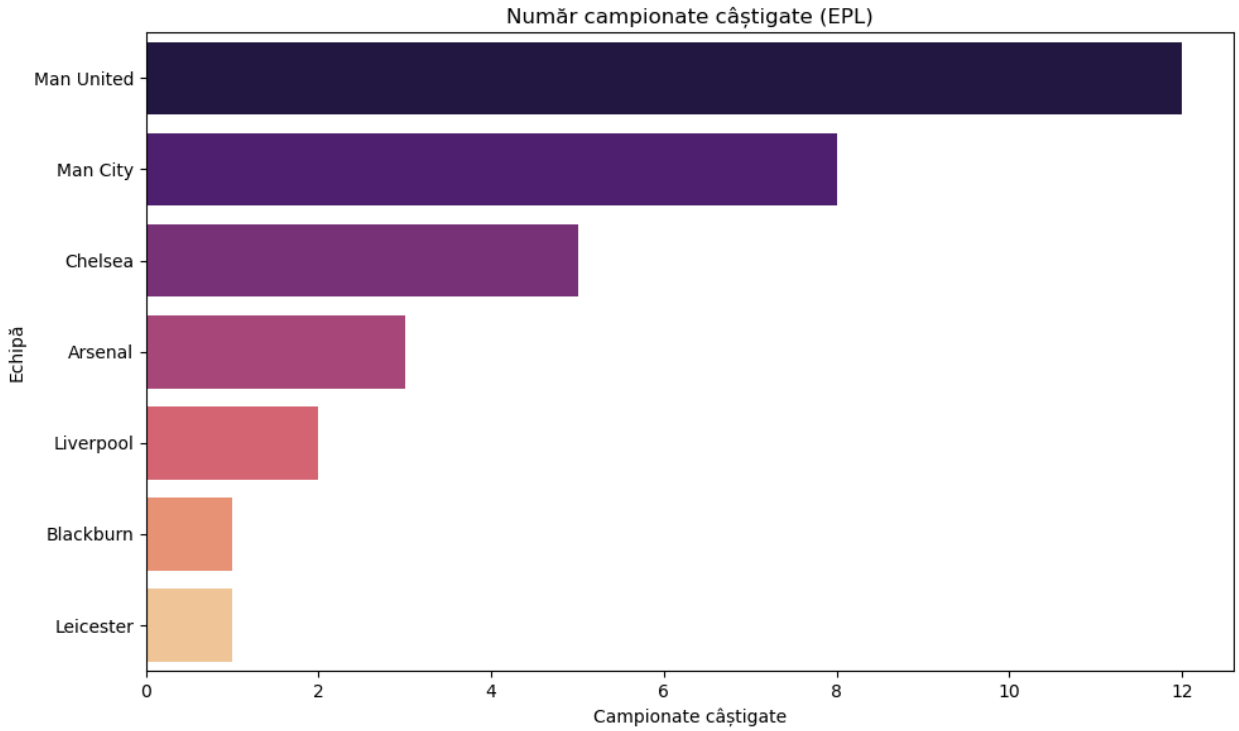

## 2. Bar chart: Top echipe după numărul de campionate câștigate

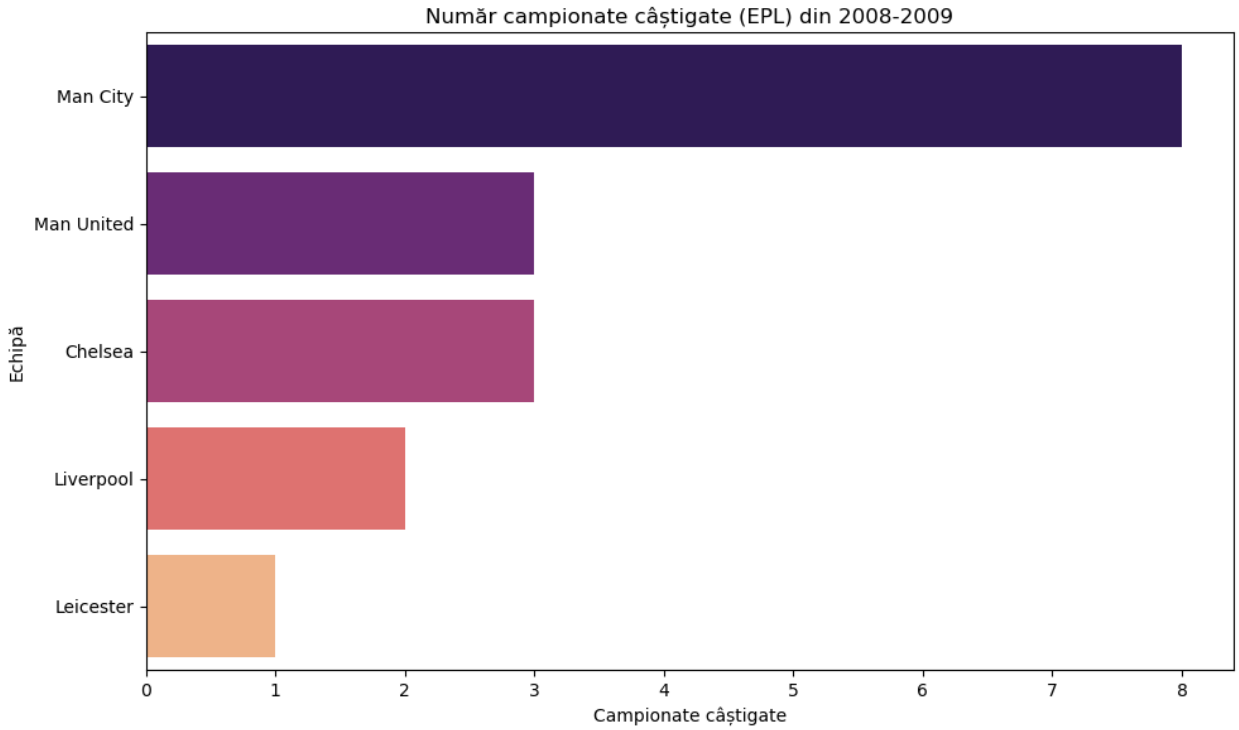
Graficul de mai jos ilustrează numărul de campionate câștigate de fiecare echipă în Premier League. Această vizualizare evidențiază cluburile cu cele mai multe titluri de campioană, oferind o imagine clară asupra palmaresului echipelor de top din EPL.

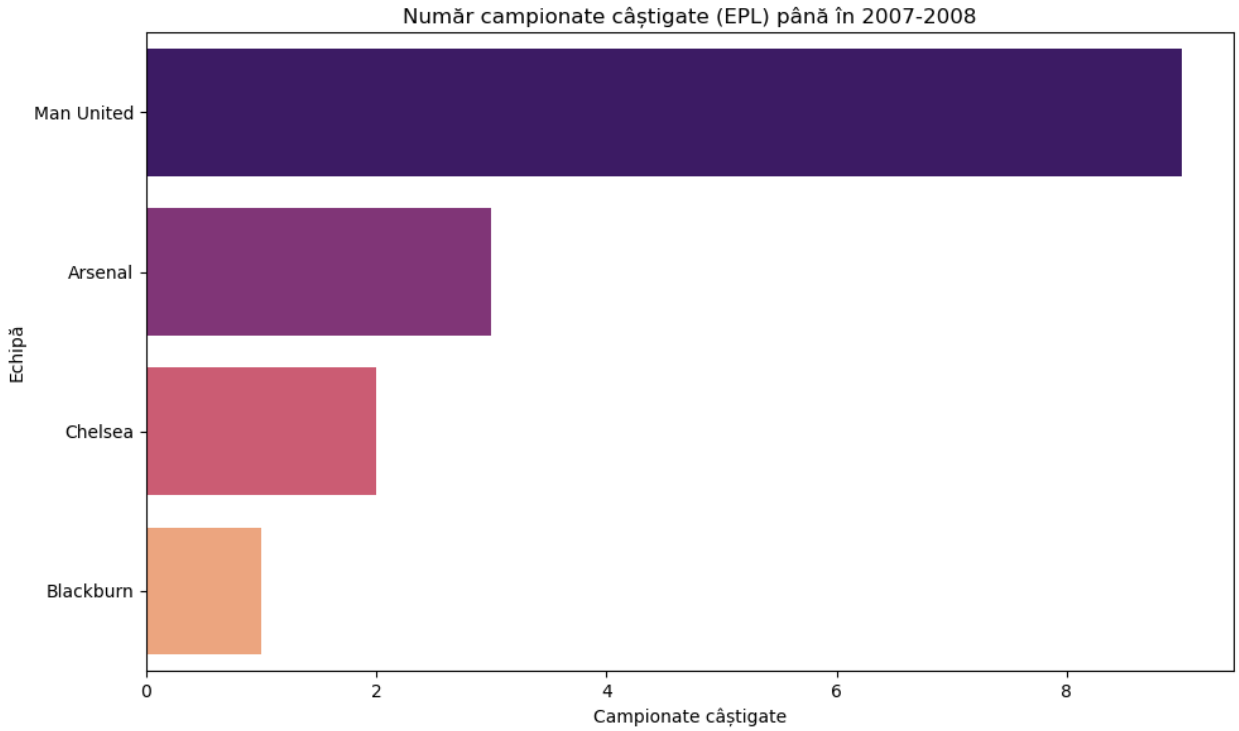
Aici putem observa distributia echipelor care au castigat titlul de campioana in ultimii 30 de ani. Am facut si o vizualizare a echipelor care au castigat titlul de campioana in ultimii 30 de ani, si pentru studiul nostru despre comparatia intre Manchester United si Manchester City, am facut o vizualizare a echipelor care au castigat titlul de campioana pana in sezonul 2007-2008 si o alta vizualizare care ia in calcul datele din sezonul 2008-2009 si am observat ca se vede clar o cumpana intre cele 2 echipe. Din anul 2008-2009, Manchester City a inceput sa castige titluri de campioana, cand Manchester United a inceput sa fie mai deficitara cu rezultatele.

```
# Convertim winners_counter în Pandas
winners_counter_pd = winners_counter.toPandas()

plt.figure(figsize=(10, 6))
sns.barplot(data=winners_counter_pd, x="Championships", y="Team",
            palette="magma", hue="Team", dodge=False, legend=False)
plt.title("Număr campionate câștigate (EPL)")
plt.xlabel("Campionate câștigate")
plt.ylabel("Echipă")
plt.tight_layout()
plt.show()

# Filtrăm câștigătorii din 2008-2009 inclusiv și după
```

```
winners_2008plus = winners.filter(col("Season") >= "2008-2009")

# Numărăm câte campionate a câștigat fiecare echipă în această
perioadă
winners_counter_2008plus = winners_2008plus.groupBy("Team").agg(
    count("Season").alias("Championships")
).orderBy(col("Championships").desc())

# Convertim în Pandas și plot
winners_counter_2008plus_pd = winners_counter_2008plus.toPandas()

plt.figure(figsize=(10, 6))
sns.barplot(data=winners_counter_2008plus_pd, x="Championships",
y="Team",
            palette="magma", hue="Team", dodge=False, legend=False)
plt.title("Număr campionate câștigate (EPL) din 2008-2009")
plt.xlabel("Campionate câștigate")
plt.ylabel("Echipă")
plt.tight_layout()
plt.show()

# Filtrăm câștigătorii până în 2007-2008 inclusiv
winners_pre2008 = winners.filter(col("Season") <= "2007-2008")

# Numărăm câte campionate a câștigat fiecare echipă în această
perioadă
winners_counter_pre2008 = winners_pre2008.groupBy("Team").agg(
    count("Season").alias("Championships")
).orderBy(col("Championships").desc())

# Convertim în Pandas și plot
winners_counter_pre2008_pd = winners_counter_pre2008.toPandas()

plt.figure(figsize=(10, 6))
sns.barplot(data=winners_counter_pre2008_pd, x="Championships",
y="Team",
            palette="magma", hue="Team", dodge=False, legend=False)
plt.title("Număr campionate câștigate (EPL) până în 2007-2008")
plt.xlabel("Campionate câștigate")
plt.ylabel("Echipă")
plt.tight_layout()
plt.show()
```


Număr campionate câștigate (EPL)
Man United
Man City
Chelsea
Arsenal
Liverpool
Blackburn
Leicester
Echipă
0
2
4
6
8
10
12
Campionate câștigate


Număr campionate câștigate (EPL) din 2008-2009
Man City
Man United
Chelsea
Liverpool
Leicester
Echipă
0
1
2
3
4
5
6
7
8
Campionate câștigate




## 3. Line plot: Evoluția punctelor si clasamentului pentru o echipă de top pe sezoane

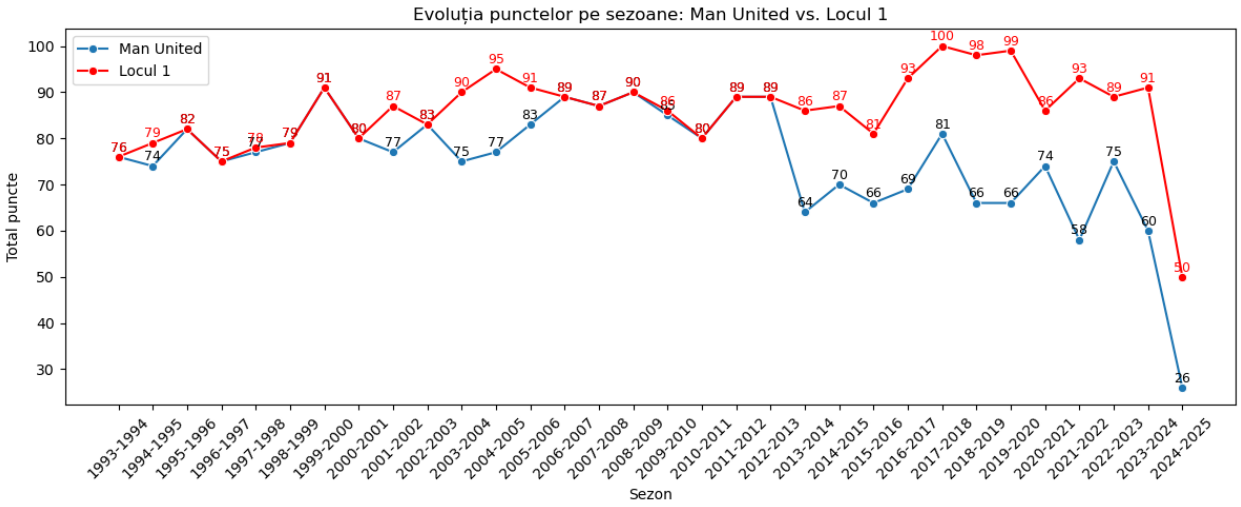
Graficele următoare ilustrează evoluția echipelor Manchester United si Manchester City în Premier League de-a lungul sezoanelor analizate. Primul grafic prezintă totalul punctelor obținute al echipelor în fiecare sezon, evidențiind performanța lor de la an la an. Al doilea grafic arată poziția ocupată în clasament la finalul fiecărui sezon, oferind o perspectivă clară asupra constanței și progresului echipelor în EPL.

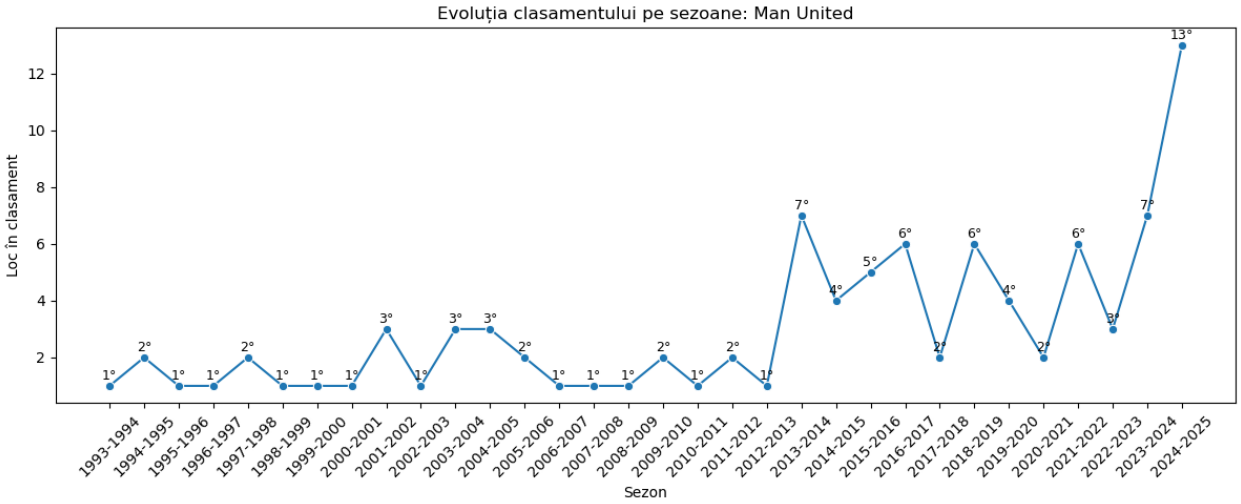
Se poate observa cum pentru echipa Manchester United, pana in sezonul 2012-2013, echipa a fost una dintre cele mai bune echipe din EPL, luptand in fiecare an pentru titlul de campioana; Din sezonul urmator a inceput o perioada de cadere, in care echipa nu a reusit sa castige titlul de campioana, si nici sa se califice in competitia europeana Champions League in mai multe ocazii (fiind calificarea intre primele 4 locuri).

Iar pentru Manchester City, se poate observa ca din sezonul 2009-2010 echipa a inceput sa castige mai multe puncte (multumita si faptului ca a fost achizitionata de catre Abu Dhabi United Group in Septembrie 2008 si a inceput sa investeasca in echipa); din momentul acela echipa a castigat 8 titluri de campioana (primul find acela din sezonul 2011-2012); Se mai poate observa ca de atunci a fost aproape un monopol al echipei, care din sezonul 2017-2018 a castigat 6 titluri in 7 ani (lant rupt de Liverpool in 2019-2020).

```
team_name = "Man United"
team_history = standings_season.filter(
    col("Team") == team_name).orderBy("Season").toPandas()

# Obținem punctele echipei de pe primul loc pentru fiecare sezon
first_place = standings_season.filter(
```

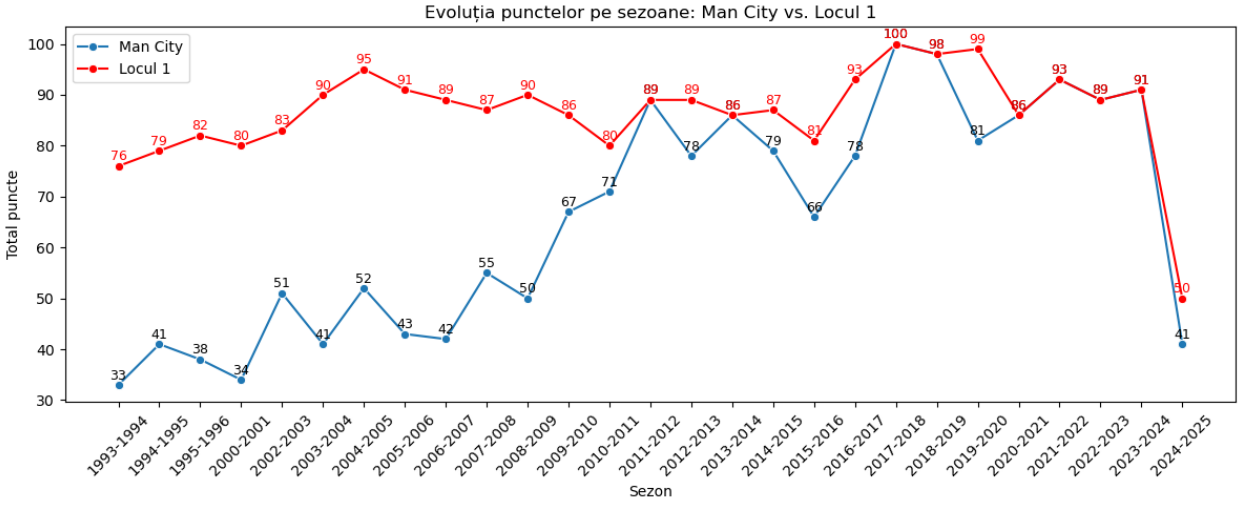
```
    col("Rank") == 1).orderBy("Season").toPandas()
first_place = first_place[["Season", "TotalPoints"]].rename(
    columns={"TotalPoints": "FirstPlacePoints"})

# Facem merge pentru a avea și punctele primului loc în același
DataFrame
team_history = team_history.merge(first_place, on="Season",
how="left")

plt.figure(figsize=(12, 5))
sns.lineplot(data=team_history, x="Season",
             y="TotalPoints", marker="o", label=team_name)
sns.lineplot(data=team_history, x="Season", y="FirstPlacePoints",
             marker="o", color="red", label="Locul 1")
plt.title(f"Evoluția punctelor pe sezoane: {team_name} vs. Locul 1")
plt.xlabel("Sezon")
plt.ylabel("Total puncte")
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()

for i, row in team_history.iterrows():
    plt.text(row["Season"], row["TotalPoints"] + 0.5,
str(int(row["TotalPoints"])),
             ha='center', va='bottom', fontsize=9, color='black')
    plt.text(row["Season"], row["FirstPlacePoints"] + 0.5,
str(int(row["FirstPlacePoints"])),
             ha='center', va='bottom', fontsize=9, color='red')

plt.show()

plt.figure(figsize=(12, 5))
sns.lineplot(data=team_history, x="Season", y="Rank", marker="o")
plt.title(f"Evoluția clasamentului pe sezoane: {team_name}")
plt.xlabel("Sezon")
plt.ylabel("Loc în clasament")
plt.xticks(rotation=45)
plt.tight_layout()

# Adaugă valoarea la fiecare punct
for i, row in team_history.iterrows():
    plt.text(row["Season"], row["Rank"] + 0.1, str(int(row["Rank"])) +
"°",
             ha='center', va='bottom', fontsize=9, color='black')

plt.show()
```

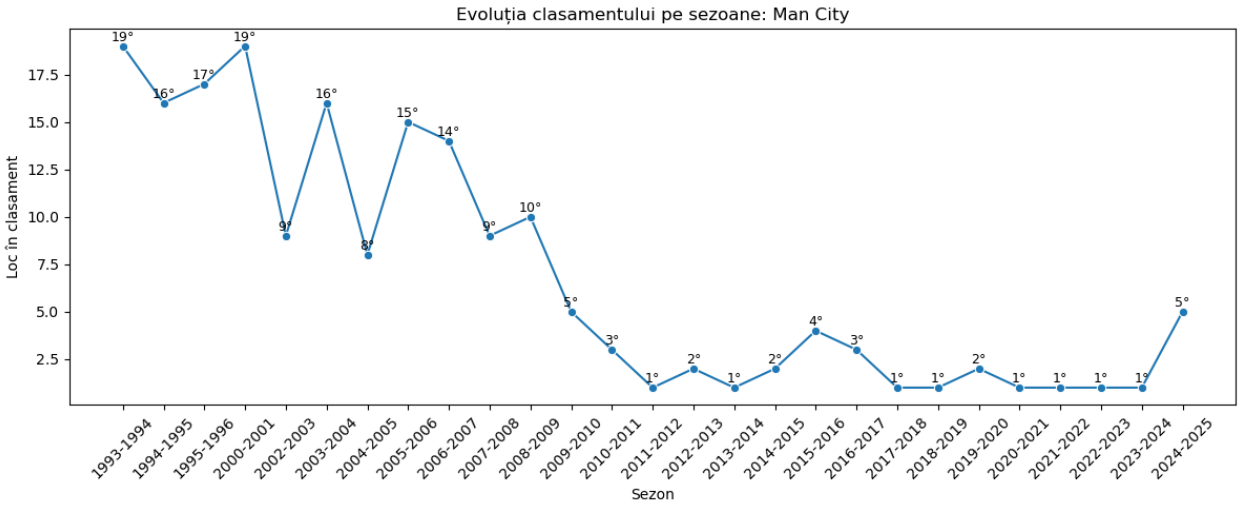



```
team_name = "Man City"
team_history = standings_season.filter(
    col("Team") == team_name).orderBy("Season").toPandas()

# Obținem punctele echipei de pe primul loc pentru fiecare sezon
first_place = standings_season.filter(
    col("Rank") == 1).orderBy("Season").toPandas()
first_place = first_place[["Season", "TotalPoints"]].rename(
    columns={"TotalPoints": "FirstPlacePoints"})

# Facem merge pentru a avea și punctele primului loc în același
DataFrame
team_history = team_history.merge(first_place, on="Season",
how="left")

plt.figure(figsize=(12, 5))
sns.lineplot(data=team_history, x="Season",
             y="TotalPoints", marker="o", label=team_name)
```

```
sns.lineplot(data=team_history, x="Season", y="FirstPlacePoints",
             marker="o", color="red", label="Locul 1")
plt.title(f"Evoluția punctelor pe sezoane: {team_name} vs. Locul 1")
plt.xlabel("Sezon")
plt.ylabel("Total puncte")
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()

for i, row in team_history.iterrows():
    plt.text(row["Season"], row["TotalPoints"] + 0.5,
str(int(row["TotalPoints"])),
             ha='center', va='bottom', fontsize=9, color='black')
    plt.text(row["Season"], row["FirstPlacePoints"] + 0.5,
str(int(row["FirstPlacePoints"])),
             ha='center', va='bottom', fontsize=9, color='red')

plt.show()

plt.figure(figsize=(12, 5))
sns.lineplot(data=team_history, x="Season", y="Rank", marker="o")
plt.title(f"Evoluția clasamentului pe sezoane: {team_name}")
plt.xlabel("Sezon")
plt.ylabel("Loc în clasament")
plt.xticks(rotation=45)
plt.tight_layout()

# Adaugă valoarea la fiecare punct
for i, row in team_history.iterrows():
    plt.text(row["Season"], row["Rank"] + 0.1, str(int(row["Rank"])) +
"°",
             ha='center', va='bottom', fontsize=9, color='black')

plt.show()
```



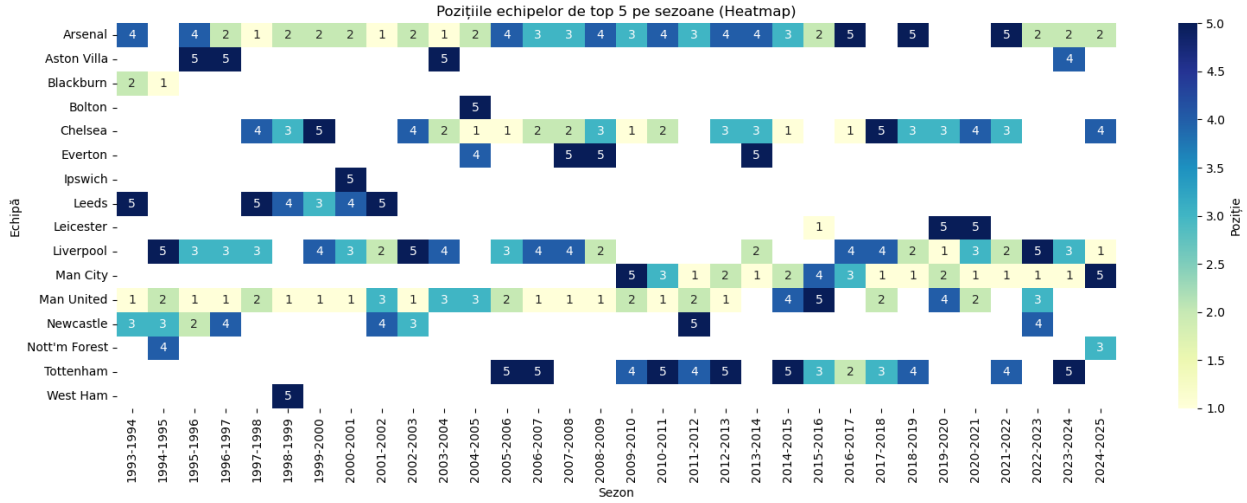



## 4. Heatmap: Pozițiile echipelor de top pe sezoane

In aceasta sectiune vom analiza cum au evoluat echipele in timp, si cum au evoluat si clasamentul lor.

Putem observa spre exemplu echipa Leicester City, care a castigat titlul de campioana in sezonul 2015-2016, echipa care la inceputul sezonului 2015-2016 era considerata ca o echipa care avea sanse de a retrogada in Liga 2 -> Chmapionship. Alt exemplu sunt echipele Liverpool, Arsenal si Manchester United care pot fi considerate cele "4 surori" al campionatului deoarece toate au fost în mod regulat în topul clasamentului pe termen lung (nu doar un boom de cativa ani, cum a fost la Manchester City sau Tottenham mai recent).

```
bump_data = standings_season.filter(
    col("Rank") <= 5).orderBy("Season", "Team").toPandas()

# Pregătim un pivot pentru heatmap
heatmap_data = bump_data.pivot(index="Team", columns="Season",
values="Rank")

plt.figure(figsize=(16, 6))
sns.heatmap(heatmap_data, annot=True, cmap="YlGnBu",
            cbar_kws={'label': 'Poziție'})
plt.title("Pozițiile echipelor de top 5 pe sezoane (Heatmap)")
plt.xlabel("Sezon")
plt.ylabel("Echipă")
plt.tight_layout()
plt.show()
```




## II. Distribuția rezultatelor (victorii, egaluri, înfrângeri) pentru fiecare echipă

Vom calcula pentru fiecare echipă câte victorii, egaluri și înfrângeri a avut de-a lungul timpului, atât acasă cât și în deplasare.

```
from pyspark.sql.functions import when

# Victorii acasă
home_wins = df_general.groupBy("HomeTeam").agg(
    (spark_sum(when(col("FullTimeResult") == "H",
1).otherwise(0))).alias("HomeWins"),
    (spark_sum(when(col("FullTimeResult") == "D",
1).otherwise(0))).alias("HomeDraws"),
    (spark_sum(when(col("FullTimeResult") == "A",
1).otherwise(0))).alias("HomeLosses")
).withColumnRenamed("HomeTeam", "Team")

# Victorii în deplasare
away_wins = df_general.groupBy("AwayTeam").agg(
    (spark_sum(when(col("FullTimeResult") == "A",
1).otherwise(0))).alias("AwayWins"),
    (spark_sum(when(col("FullTimeResult") == "D",
1).otherwise(0))).alias("AwayDraws"),
    (spark_sum(when(col("FullTimeResult") == "H",
1).otherwise(0))).alias("AwayLosses")
).withColumnRenamed("AwayTeam", "Team")

# Join și totaluri
results = home_wins.join(away_wins, on="Team", how="outer") \
    .withColumn("TotalWins", coalesce(col("HomeWins"), lit(0)) +
coalesce(col("AwayWins"), lit(0))) \
    .withColumn("TotalDraws", coalesce(col("HomeDraws"), lit(0)) +
coalesce(col("AwayDraws"), lit(0))) \
```

```
    .withColumn("TotalLosses", coalesce(col("HomeLosses"), lit(0)) +
coalesce(col("AwayLosses"), lit(0)))

results = results.orderBy(col("TotalWins").desc())
results.show(10, truncate=False)
```

```
+-----------+--------+---------+----------+--------+---------
+----------+---------+----------+-----------+
|Team       |HomeWins|HomeDraws|HomeLosses|AwayWins|AwayDraws|
AwayLosses|TotalWins|TotalDraws|TotalLosses|
+-----------+--------+---------+----------+--------+---------
+----------+---------+----------+-----------+
|Man United |413     |110      |75        |305     |149      |143
|718      |259       |218        |
|Arsenal    |394     |131      |72        |272     |156      |169
|666      |287       |241        |
|Liverpool  |381     |143      |71        |263     |151      |181
|644      |294       |252        |
|Chelsea    |363     |150      |84        |277     |139      |180
|640      |289       |264        |
|Tottenham  |326     |129      |142       |200     |152      |243
|526      |281       |385        |
|Man City   |308     |94       |100       |213     |116      |174
|521      |210       |274        |
|Everton    |275     |155      |167       |149     |180      |268
|424      |335       |435        |
|Newcastle  |277     |138      |143       |147     |148      |265
|424      |286       |408        |
|Aston Villa|223     |154      |164       |155     |148      |235
|378      |302       |399        |
|West Ham   |231     |135      |172       |129     |134      |276
|360      |269       |448        |
+-----------+--------+---------+----------+--------+---------
+----------+---------+----------+-----------+
only showing top 10 rows
```

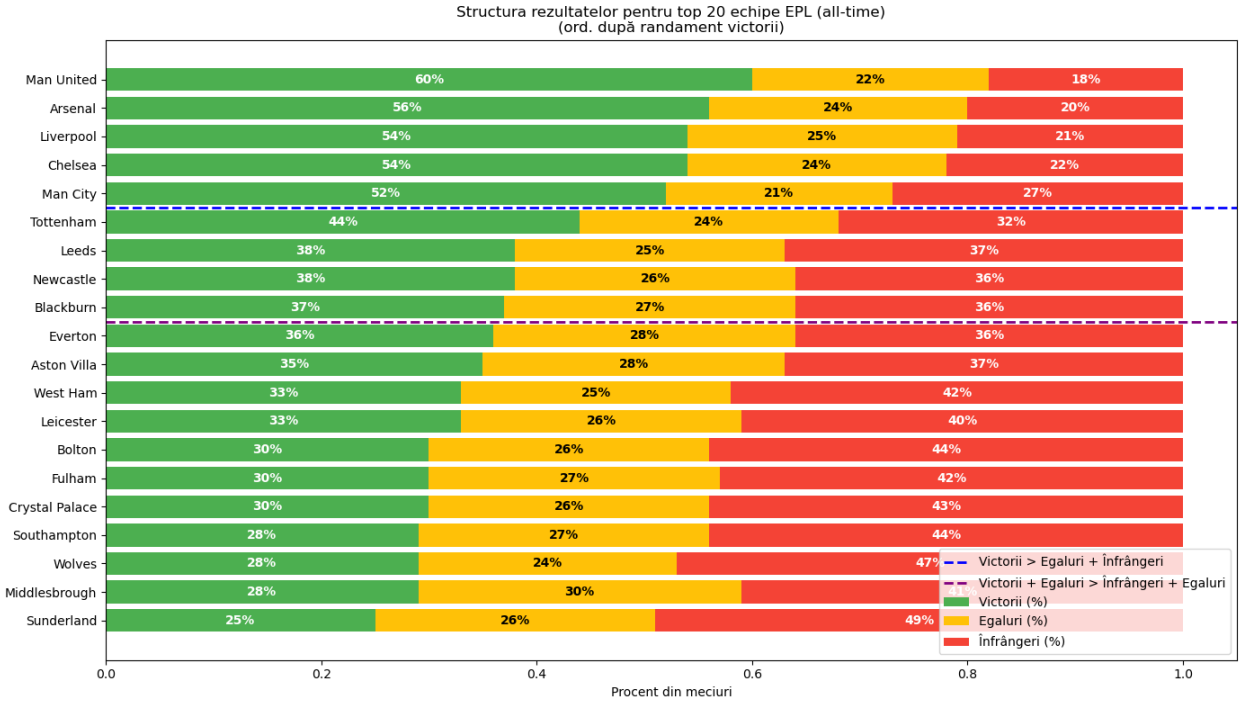
1. Stacked Bar Chart pentru Top 10 echipe (victorii, egaluri, înfrângeri)

In graficul acesta putem observa ca din totate echipele care au jucat in Premier League, doar 5 au un randament super pozitiv unde au castigat mai multe meciuri decat au pierdut si egalat; In cazul in care luam in calcul victorii + egaluri, putem observa ca 9 echipe au un randament pozitiv;

```
import matplotlib.pyplot as plt
import numpy as np

# Convertim rezultatele în Pandas și selectăm top 20 echipe după
victorii
top20_results =
results.orderBy(col("TotalWins").desc()).limit(20).toPandas()
```

```
# Calculăm total meciuri și randamentul
top20_results["TotalMatches"] = top20_results["TotalWins"] + \
    top20_results["TotalDraws"] + top20_results["TotalLosses"]
top20_results["WinRatio"] = top20_results["TotalWins"] / \
    top20_results["TotalMatches"]

# Ordonăm descrescător după randament
top20_results = top20_results.sort_values(
    by="WinRatio", ascending=False).reset_index(drop=True)

teams = top20_results["Team"]
wins = top20_results["TotalWins"]
draws = top20_results["TotalDraws"]
losses = top20_results["TotalLosses"]
total_matches = top20_results["TotalMatches"]

wins_pct = (wins / total_matches).round(2)
draws_pct = (draws / total_matches).round(2)
losses_pct = (losses / total_matches).round(2)

# Ajustăm ultima categorie pentru a evita depășirea 100%
for i in range(len(teams)):
    s = wins_pct.iloc[i] + draws_pct.iloc[i] + losses_pct.iloc[i]
    if s != 1.0:
        losses_pct.iloc[i] = 1.0 - wins_pct.iloc[i] -
draws_pct.iloc[i]

# Calculăm pozițiile pentru linii separatorii
# 1. Unde victorii > egaluri + înfrângeri
sep1 = np.where(wins.values > (draws.values + losses.values))[0]
sep1_pos = sep1[-1] + 0.5 if len(sep1) > 0 else None

# 2. Unde victorii + egaluri > înfrângeri + egaluri (adică victorii >
înfrângeri)
sep2 = np.where((wins.values + draws.values) >
                (losses.values + draws.values))[0]
sep2_pos = sep2[-1] + 0.5 if len(sep2) > 0 else None

plt.figure(figsize=(14, 8))
plt.barh(teams, wins_pct, color="#4CAF50", label="Victorii (%)")
plt.barh(teams, draws_pct, left=wins_pct, color="#FFC107",
label="Egaluri (%)")
plt.barh(teams, losses_pct, left=wins_pct+draws_pct,
         color="#F44336", label="Înfrângeri (%)")

plt.xlabel("Procent din meciuri")
plt.title("Structura rezultatelor pentru top 20 echipe EPL (all-time)\
n(ord. după randament victorii)")
plt.legend(loc="lower right")
```

```
plt.gca().invert_yaxis()
plt.tight_layout()

# Adăugăm procentul în fiecare dreptunghi
for i, team in enumerate(teams):
    if wins_pct.iloc[i] > 0.03:
        plt.text(wins_pct.iloc[i]/2, i, f"{int(wins_pct.iloc[i]*100)}
%",
                 va='center', ha='center', color='white',
fontweight='bold')
    if draws_pct.iloc[i] > 0.03:
        plt.text(wins_pct.iloc[i] + draws_pct.iloc[i]/2, i,
                 f"{int(draws_pct.iloc[i]*100)}%", va='center',
ha='center', color='black', fontweight='bold')
    if losses_pct.iloc[i] > 0.03:
        plt.text(wins_pct.iloc[i] + draws_pct.iloc[i] +
losses_pct.iloc[i]/2, i,
                 f"{int(losses_pct.iloc[i]*100)}%", va='center',
ha='center', color='white', fontweight='bold')

# Adăugăm liniile separatorii
if sep1_pos is not None:
    plt.axhline(sep1_pos, color='blue', linestyle='--',
                linewidth=2, label="Victorii > Egaluri + Înfrângeri")
if sep2_pos is not None:
    plt.axhline(sep2_pos, color='purple', linestyle='--', linewidth=2,
                label="Victorii + Egaluri > Înfrângeri + Egaluri")

# Adăugăm legendă pentru linii
handles, labels = plt.gca().get_legend_handles_labels()
if sep1_pos is not None or sep2_pos is not None:
    plt.legend(handles, labels, loc="lower right")

plt.show()
```

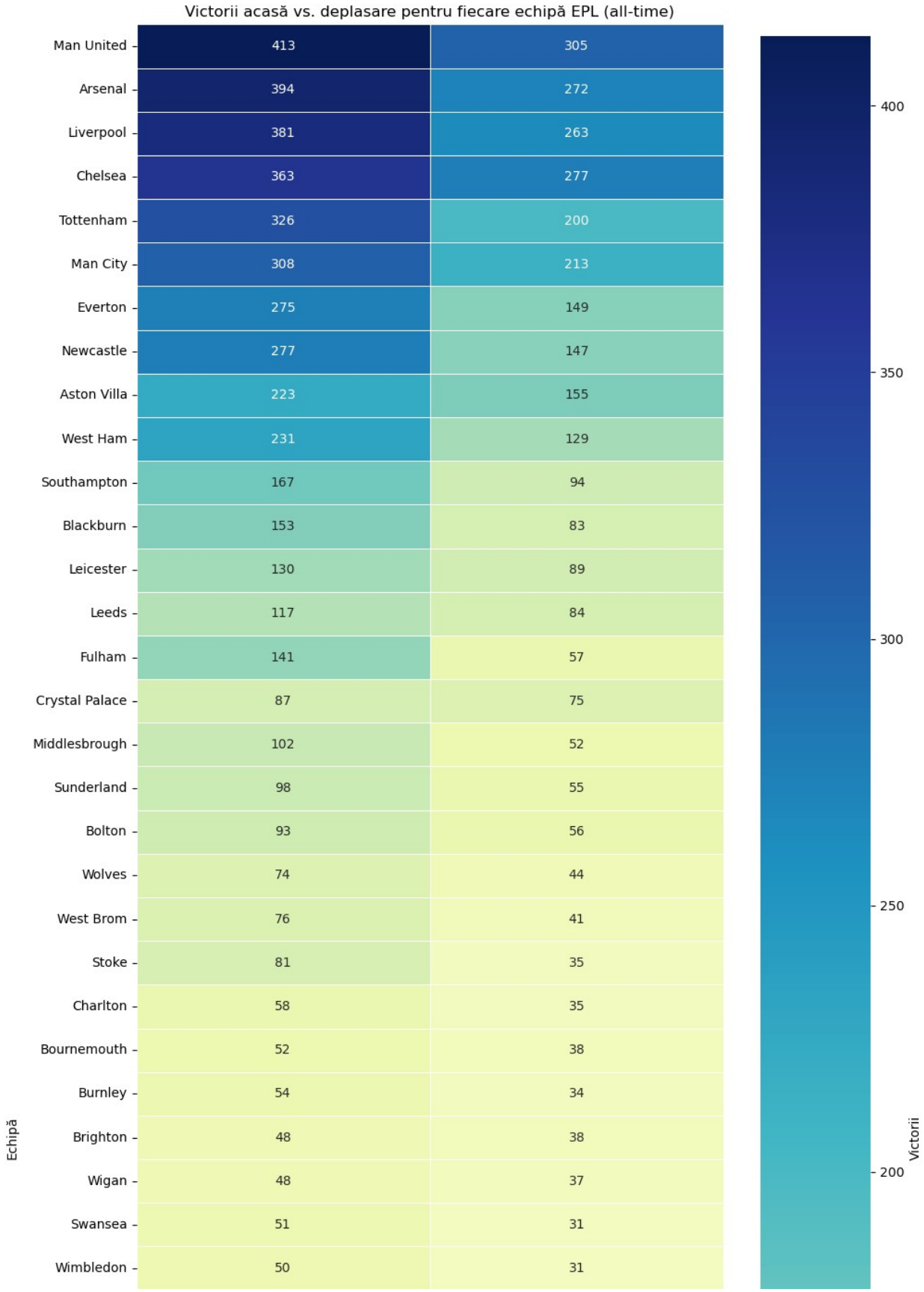



## 2. Heatmap Victorii Acasă vs. Deplasare pentru fiecare echipă

Din Vizualizarea de mai sus putem observa ca echipele de obicei castiga mai multe meciuri acasa decat in deplasare, unicele echipe care au Acasa = Deplasare sunt Blackpool si Oldham cu 5 si respectiv 4 meciuri castigate in ambele situatii.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Selectăm doar coloanele relevante și convertim în Pandas
victories = results.select("Team", "HomeWins", "AwayWins").toPandas()

# Facem un melt pentru a avea o coloană 'Type' (Home/Away) și una cu
valorile
victories_melted = victories.melt(id_vars="Team",
value_vars=["HomeWins", "AwayWins"],
                                  var_name="Type", value_name="Wins")
victories_melted["Type"] = victories_melted["Type"].replace(
    {"HomeWins": "Acasă", "AwayWins": "Deplasare"})

# Pentru claritate, sortăm echipele după total victorii
victories["TotalWins"] = victories["HomeWins"] + victories["AwayWins"]
top_teams = victories.sort_values("TotalWins", ascending=False)[
    "Team"].tolist()

plt.figure(figsize=(10, max(6, len(top_teams)//2)))
heatmap_data = victories_melted.pivot(
```

```
    index="Team", columns="Type", values="Wins").loc[top_teams]

sns.heatmap(heatmap_data, annot=True, fmt=".0f", cmap="YlGnBu",
            linewidths=0.5, cbar_kws={'label': 'Victorii'})
plt.title("Victorii acasă vs. deplasare pentru fiecare echipă EPL
(all-time)")
plt.xlabel("Tip victorie")
plt.ylabel("Echipă")
plt.tight_layout()
plt.show()
```


Victorii acasă vs. deplasare pentru fiecare echipă EPL (all-time)
Man United 413 305
Arsenal 394 272
Liverpool 381 263
Chelsea 363 277
Tottenham 326 200
Man City 308 213
Everton 275 149
Newcastle 277 147
Aston Villa 223 155
West Ham 231 129
Southampton 167 94
Blackburn 153 83
Leicester 130 89
Leeds 117 84
Fulham 141 57
Crystal Palace 87 75
Middlesbrough 102 52
Sunderland 98 55
Bolton 93 56
Wolves 74 44
West Brom 76 41
Stoke 81 35
Charlton 58 35
Bournemouth 52 38
Burnley 54 34
Brighton 48 38
Wigan 48 37
Swansea 51 31
Wimbledon 50 31
Echipă
400
350
300
250
200
Victorii

## III. Total goluri marcate și primite de fiecare echipă

Vom calcula câte goluri a marcat și a primit fiecare echipă, atât acasă cât și în deplasare, și totalul pe toată perioada.

```
# Goluri marcate acasă și primite acasă
home_goals = df_general.groupBy("HomeTeam").agg(
    spark_sum("FullTimeHomeTeamGoals").alias("GoalsScoredHome"),
    spark_sum("FullTimeAwayTeamGoals").alias("GoalsConcededHome")
).withColumnRenamed("HomeTeam", "Team")

# Goluri marcate în deplasare și primite în deplasare
away_goals = df_general.groupBy("AwayTeam").agg(
    spark_sum("FullTimeAwayTeamGoals").alias("GoalsScoredAway"),
    spark_sum("FullTimeHomeTeamGoals").alias("GoalsConcededAway")
).withColumnRenamed("AwayTeam", "Team")

# Join și totaluri
goals = home_goals.join(away_goals, on="Team", how="outer") \
    .withColumn("TotalGoalsScored", coalesce(col("GoalsScoredHome"),
lit(0)) + coalesce(col("GoalsScoredAway"), lit(0))) \
    .withColumn("TotalGoalsConceded",
coalesce(col("GoalsConcededHome"), lit(0)) +
coalesce(col("GoalsConcededAway"), lit(0)))

goals = goals.orderBy(col("TotalGoalsScored").desc())
goals.show(10, truncate=False)
```

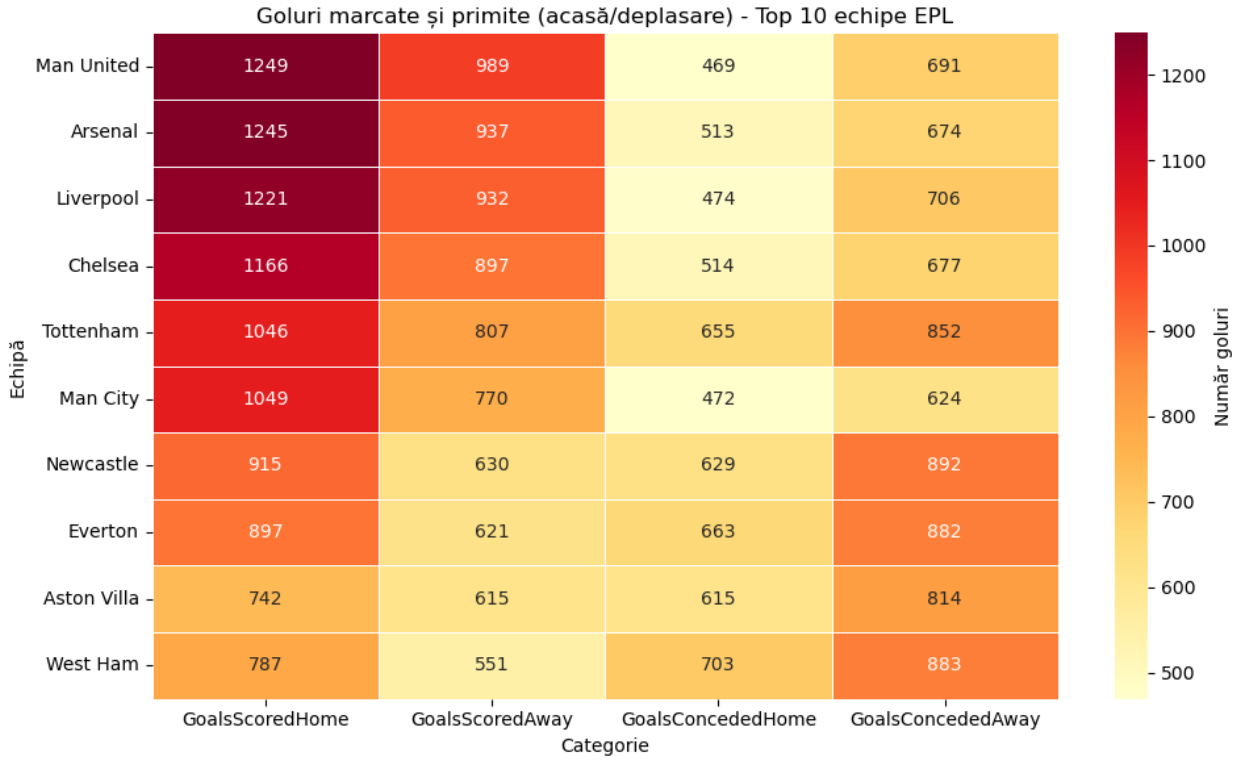
```
+----------+--------------+----------------+--------------
+----------------+---------------+-----------------+
|Team      |GoalsScoredHome|GoalsConcededHome|GoalsScoredAway|
GoalsConcededAway|TotalGoalsScored|TotalGoalsConceded|
+----------+--------------+----------------+--------------
+----------------+---------------+-----------------+
|Man United |1249           |469              |989            |691
|2238            |1160             |
|Arsenal    |1245           |513              |937            |674
|2182            |1187             |
|Liverpool  |1221           |474              |932            |706
|2153            |1180             |
|Chelsea    |1166           |514              |897            |677
|2063            |1191             |
|Tottenham  |1046           |655              |807            |852
|1853            |1507             |
|Man City   |1049           |472              |770            |624
|1819            |1096             |
|Newcastle  |915            |629              |630            |892
|1545            |1521             |
|Everton    |897            |663              |621            |882
|1518            |1545             |
```

```
|Aston Villa|742            |615              |615              |814
|1357            |1429              |
|West Ham   |787            |703              |551              |883
|1338            |1586              |
+-----------+---------------+-----------------+---------------
+-----------------+----------------+------------------+
only showing top 10 rows
```

1. HeatMap: Goluri Marcate vs. Primite (Top 10 echipe)

Din mapa aceasta se poate observa ca echipele cele mai multe goluri Acasa si primeste mai multe goluri in deplasare; Caz curios este echipa Aston Villa care are 615 goluri marcate in deplasare si 615 goluri primite acasa.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Selectăm top 10 echipe după goluri marcate
top10_teams = goals.orderBy(
    col("TotalGoalsScored").desc()).limit(10).toPandas()

# Pregătim un DataFrame pentru heatmap
heatmap_data = top10_teams.set_index("Team")[
    ["GoalsScoredHome", "GoalsScoredAway", "GoalsConcededHome",
"GoalsConcededAway"]]

plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, annot=True, fmt=".0f", cmap="YlOrRd",
            linewidths=0.5, cbar_kws={'label': 'Număr goluri'})
plt.title("Goluri marcate și primite (acasă/deplasare) - Top 10 echipe
EPL")
plt.xlabel("Categorie")
plt.ylabel("Echipă")
plt.tight_layout()
plt.show()
```




## IV. Distribuția scorurilor (cele mai frecvente scoruri finale)

Vom analiza cele mai frecvente scoruri finale din istoria EPL si analizam si situatia de comeback-uri, rezultat halftime/fulltime, care rezultat e mai usor de intors si de cate ori au fost acestea si ce echipe au reusit cel mai mult sa intoarca rezultatul.

```
from pyspark.sql.functions import concat_ws, when, col

# Creăm o coloană cu scorul final sub formă de string (ex: "2-1")
df_general = df_general.withColumn(
    "Score", concat_ws("-", col("FullTimeHomeTeamGoals"),
                       col("FullTimeAwayTeamGoals"))
)

# Numărăm scorurile
score_counts =
df_general.groupBy("Score").count().orderBy(col("count").desc())
score_counts.show(100, truncate=False)

# Filtrăm doar meciurile cu rezultat la pauză diferit de cel final (și
nu egal la pauză)
comebacks = df_detailed.filter(
    (col("HalfTimeResult").isin(["H", "A"])) &
    (col("FullTimeResult") != col("HalfTimeResult"))
)
```

```
# Identificăm tipul de comeback
comebacks = comebacks.withColumn(
    "ComebackType",
    when(
        (col("HalfTimeResult") == "A") & (
            col("FullTimeResult") == "H"), "Home Win Comeback"
    ).when(
        (col("HalfTimeResult") == "H") & (
            col("FullTimeResult") == "A"), "Away Win Comeback"
    ).when(
        (col("HalfTimeResult") == "A") & (
            col("FullTimeResult") == "D"), "Home Draw Comeback"
    ).when(
        (col("HalfTimeResult") == "H") & (
            col("FullTimeResult") == "D"), "Away Draw Comeback"
    )
)

# Numărăm totalul de comeback-uri pe tip
comebacks.groupBy("ComebackType").count().show(truncate=False)

# Top echipe gazdă cu cele mai multe comeback-uri (victorie după ce au
fost conduse la pauză)
home_win_comebacks = comebacks.filter(col("ComebackType") == "Home Win
Comeback") \
    .groupBy("HomeTeam").count().orderBy(col("count").desc())

# Top echipe oaspete cu cele mai multe comeback-uri (victorie după ce
au fost conduse la pauză)
away_win_comebacks = comebacks.filter(col("ComebackType") == "Away Win
Comeback") \
    .groupBy("AwayTeam").count().orderBy(col("count").desc())
```

```
+-----+-----+
|Score|count|
+-----+-----+
|1-1  |1363 |
|1-0  |1249 |
|2-1  |1040 |
|2-0  |969  |
|0-0  |944  |
|0-1  |869  |
|1-2  |747  |
|2-2  |611  |
|3-1  |547  |
|0-2  |545  |
|3-0  |504  |
|1-3  |342  |
|0-3  |269  |
|3-2  |263  |
```

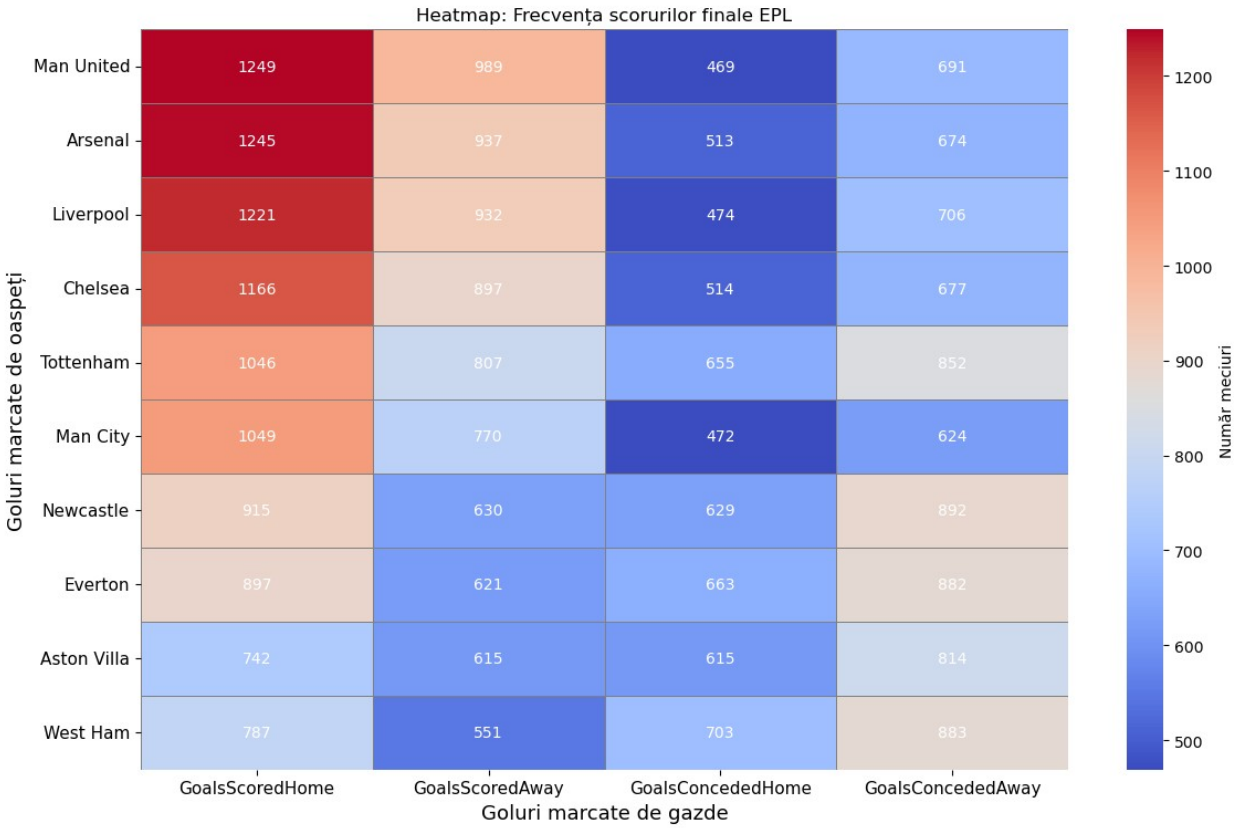
```
|4-0  |231  |
|2-3  |220  |
|4-1  |205  |
|3-3  |124  |
|0-4  |117  |
|1-4  |115  |
|4-2  |112  |
|5-0  |102  |
|5-1  |69   |
|4-3  |60   |
|2-4  |55   |
|0-5  |36   |
|3-4  |33   |
|1-5  |31   |
|6-1  |26   |
|5-2  |25   |
|6-0  |23   |
|2-5  |18   |
|4-4  |17   |
|5-3  |14   |
|0-6  |13   |
|6-2  |12   |
|1-6  |12   |
|7-1  |11   |
|7-0  |9    |
|2-6  |6    |
|7-2  |5    |
|6-3  |5    |
|8-0  |5    |
|3-5  |4    |
|5-4  |3    |
|3-6  |3    |
|9-0  |3    |
|4-5  |2    |
|1-7  |2    |
|9-1  |1    |
|0-9  |1    |
|0-8  |1    |
|1-8  |1    |
|8-2  |1    |
|7-3  |1    |
|5-5  |1    |
|0-7  |1    |
|6-4  |1    |
|7-4  |1    |
|8-1  |1    |
+-----+-----+

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

```
|ComebackType      |count|
+------------------+-----+
|Home Draw Comeback|443  |
|Away Draw Comeback|466  |
|Away Win Comeback |179  |
|Home Win Comeback |235  |
+------------------+-----+
```

### 1. Heatmap: Frecvența scorurilor finale

In aceasta harta putem observa ca cele mai des intalnite scoruri sunt 1-1, 1-0 si 2-1 cu un numar de > 1000 de meciuri. Din contra exista si meciuri care s-au terminat cu rezultate foarte spectaculoase ca 0-9, 8-2, 7-4 spre exemplu. Asta face ca EPL sa fie un campionat foarte competitiv, unde echipele trebuie sa fie foarte bine pregatite pentru fiecare meci.

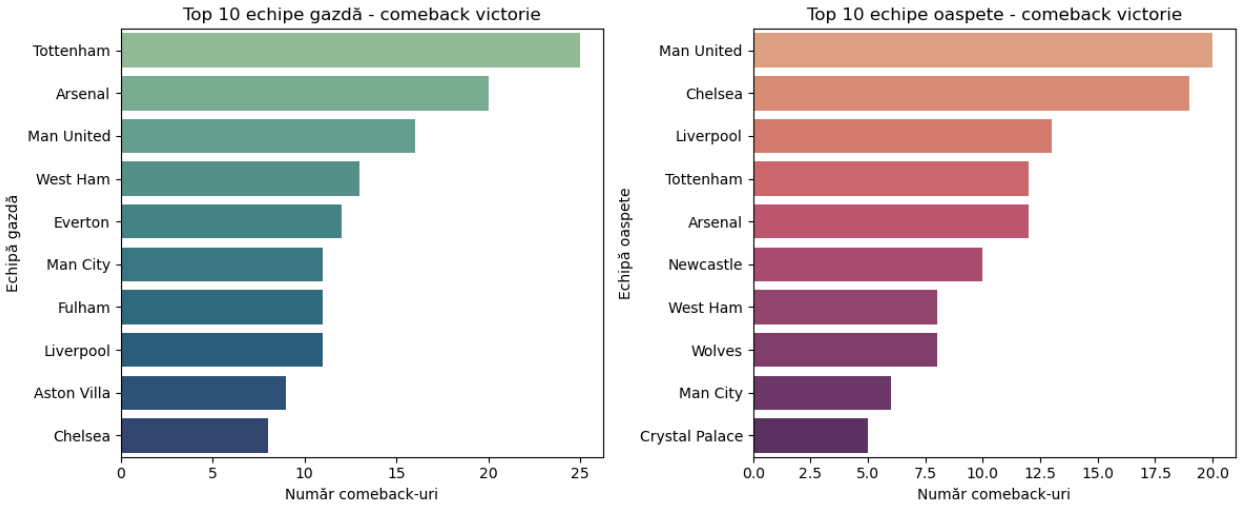
```python
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

plt.figure(figsize=(12, 8))

# Setăm ticks doar pentru scoruri care apar efectiv
x_labels = heatmap_data.columns.tolist()
y_labels = heatmap_data.index.tolist()

# Folosim o paletă mai blândă with some good palette for white text (no mako YlOrRd)
cmap = sns.color_palette("coolwarm", as_cmap=True)

ax = sns.heatmap(
    heatmap_data,
    annot=True,
    fmt="d",
    cmap=cmap,
    linewidths=0.5,
    linecolor='gray',
    cbar_kws={'label': 'Număr meciuri'},
    annot_kws={"size": 10, "color": "white"},
    xticklabels=x_labels,
    yticklabels=y_labels
)

plt.title("Heatmap: Frecvența scorurilor finale EPL")
plt.xlabel("Goluri marcate de gazde", fontsize=13)
plt.ylabel("Goluri marcate de oaspeți", fontsize=13)
plt.xticks(fontsize=11)
plt.yticks(fontsize=11, rotation=0)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```




Aici se poate observa o statistica destul de interesanta, si anume Top 10 echipe care au reusit sa castige meciuri care la pauza erau in dezavantaj (separat pentru deplasare si acasa). Interesant cum Chelsea si Liverpool au reusit sa castige mai multe meciuri in deplasare din situatie de dezavantaj fata de cele jucate acasa.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Convertim în Pandas pentru plotting
top10_home = home_win_comebacks.limit(10).toPandas()
top10_away = away_win_comebacks.limit(10).toPandas()

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.barplot(data=top10_home, x="count", y="HomeTeam",
            palette="crest", hue="HomeTeam", legend=False)
plt.title("Top 10 echipe gazdă - comeback victorie")
plt.xlabel("Număr comeback-uri")
plt.ylabel("Echipă gazdă")

plt.subplot(1, 2, 2)
sns.barplot(data=top10_away, x="count", y="AwayTeam",
            palette="flare", hue="AwayTeam", legend=False)
plt.title("Top 10 echipe oaspete - comeback victorie")
plt.xlabel("Număr comeback-uri")
```

```
plt.ylabel("Echipă oaspete")

plt.tight_layout()
plt.show()
```




## V. Analiza Arbitrilor

Vom analiza arbitrii care au dat cele mai multe cartonase in EPL.

```
# Înregistrăm df_detailed ca tabel SQL temporar
from pyspark.sql.functions import row_number
from pyspark.sql.window import Window
df_detailed.createOrReplaceTempView("matches")

arbitri_cartonase = spark.sql("""
    SELECT
        Referee,
        COUNT(*) AS Matches,
        SUM(HomeTeamYellowCards + AwayTeamYellowCards) AS TotalYellow,
        SUM(HomeTeamRedCards + AwayTeamRedCards) AS TotalRed,
        ROUND(AVG(HomeTeamYellowCards + AwayTeamYellowCards), 2) AS
AvgYellowPerMatch,
        ROUND(AVG(HomeTeamRedCards + AwayTeamRedCards), 2) AS
AvgRedPerMatch
    FROM matches
    GROUP BY Referee
    ORDER BY TotalYellow DESC
""")
arbitri_cartonase.show(20, truncate=False)

# Cartonașe galbene primite de fiecare echipă de la fiecare arbitru
fav_yellow = spark.sql("""
    SELECT
```

```
        Referee,
        HomeTeam AS Team,
        SUM(HomeTeamYellowCards) AS YellowCards
    FROM matches
    GROUP BY Referee, HomeTeam
    UNION ALL
    SELECT
        Referee,
        AwayTeam AS Team,
        SUM(AwayTeamYellowCards) AS YellowCards
    FROM matches
    GROUP BY Referee, AwayTeam
""")

# Pentru fiecare arbitru, echipa cu cele mai multe cartonașe galbene
primite de la el

window =
Window.partitionBy("Referee").orderBy(col("YellowCards").desc())
fav_yellow_ranked = fav_yellow.withColumn("Rank",
row_number().over(window))
fav_yellow_max = fav_yellow_ranked.filter(
    col("Rank") == 1).select("Referee", "Team", "YellowCards")

fav_yellow_max.orderBy(col("YellowCards").desc()).show(20,
truncate=False)
```

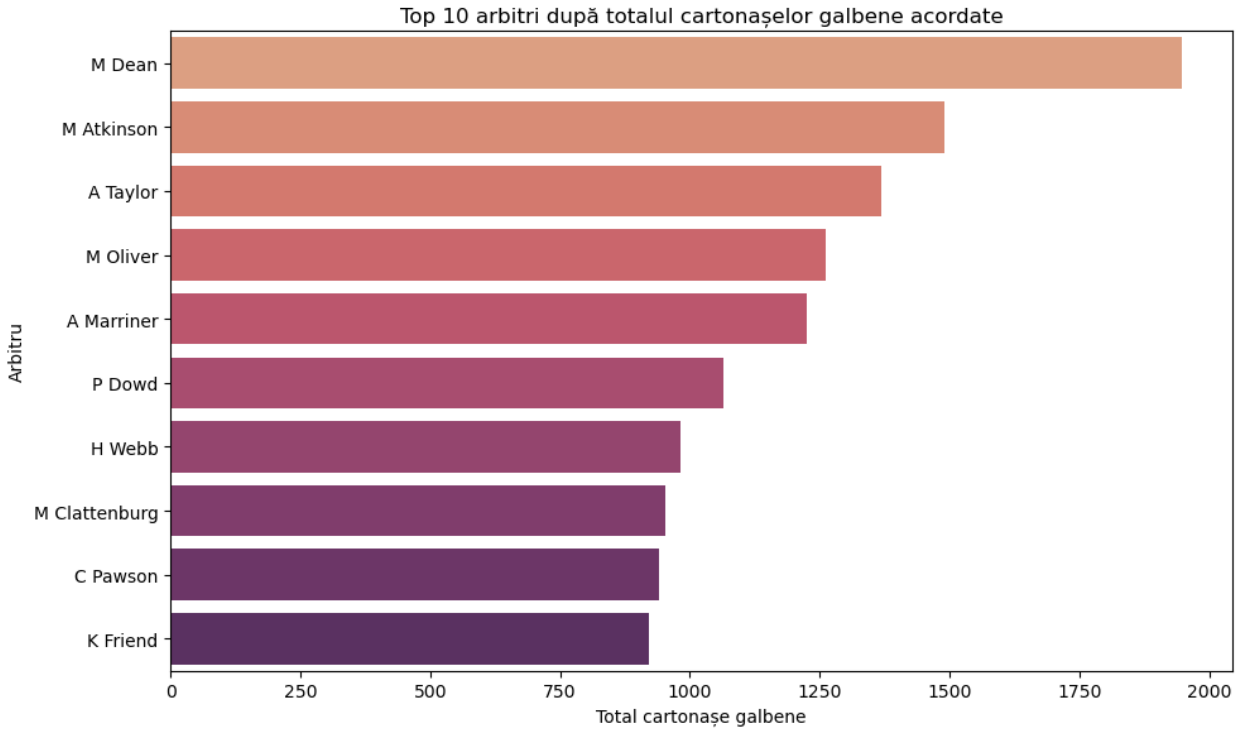
```
+-------------+-------+-----------+--------+-----------------
+--------------+
|Referee      |Matches|TotalYellow|TotalRed|AvgYellowPerMatch|
AvgRedPerMatch|
+-------------+-------+-----------+--------+-----------------
+--------------+
|M Dean       |529    |1947       |108     |3.68             |0.2
|
|M Atkinson   |460    |1489       |67      |3.24             |0.15
|
|A Taylor     |391    |1369       |53      |3.5              |0.14
|
|M Oliver     |390    |1261       |52      |3.23             |0.13
|
|A Marriner   |392    |1226       |61      |3.13             |0.16
|
|P Dowd       |301    |1065       |65      |3.54             |0.22
|
|H Webb       |297    |982        |33      |3.31             |0.11
|
|M Clattenburg|293    |953        |47      |3.25             |0.16
|
|C Pawson     |259    |940        |37      |3.63             |0.14
```

```
|
|K Friend     |272    |920         |31      |3.38              |0.11
|
|J Moss       |274    |886         |38      |3.23              |0.14
|
|L Mason      |287    |882         |40      |3.07              |0.14
|
|S Attwell    |212    |763         |29      |3.6               |0.14
|
|S Bennett    |205    |719         |41      |3.51              |0.2
|
|C Foy        |257    |696         |37      |2.71              |0.14
|
|M Jones      |202    |640         |27      |3.17              |0.13
|
|A Wiley      |207    |638         |28      |3.08              |0.14
|
|P Tierney    |183    |623         |19      |3.4               |0.1
|
|M Riley      |169    |609         |36      |3.6               |0.21
|
|R Styles     |176    |570         |50      |3.24              |0.28
|
+-------------+-------+------------+--------+------------------
+--------------+
only showing top 20 rows

+-------------+----------+-----------+
|Referee      |Team      |YellowCards|
+-------------+----------+-----------+
|M Dean       |Man United|77         |
|A Taylor     |Liverpool |72         |
|M Atkinson   |West Ham  |65         |
|A Marriner   |Man City  |58         |
|P Dowd       |Sunderland|55         |
|M Oliver     |Arsenal   |54         |
|H Webb       |Chelsea   |47         |
|A Wiley      |Bolton    |43         |
|M Clattenburg|Blackburn |43         |
|C Pawson     |Tottenham |42         |
|J Moss       |Chelsea   |38         |
|S Attwell    |Newcastle |37         |
|L Mason      |Stoke     |36         |
|S Bennett    |Man United|36         |
|M Halsey     |Chelsea   |35         |
|K Friend     |Stoke     |34         |
|C Foy        |Sunderland|33         |
|M Riley      |Liverpool |32         |
|M Jones      |Stoke     |29         |
```

```
|C Kavanagh   |Arsenal   |27          |
+-------------+----------+------------+
only showing top 20 rows
```

1. Bar chart: Top 10 arbitri după totalul cartonașelor galbene acordate

```
import matplotlib.pyplot as plt
import seaborn as sns

# Convertim arbitri_cartonase în Pandas
arbitri_cartonase_pd = arbitri_cartonase.limit(10).toPandas()

plt.figure(figsize=(10, 6))
sns.barplot(data=arbitri_cartonase_pd, x="TotalYellow",
            y="Referee", palette="flare", hue="Referee", dodge=False,
legend=False)
plt.title("Top 10 arbitri după totalul cartonașelor galbene acordate")
plt.xlabel("Total cartonașe galbene")
plt.ylabel("Arbitru")
plt.tight_layout()
plt.show()
```

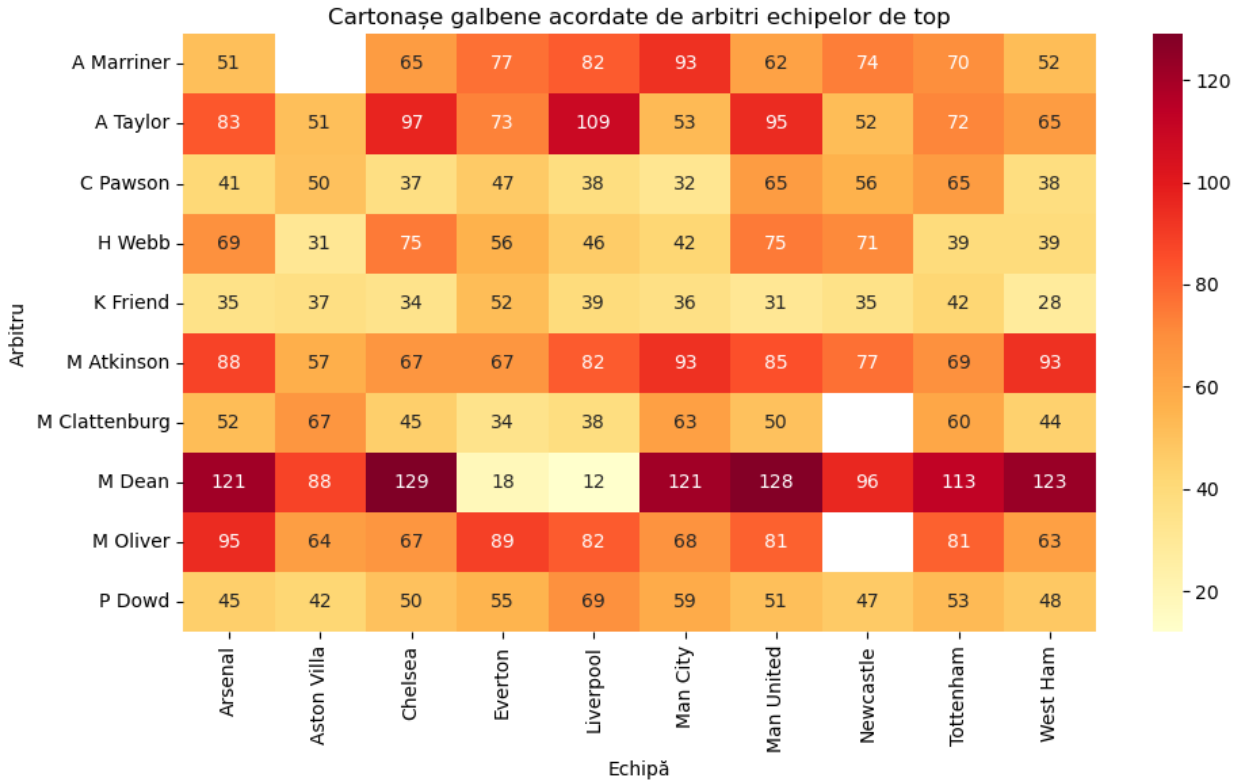


## 2. Heatmap: Cartonașe galbene acordate de arbitri echipelor de top

Se poate observa spre exemplu cum Liverpool si Everton au primit cele mai putine cartonașe galbene din partea lui Mike Dean (probabil deoarece el este dintr-un oras din apropriere de Liverpool si asociatia de fotbal a decis sa nu-l trimita la meciuri celor 2 echipe).

In schimb, tot Mike Dean este cel mai sever arbitru din EPL.

```
top_arbitri = [row["Referee"] for row in arbitri_cartonase.orderBy(
    col("TotalYellow").desc()).limit(10).collect()]
top_echipe = [row["Team"] for row in fav_yellow.groupBy("Team").sum(

"YellowCards").orderBy(col("sum(YellowCards)").desc()).limit(10).colle
ct()]

# Agregăm totalul cartonașelor galbene pentru fiecare pereche
(arbitru, echipă)
fav_yellow_heatmap_agg = fav_yellow.groupBy("Referee",
"Team").sum("YellowCards") \
    .withColumnRenamed("sum(YellowCards)", "TotalYellowCards") \
    .filter(col("Referee").isin(top_arbitri) &
col("Team").isin(top_echipe)) \
    .toPandas()

# Pivot pentru heatmap
pivot = fav_yellow_heatmap_agg.pivot(
    index="Referee", columns="Team", values="TotalYellowCards")

plt.figure(figsize=(10, 6))
sns.heatmap(pivot, annot=True, cmap="YlOrRd", fmt=".0f")
plt.title("Cartonașe galbene acordate de arbitri echipelor de top")
plt.xlabel("Echipă")
plt.ylabel("Arbitru")
plt.tight_layout()
plt.show()
```




## d. Regresie Liniară - Predicția numărului total de goluri

În această secțiune, vom construi un model de regresie liniară pentru a prezice numărul total de goluri dintr-un meci pe baza unor caracteristici precum:

- Numărul de șuturi ale ambelor echipe
- Numărul de șuturi pe poartă
- Numărul de cornere
- Numărul de faulturi

Acest tip de analiză ne poate ajuta să înțelegem care factori influențează cel mai mult productivitatea ofensivă într-un meci de fotbal. Vom folosi doar meciurile care au date complete pentru aceste statistici.

```
# Importăm bibliotecile necesare pentru regresie liniară
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml import Pipeline

# Pregătim datele pentru regresie liniară - predicția numărului total de goluri
print("=== Pregătirea datelor pentru Regresie Liniară ===")

regression_data = df_detailed.select([
```

```
    "HomeTeamShots", "AwayTeamShots",
    "HomeTeamShotsOnTarget", "AwayTeamShotsOnTarget",
    "HomeTeamCorners", "AwayTeamCorners",
    "HomeTeamFouls", "AwayTeamFouls",
    "FullTimeHomeTeamGoals", "FullTimeAwayTeamGoals"
]).dropna()

# Calculăm numărul total de goluri
regression_data = regression_data.withColumn(
    "total_goals",
    col("FullTimeHomeTeamGoals") + col("FullTimeAwayTeamGoals")
)

# Calculăm statisticile combinate pentru ambele echipe
regression_data = regression_data.withColumn(
    "total_shots", col("HomeTeamShots") + col("AwayTeamShots")
).withColumn(
    "total_shots_on_target", col(
        "HomeTeamShotsOnTarget") + col("AwayTeamShotsOnTarget")
).withColumn(
    "total_corners", col("HomeTeamCorners") + col("AwayTeamCorners")
).withColumn(
    "total_fouls", col("HomeTeamFouls") + col("AwayTeamFouls")
)

print(f"Numărul de meciuri cu date complete:
{regression_data.count()}")

# Afișăm statistici descriptive pentru variabila target
print("Statistici pentru numărul total de goluri:")
regression_data.select("total_goals").describe().show()

# Afișăm și statistici pentru cornerele totale pentru a verifica
corelația
print("Statistici pentru cornerele totale:")
regression_data.select("total_corners").describe().show()
```

```
=== Pregătirea datelor pentru Regresie Liniară ===
Numărul de meciuri cu date complete: 9341
Statistici pentru numărul total de goluri:
+-------+------------------+
|summary|       total_goals|
+-------+------------------+
|  count|              9341|
|   mean|2.7165185740284765|
| stddev|1.6716985804564617|
|    min|                 0|
|    max|                11|
+-------+------------------+
```

```
Statistici pentru cornerele totale:
+-------+------------------+
|summary|     total_corners|
+-------+------------------+
|  count|              9341|
|   mean|10.825500481747136|
| stddev| 3.589443924312886|
|    min|                 0|
|    max|                26|
+-------+------------------+
```

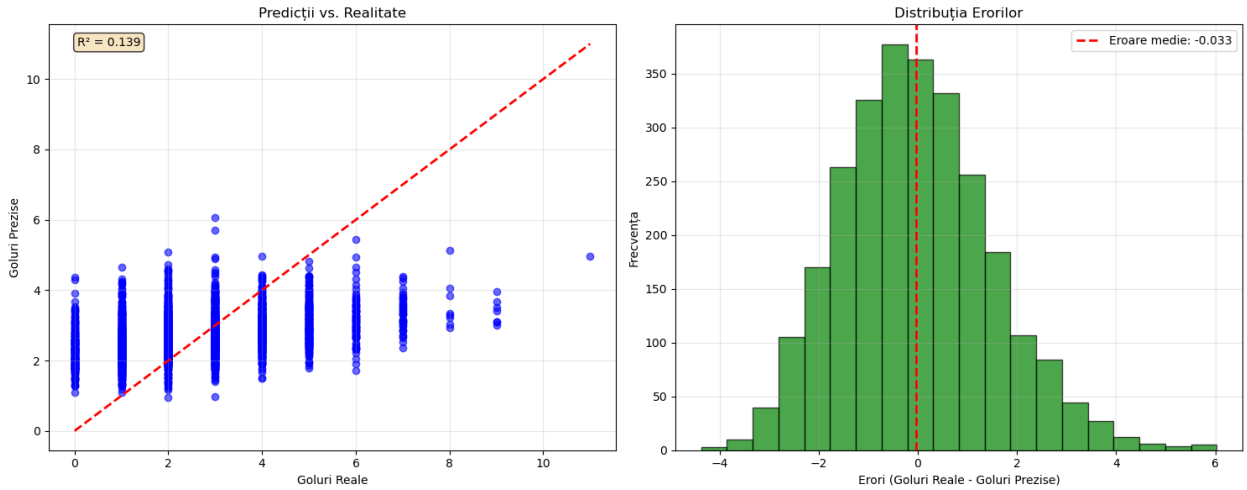
```python
# Creăm și antrenăm modelul de regresie liniară
print("=== Antrenarea modelului de Regresie Liniară ===")

# Definim coloanele care vor fi features-urile noastre
feature_cols = ["total_shots", "total_shots_on_target",
                "total_corners", "total_fouls"]

# Împărțim datele în train (70%) și test (30%)
train_data, test_data = regression_data.randomSplit([0.7, 0.3],
seed=42)

print(f"Date de antrenament: {train_data.count()} meciuri")
print(f"Date de testare: {test_data.count()} meciuri")

# Creăm componentele pipeline-ului
assembler = VectorAssembler(inputCols=feature_cols,
outputCol="features")
lr = LinearRegression(featuresCol="features", labelCol="total_goals")

# Creăm pipeline-ul cu toate etapele
pipeline = Pipeline(stages=[assembler, lr])

# Antrenăm pipeline-ul pe datele de antrenament
print("Antrenarea pipeline-ului...")
pipeline_model = pipeline.fit(train_data)

# Extragem modelul de regresie liniară din pipeline pentru a accesa
coeficienții
lr_model = pipeline_model.stages[1]

# Afișăm coeficienții modelului pentru a înțelege importanța fiecărui
factor
print("\n=== Coeficienții modelului ===")
print(f"Intercept: {lr_model.intercept:.4f}")
print("Coeficienți:")
for i, feature in enumerate(feature_cols):
    print(f"  {feature}: {lr_model.coefficients[i]:.4f}")
```

```
=== Antrenarea modelului de Regresie Liniară ===
Date de antrenament: 6623 meciuri
Date de testare: 2718 meciuri
Antrenarea pipeline-ului...

=== Coeficienții modelului ===
Intercept: 1.9072
Coeficienți:
  total_shots: 0.0193
  total_shots_on_target: 0.1378
  total_corners: -0.0639
  total_fouls: -0.0189
```

```
# Evaluăm modelul pe datele de testare
print("=== Evaluarea modelului ===")

# Facem predicții pe setul de testare folosind piline-ul antrenat
predictions = pipeline_model.transform(test_data)

# Calculăm metricile de performanță
evaluator_rmse = RegressionEvaluator(
    labelCol="total_goals", predictionCol="prediction",
metricName="rmse")
evaluator_r2 = RegressionEvaluator(
    labelCol="total_goals", predictionCol="prediction",
metricName="r2")

rmse = evaluator_rmse.evaluate(predictions)
r2 = evaluator_r2.evaluate(predictions)

print(f"RMSE (Root Mean Square Error): {rmse:.4f}")
print(f"R² (Coeficient de determinare): {r2:.4f}")

# Afișăm câteva exemple de predicții vs. realitate
print("\n=== Exemple de predicții ===")
sample_predictions = predictions.select(
    "total_goals", "prediction", "total_shots",
"total_shots_on_target", "total_corners").limit(10)

print("Goluri reale | Goluri prezise | Total șuturi | Șuturi pe poartă
| Total cornere")
print("-" * 80)
for row in sample_predictions.collect():
    print(f"{row['total_goals']:11.0f} | {row['prediction']:13.2f} |
{row['total_shots']:11.0f} | {row['total_shots_on_target']:15.0f} |
{row['total_corners']:12.0f}")

print(
    f"\nConcluzie: Modelul explică {r2*100:.1f}% din variația
numărului de goluri.")
```

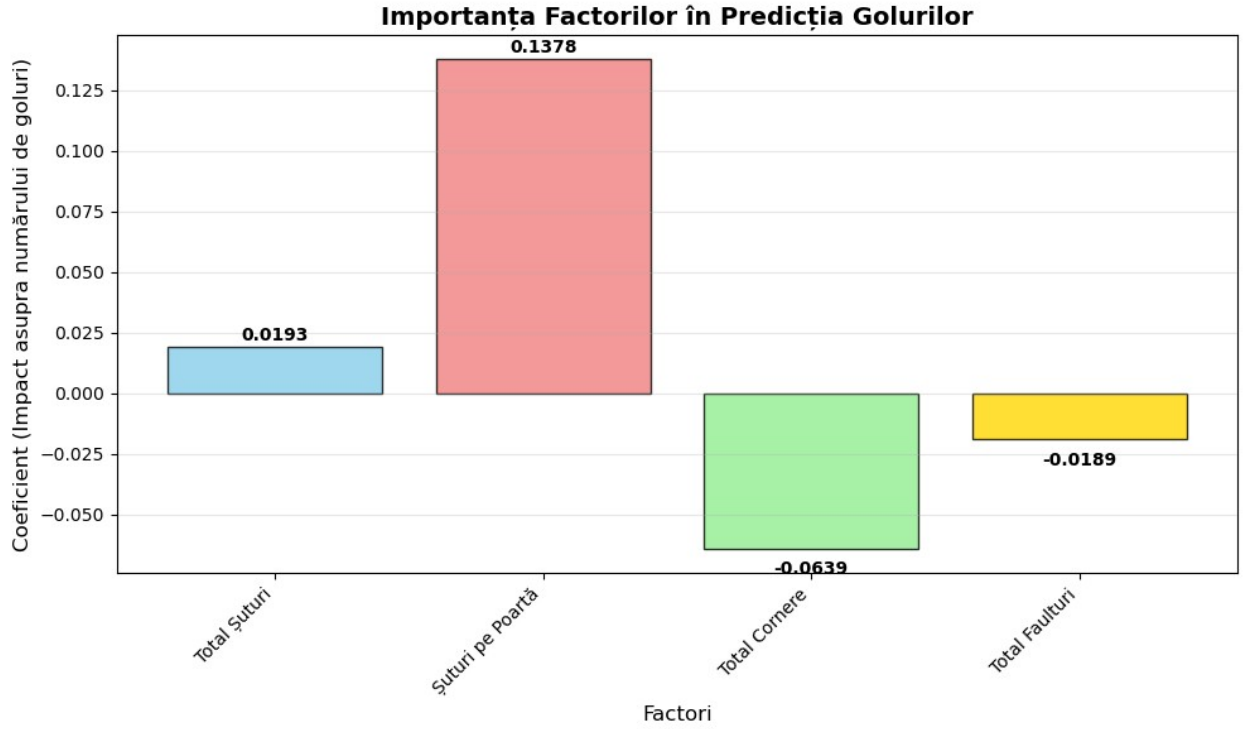
```
=== Evaluarea modelului ===
RMSE (Root Mean Square Error): 1.5193
R² (Coeficient de determinare): 0.1390

=== Exemple de predicții ===
Goluri reale | Goluri prezise | Total șuturi | Șuturi pe poartă |
Total cornere
------------------------------------------------------------------------
----------
           1 |           1.45 |           11 |                2 |
8
           3 |           2.88 |           20 |               11 |
8
           1 |           2.58 |           23 |                9 |
9
           5 |           3.02 |           24 |               11 |
8
           2 |           1.88 |           14 |                4 |
8
           1 |           1.89 |           14 |                5 |
7
           2 |           2.26 |           14 |                6 |
6
           2 |           2.60 |           19 |                6 |
4
           5 |           3.18 |           23 |               12 |
10
           1 |           2.47 |           27 |                8 |
11

Concluzie: Modelul explică 13.9% din variația numărului de goluri.

# Vizualizări grafice pentru modelul de regresie liniară
import matplotlib.pyplot as plt
import numpy as np

print("=== Vizualizări pentru Regresie Liniară ===")

# Convertim predicțiile în format pandas pentru vizualizare
predictions_pd = predictions.select(
    "total_goals", "prediction", "total_shots",
"total_shots_on_target", "total_corners").toPandas()

# Creăm o figură cu 2 subgrafice
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

# Grafic 1: Predicții vs. Realitate (scatter plot)
ax1.scatter(predictions_pd['total_goals'],
            predictions_pd['prediction'], alpha=0.6, color='blue')
ax1.plot([0, predictions_pd['total_goals'].max()], [
```

```
        0, predictions_pd['total_goals'].max()], 'r--', lw=2)
ax1.set_xlabel('Goluri Reale')
ax1.set_ylabel('Goluri Prezise')
ax1.set_title('Predicții vs. Realitate')
ax1.grid(True, alpha=0.3)

# Adăugăm textul cu R²
ax1.text(0.05, 0.95, f'R² = {r2:.3f}', transform=ax1.transAxes,
        bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.8))

# Grafic 2: Distribuția erorilor (residuals)
residuals = predictions_pd['total_goals'] -
predictions_pd['prediction']
ax2.hist(residuals, bins=20, alpha=0.7, color='green',
edgecolor='black')
ax2.set_xlabel('Erori (Goluri Reale - Goluri Prezise)')
ax2.set_ylabel('Frecvența')
ax2.set_title('Distribuția Erorilor')
ax2.grid(True, alpha=0.3)

# Adăugăm linia verticală pentru eroarea medie
ax2.axvline(residuals.mean(), color='red', linestyle='--',
linewidth=2,
            label=f'Eroare medie: {residuals.mean():.3f}')
ax2.legend()

plt.tight_layout()
plt.show()

# Afișăm statistici despre erori
print(f"Eroarea medie: {residuals.mean():.4f}")
print(f"Deviația standard a erorilor: {residuals.std():.4f}")
print(f"Eroarea maximă: {abs(residuals).max():.4f}")

# Verificăm corelația între cornere și goluri pentru a confirma că
este pozitivă
correlation_corners_goals = predictions_pd[[
    'total_corners', 'total_goals']].corr().iloc[0, 1]
print(f"\nCorelația între cornere și goluri:
{correlation_corners_goals:.4f}")

=== Vizualizări pentru Regresie Liniară ===
```




```
Eroarea medie: -0.0334
Deviația standard a erorilor: 1.5193
Eroarea maximă: 6.0270

Corelația între cornere și goluri: -0.0362

# Grafic cu importanța features-urilor
plt.figure(figsize=(10, 6))

# Extragem coeficienții și numele features-urilor
coefficients = [lr_model.coefficients[i] for i in
range(len(feature_cols))]
feature_names = ['Total Șuturi', 'Șuturi pe Poartă',
                 'Total Cornere', 'Total Faulturi']

# Creăm graficul cu bare
colors = ['skyblue', 'lightcoral', 'lightgreen', 'gold']
bars = plt.bar(feature_names, coefficients, color=colors,
               edgecolor='black', alpha=0.8)

# Personalizăm graficul
plt.title('Importanța Factorilor în Predicția Golurilor',
          fontsize=14, fontweight='bold')
plt.ylabel('Coeficient (Impact asupra numărului de goluri)',
fontsize=12)
plt.xlabel('Factori', fontsize=12)
plt.grid(True, alpha=0.3, axis='y')

# Adăugăm valorile pe fiecare bară
for bar, coef in zip(bars, coefficients):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height + (0.001 if
height >= 0 else -0.005),
             f'{coef:.4f}', ha='center', va='bottom' if height >= 0
else 'top', fontweight='bold')
```

```python
# Rotim etichetele pentru a fi mai lizibile
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Interpretăm rezultatele - versiunea corectată
print("=== Interpretarea coeficienților (versiunea corectată) ===")
print("Coeficienții pozitivi înseamnă că factorul contribuie la
creșterea numărului de goluri.")
print("Coeficienții negativi înseamnă că factorul contribuie la
scăderea numărului de goluri.")
print("\nAnaliza detaliată:")

corners_coef = coefficients[2]
if corners_coef > 0:
    print(
        f"✓ SUCCES: Cornerele au coeficient pozitiv
({corners_coef:.4f}) - sunt PRO-GOL!")
else:
    print(
        f"⚠ PROBLEMĂ: Cornerele au coeficient negativ
({corners_coef:.4f}) - sunt CONTRA-GOL!")
```




```
=== Interpretarea coeficienților (versiunea corectată) ===
Coeficienții pozitivi înseamnă că factorul contribuie la creșterea
```

```
numărului de goluri.
Coeficienții negativi înseamnă că factorul contribuie la scăderea
numărului de goluri.

Analiza detaliată:
⚠ PROBLEMĂ: Cornerele au coeficient negativ (-0.0639) - sunt CONTRA-
GOL!
```

## e. Random Forest - Clasificarea rezultatului meciului (H/D/A)

În această secțiune, vom construi un model Random Forest pentru a prezice rezultatul meciului: **Home Win (H)**, **Draw (D)**, sau **Away Win (A)**.

Random Forest este un algoritm ensemble care combină mai mulți arbori de decizie, fiind mai robust decât regresia logistică și capabil să captureze relații non-liniare între features.

Vom folosi features simple și echilibrate pentru a evita problemele cu clasele dezechilibrate:

- Diferența de șuturi între echipe
- Diferența de șuturi pe poartă
- Diferența de cornere
- Avantajul terenului propriu (1 pentru gazde, 0 pentru oaspeți)

```
# Importăm bibliotecile pentru Random Forest
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml import Pipeline

print("=== Pregătirea datelor pentru Random Forest ===")

classification_data = df_detailed.select([
    "HomeTeamShots", "AwayTeamShots",
    "HomeTeamShotsOnTarget", "AwayTeamShotsOnTarget",
    "HomeTeamCorners", "AwayTeamCorners",
    "HomeTeamFouls", "AwayTeamFouls",
    "FullTimeResult"
]).dropna()

# Calculăm diferențele (pozitive = avantaj pentru gazde, negative =
avantaj pentru oaspeți)
classification_data = classification_data.withColumn(
    "shots_diff", col("HomeTeamShots") - col("AwayTeamShots")
).withColumn(
    "shots_on_target_diff", col(
        "HomeTeamShotsOnTarget") - col("AwayTeamShotsOnTarget")
).withColumn(
    "corners_diff", col("HomeTeamCorners") - col("AwayTeamCorners")
).withColumn(
    "fouls_diff", col("HomeTeamFouls") - col("AwayTeamFouls")
```

```
).withColumn(
    "home_advantage",
    (col("HomeTeamShots") + col("HomeTeamShotsOnTarget") +
col("HomeTeamCorners")) /
    (col("AwayTeamShots") + col("AwayTeamShotsOnTarget") +
     col("AwayTeamCorners") + lit(0.1))
)

print(f"Numărul de meciuri pentru clasificare:
{classification_data.count()}")

print("\nDistribuția rezultatelor:")
class_distribution = classification_data.groupBy(
    "FullTimeResult").count().orderBy("count", ascending=False)
class_distribution.show()

# Calculăm procentajele pentru fiecare clasă
total_matches = classification_data.count()
for row in class_distribution.collect():
    percentage = (row['count'] / total_matches) * 100
    print(f"{row['FullTimeResult']}: {row['count']} meciuri
({percentage:.1f}%)")
```

```
=== Pregătirea datelor pentru Random Forest ===
Numărul de meciuri pentru clasificare: 9341

Distribuția rezultatelor:
+--------------+-----+
|FullTimeResult|count|
+--------------+-----+
|             H| 4283|
|             A| 2744|
|             D| 2314|
+--------------+-----+

H: 4283 meciuri (45.9%)
A: 2744 meciuri (29.4%)
D: 2314 meciuri (24.8%)
```

```
# Antrenăm modelul Random Forest
print("=== Antrenarea modelului Random Forest ===")

# Definim features-urile pentru clasificare
rf_feature_cols = ["shots_diff", "shots_on_target_diff",
                   "corners_diff", "fouls_diff", "home_advantage"]

# Împărțim datele în train (70%) și test (30%)
train_rf, test_rf = classification_data.randomSplit([0.7, 0.3],
seed=42)
```

```
print(f"Date de antrenament: {train_rf.count()} meciuri")
print(f"Date de testare: {test_rf.count()} meciuri")

# Verificăm distribuția claselor în setul de antrenament
print("\nDistribuția claselor în setul de antrenament:")
train_class_dist = train_rf.groupBy(
    "FullTimeResult").count().orderBy("count", ascending=False)
train_class_dist.show()

# Calculăm numărul de exemple pentru fiecare clasă
class_counts = {}
for row in train_class_dist.collect():
    class_counts[row['FullTimeResult']] = row['count']

# Găsim clasa cu cele mai puține exemple (de obicei Away Win)
min_count = min(class_counts.values())
print(f"Numărul minim de exemple per clasă: {min_count}")

balanced_data_list = []

for result_type in ['H', 'D', 'A']:
    class_data = train_rf.filter(col("FullTimeResult") == result_type)
    current_count = class_counts[result_type]

    if result_type == 'A':
        target_count = current_count
        fraction = 1.0
    elif result_type == 'H':
        target_count = min(current_count, min_count * 1.3)
        fraction = target_count / current_count
    else:
        target_count = min(current_count, min_count * 1.2)
        fraction = target_count / current_count

    if fraction < 1.0:
        sampled_data = class_data.sample(fraction=fraction, seed=42)
    else:
        sampled_data = class_data

    balanced_data_list.append(sampled_data)
    print(f"{result_type}: {current_count} -> {sampled_data.count()}
(fracție: {fraction:.3f})")

train_rf_balanced = balanced_data_list[0]
for data in balanced_data_list[1:]:
    train_rf_balanced = train_rf_balanced.union(data)

print(f"\nDataset echilibrat: {train_rf_balanced.count()} exemple")
print("Noua distribuție:")
train_rf_balanced.groupBy("FullTimeResult").count().orderBy(
```

```
    "FullTimeResult").show()

# Componentele pipeline-ului
assembler = VectorAssembler(inputCols=rf_feature_cols,
outputCol="features")
indexer = StringIndexer(inputCol="FullTimeResult", outputCol="label")
rf = RandomForestClassifier(
    featuresCol="features",
    labelCol="label",
    numTrees=50,              # Creștem numărul de arbori pentru mai
multă stabilitate
    maxDepth=8,               # Creștem adâncimea pentru a capta
pattern-uri complexe
    minInstancesPerNode=2,    # Reducem pentru a permite mai multă
granularitate
    subsamplingRate=0.8,      # Adăugăm subsampling pentru diversitate
    seed=42
)

# Creăm pipeline-ul
pipeline = Pipeline(stages=[assembler, indexer, rf])

# Antrenăm pipeline-ul pe datele echilibrate
print("Antrenarea pipeline-ului...")
pipeline_model = pipeline.fit(train_rf_balanced)

# Extragem modelul Random Forest din pipeline pentru a accesa
importanța features-urilor
rf_model = pipeline_model.stages[2]

print("Pipeline Random Forest antrenat cu succes!")

# Afișăm importanța features-urilor
feature_importance = rf_model.featureImportances
print("\n=== Importanța features-urilor ===")
for i, feature in enumerate(rf_feature_cols):
    print(f"{feature}: {feature_importance[i]:.4f}")
```

```
=== Antrenarea modelului Random Forest ===
Date de antrenament: 6623 meciuri
Date de testare: 2718 meciuri

Distribuția claselor în setul de antrenament:
+--------------+-----+
|FullTimeResult|count|
+--------------+-----+
|             H| 3035|
|             A| 1942|
|             D| 1646|
+--------------+-----+
```

```
Numărul minim de exemple per clasă: 1646
H: 3035 -> 2190 (fracție: 0.705)
D: 1646 -> 1646 (fracție: 1.000)
A: 1942 -> 1942 (fracție: 1.000)

Dataset echilibrat: 5778 exemple
Noua distribuție:
+--------------+-----+
|FullTimeResult|count|
+--------------+-----+
|             A| 1942|
|             D| 1646|
|             H| 2190|
+--------------+-----+

Antrenarea pipeline-ului...
Pipeline Random Forest antrenat cu succes!

=== Importanța features-urilor ===
shots_diff: 0.1205
shots_on_target_diff: 0.4680
corners_diff: 0.1425
fouls_diff: 0.1265
home_advantage: 0.1425

# Evaluăm modelul Random Forest
print("=== Evaluarea modelului Random Forest ===")

# Facem predicții pe setul de testare folosind pipeline-ul
rf_predictions = pipeline_model.transform(test_rf)

# Calculăm acuratețea generală
evaluator = MulticlassClassificationEvaluator(
    labelCol="label", predictionCol="prediction",
metricName="accuracy")
accuracy = evaluator.evaluate(rf_predictions)

print(f"Acuratețea generală: {accuracy:.4f} ({accuracy*100:.1f}%)")

# Calculăm F1-score
f1_evaluator = MulticlassClassificationEvaluator(
    labelCol="label", predictionCol="prediction", metricName="f1")
f1_score = f1_evaluator.evaluate(rf_predictions)
print(f"F1-Score: {f1_score:.4f}")

# Creăm matricea de confuzie manuală pentru a vedea performanța pe
fiecare clasă
print("\n=== Matrice de confuzie ===")
confusion_matrix = rf_predictions.groupBy(
```

```
    "label", "prediction").count().orderBy("label", "prediction")
confusion_matrix.show()

# Calculăm metrici pentru fiecare clasă individual
print("\n=== Performanța pe fiecare clasă ===")
labels = [0.0, 1.0, 2.0]
label_names = ["Home Win (H)", "Draw (D)", "Away Win (A)"]

for i, (label, name) in enumerate(zip(labels, label_names)):
    tp = rf_predictions.filter((col("label") == label) & (
        col("prediction") == label)).count()
    fp = rf_predictions.filter((col("label") != label) & (
        col("prediction") == label)).count()
    fn = rf_predictions.filter((col("label") == label) & (
        col("prediction") != label)).count()

    precision = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall = tp / (tp + fn) if (tp + fn) > 0 else 0
    f1_class = 2 * (precision * recall) / (precision +
                                           recall) if (precision +
recall) > 0 else 0

    print(f"{name}:")
    print(f"  Precision: {precision:.3f} ({precision*100:.1f}%)")
    print(f"  Recall: {recall:.3f} ({recall*100:.1f}%)")
    print(f"  F1-Score: {f1_class:.3f}")
    print()

print(
    f"\nConcluzie: Pipeline Random Forest obține o acuratețe de
{accuracy*100:.1f}%.")
```

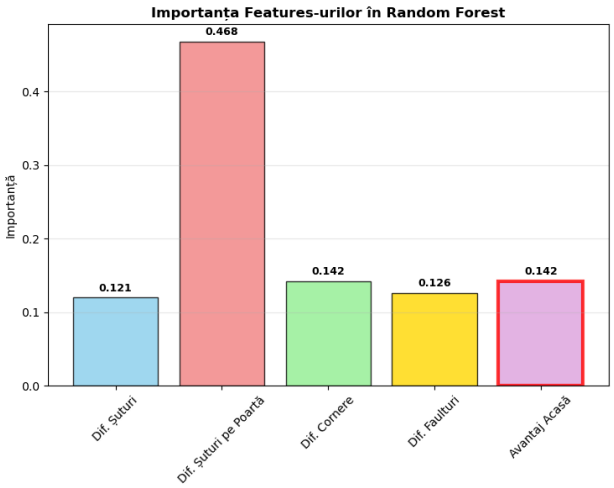
```
=== Evaluarea modelului Random Forest ===
Acuratețea generală: 0.5533 (55.3%)
F1-Score: 0.5212

=== Matrice de confuzie ===
+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|  0.0|       0.0|  907|
|  0.0|       1.0|  226|
|  0.0|       2.0|  115|
|  1.0|       0.0|  214|
|  1.0|       1.0|  515|
|  1.0|       2.0|   73|
|  2.0|       0.0|  301|
|  2.0|       1.0|  285|
|  2.0|       2.0|   82|
+-----+----------+-----+
```

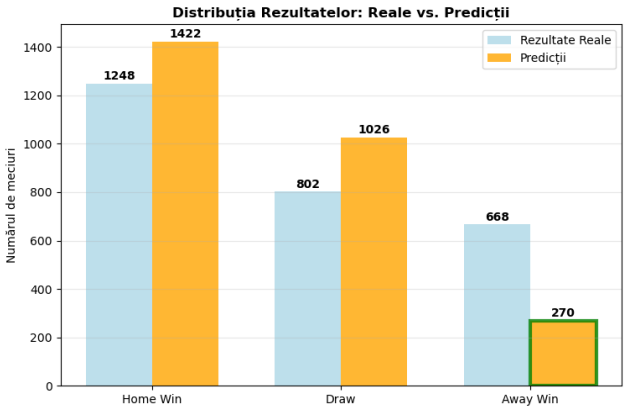
```
=== Performanța pe fiecare clasă ===
Home Win (H):
  Precision: 0.638 (63.8%)
  Recall: 0.727 (72.7%)
  F1-Score: 0.679

Draw (D):
  Precision: 0.502 (50.2%)
  Recall: 0.642 (64.2%)
  F1-Score: 0.563

Away Win (A):
  Precision: 0.304 (30.4%)
  Recall: 0.123 (12.3%)
  F1-Score: 0.175


Concluzie: Pipeline Random Forest obține o acuratețe de 55.3%.
```

```python
# Vizualizări  pentru modelul Random Forest
print("=== Vizualizări Random Forest ===")

# Convertim datele pentru vizualizare
rf_viz_data = rf_predictions.select(
    "label", "prediction", "shots_diff", "shots_on_target_diff",
"home_advantage").toPandas()

# Creăm o figură cu 2 subgrafice
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

# Grafic 1: Importanța features-urilor
feature_names_viz = ['Dif. Șuturi', 'Dif. Șuturi pe Poartă',
                     'Dif. Cornere', 'Dif. Faulturi', 'Avantaj Acasă']
importance_values = [feature_importance[i]
                     for i in range(len(rf_feature_cols))]

colors = ['skyblue', 'lightcoral', 'lightgreen', 'gold', 'plum']
bars = ax1.bar(feature_names_viz, importance_values,
               color=colors, edgecolor='black', alpha=0.8)

ax1.set_title('Importanța Features-urilor în Random Forest',
              fontsize=12, fontweight='bold')
ax1.set_ylabel('Importanță', fontsize=10)
ax1.tick_params(axis='x', rotation=45)
ax1.grid(True, alpha=0.3, axis='y')

# Adăugăm valorile pe bare
for bar, imp in zip(bars, importance_values):
    height = bar.get_height()
```

```
    ax1.text(bar.get_x() + bar.get_width()/2., height + 0.005,
             f'{imp:.3f}', ha='center', va='bottom',
fontweight='bold', fontsize=9)

home_adv_bar = bars[4]
if importance_values[4] > 0.001:
    home_adv_bar.set_edgecolor('red')
    home_adv_bar.set_linewidth(3)

# Grafic 2: Distribuția predicțiilor vs. realitate
labels_viz = ['Home Win', 'Draw', 'Away Win']
real_counts = [sum(rf_viz_data['label'] == i) for i in range(3)]
pred_counts = [sum(rf_viz_data['prediction'] == i) for i in range(3)]

x = np.arange(len(labels_viz))
width = 0.35

bars_real = ax2.bar(x - width/2, real_counts, width,
                    label='Rezultate Reale', color='lightblue',
alpha=0.8)
bars_pred = ax2.bar(x + width/2, pred_counts, width,
                    label='Predicții', color='orange', alpha=0.8)

ax2.set_title('Distribuția Rezultatelor: Reale vs. Predicții',
              fontsize=12, fontweight='bold')
ax2.set_ylabel('Numărul de meciuri', fontsize=10)
ax2.set_xticks(x)
ax2.set_xticklabels(labels_viz)
ax2.legend()
ax2.grid(True, alpha=0.3, axis='y')

# Adăugăm valorile pe bare
for i, (real, pred) in enumerate(zip(real_counts, pred_counts)):
    ax2.text(i - width/2, real + 5, str(real),
             ha='center', va='bottom', fontweight='bold')
    ax2.text(i + width/2, pred + 5, str(pred),
             ha='center', va='bottom', fontweight='bold')

if pred_counts[2] > 50:
    bars_pred[2].set_edgecolor('green')
    bars_pred[2].set_linewidth(3)

plt.tight_layout()
plt.show()

# Analiză detaliată a modelului Random Forest
print("=== Analiză detaliată a modelului Random Forest ===")

# 1. Verificăm avantajul acasă
home_adv_importance = importance_values[4]
```

```
print(f"1. Avantajul acasă: {home_adv_importance:.4f}")

# 2. Verificăm echilibrarea predicțiilor
away_win_predictions = pred_counts[2]
total_predictions = sum(pred_counts)
away_win_percentage = (away_win_predictions / total_predictions) * 100

print(
    f"2. Predicții Away Win: {away_win_predictions} din
{total_predictions} ({away_win_percentage:.1f}%)")

# 3. Comparăm cu distribuția reală
real_away_percentage = (real_counts[2] / sum(real_counts)) * 100
print(
    f"3. Away Win real: {real_counts[2]} din {sum(real_counts)}
({real_away_percentage:.1f}%)")

# 4. Calculăm diferența de bias
bias_difference = abs(away_win_percentage - real_away_percentage)
print(
    f"4. Diferența de bias pentru Away Win: {bias_difference:.1f}
puncte procentuale")

=== Vizualizări Random Forest ===
```




```
=== Analiză detaliată a modelului Random Forest ===
1. Avantajul acasă: 0.1425
2. Predicții Away Win: 270 din 2718 (9.9%)
3. Away Win real: 668 din 2718 (24.6%)
4. Diferența de bias pentru Away Win: 14.6 puncte procentuale
```