ativ_1

April 10, 2023

[1]:

```
# pyright: reportUnusedExpression=false
```

[2]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn
from scipy.stats import poisson, skellam
```

[3]:

```
from math import ceil
from typing import Tuple
```

1 [CDAF] Atividade 1

1.1 Nome e matrícula

Nome: Igor Lacerda Faria da Silva Matrícula: 2020041973

1.2 Introdução

Nesta atividade, vamos revisar os conceitos aprendidos em sala de aula sobre aleatoriedade e previsão, trabalhando em cima do dataset do Soccer Prediction Challenge, disponível no Moodle.

1.3 Questão 1

- Carregue o dataset 'TrainingSet_2023_02_08'
- Crie um histograma para a quantidade de gols marcados por jogo do time da casa, do time fora, de gols totais e da diferença de gols por partida.
- Caso hajam instâncias com valores nitidamente errados, destaque-os e remova-os antes de gerar os histogramas.
- Calcule o mínimo, o máximo e a média de cada um dos 4 histogramas solicitados acima.

[4]:

```
df = pd.read_excel("TrainingSet_2023_02_08.xlsx")
```

[5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299214 entries, 0 to 299213
```

```
Data columns (total 9 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   Sea     299214 non-null  object
 1   Lge     299214 non-null  object
 2   Date    299214 non-null  object
 3   HT      299214 non-null  object
 4   AT      299214 non-null  object
 5   HS      299214 non-null  int64 
 6   AS      299214 non-null  int64 
 7   GD      299214 non-null  int64 
 8   WDL     299214 non-null  object
dtypes: int64(3), object(6)
memory usage: 20.5+ MB
```

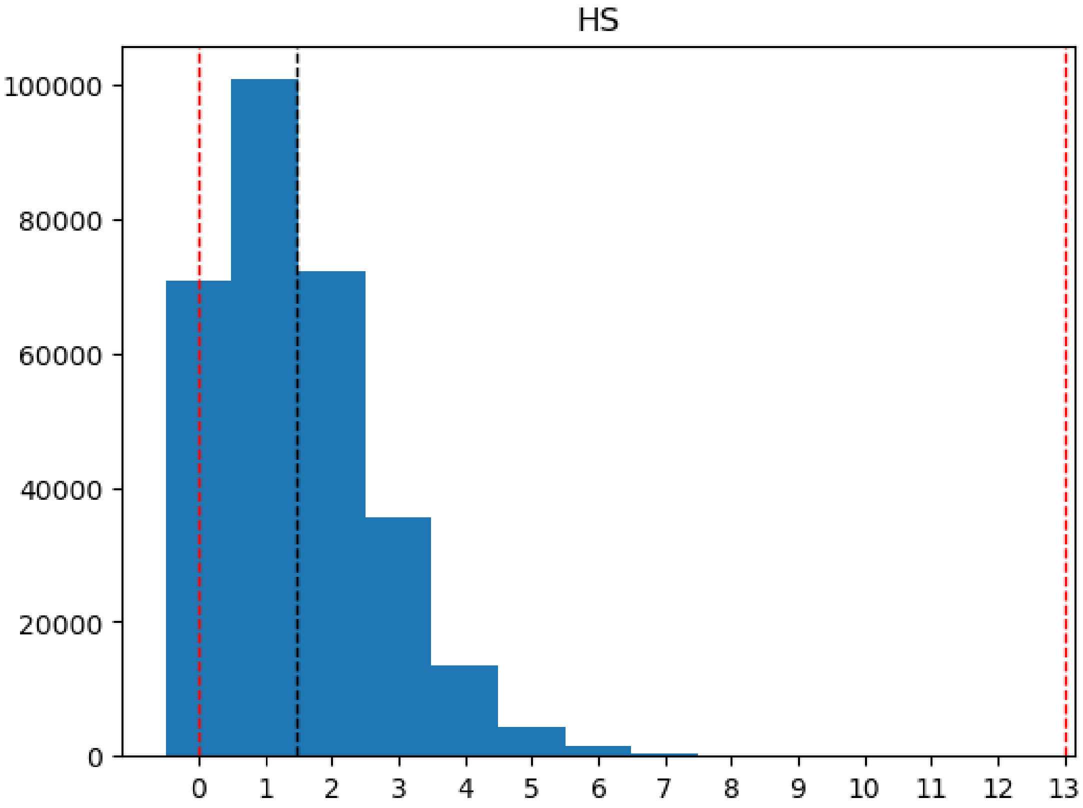
[6]:
```python
df.head()
```

[6]:
```
     Sea   Lge        Date              HT              AT  HS  AS  GD WDL
0  00-01  GER1  11/08/2000        Dortmund   Hansa Rostock   1   0   1   W
1  00-01  GER1  12/08/2000   Bayern Munich   Hertha Berlin   4   1   3   W
2  00-01  GER1  12/08/2000        Freiburg   VfB Stuttgart   4   0   4   W
3  00-01  GER1  12/08/2000     Hamburger SV     Munich 1860   2   2   0   D
4  00-01  GER1  12/08/2000  Kaiserslautern          Bochum   0   1  -1   L
```

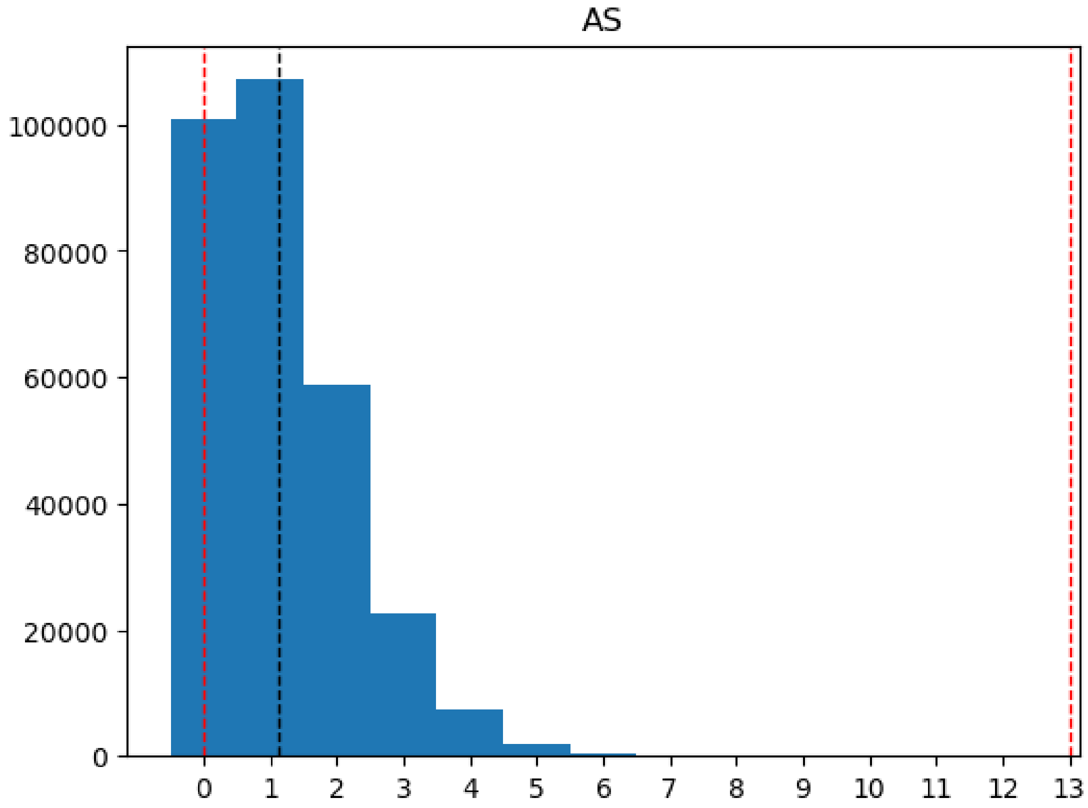
[7]:
```python
def histogram(column: pd.Series):
    mean, min, max = column.mean(), column.min(), column.max()
    bins = np.arange(min, max + 1, 1)
    plt.hist(column, align="left", bins=bins)
    plt.xticks(bins)
    plt.title(column.name)
    plt.axvline(mean, color="k", linestyle="dashed", linewidth=1)
    plt.axvline(min, color="r", linestyle="dashed", linewidth=1)
    plt.axvline(max, color="r", linestyle="dashed", linewidth=1)
    plt.show()
    print(f"Média: {mean}   Mínimo: {min}   Máximo: {max}")
```

[8]:
```python
# Tirando -1, como alguém faz gol negativo?
df["HS"] = df.query("HS > -1")["HS"]
df["AS"] = df.query("AS > -1")["AS"]
```

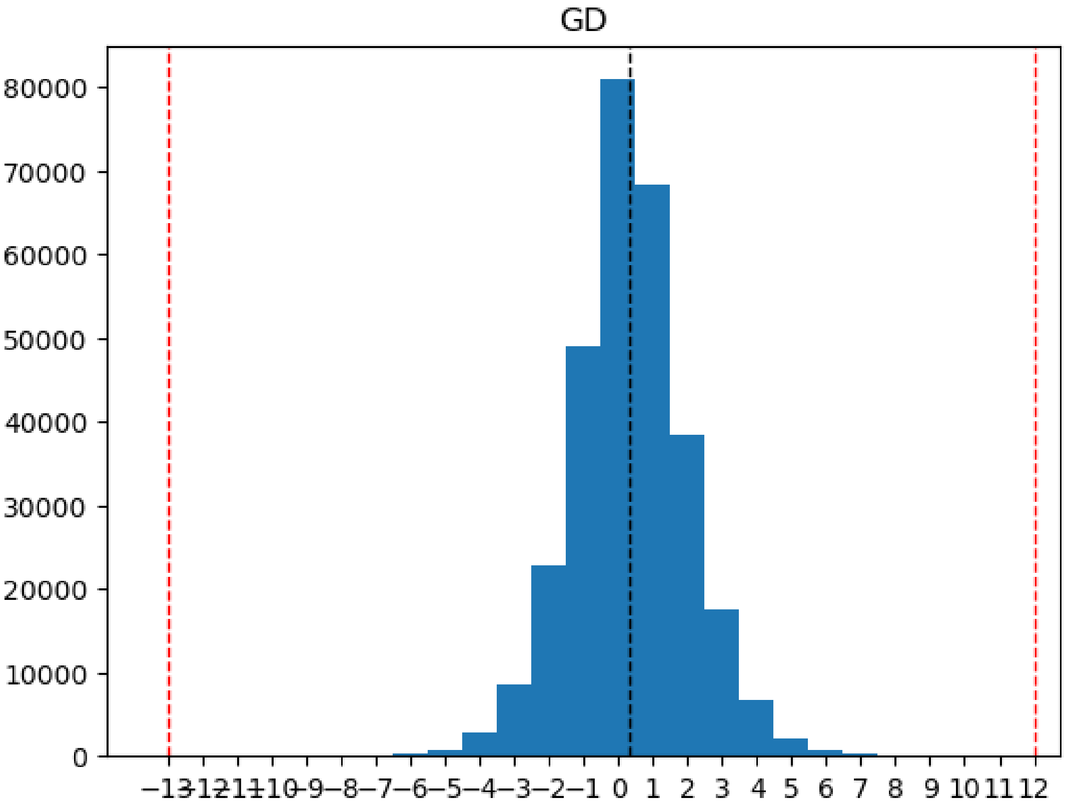
[9]:
```python
# Home, Adversary, Diference
for column in ["HS", "AS", "GD"]:
    histogram(df[column])
```



```
Média: 1.471700194179949   Mínimo: 0.0   Máximo: 13.0
```



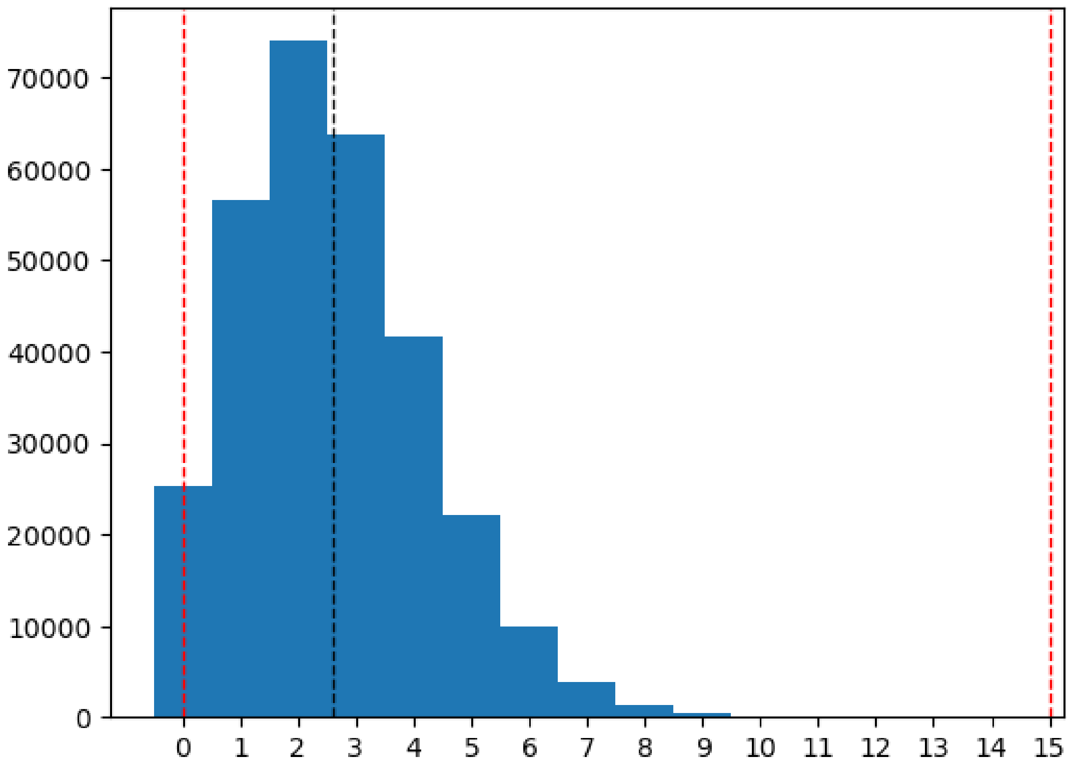
```
Média: 1.1214410090673013   Mínimo: 0.0   Máximo: 13.0
```



```
Média: 0.3502509909295688   Mínimo: -13   Máximo: 12
```

[10]:

```python
# Total
histogram(abs(df["AS"]) + abs(df["HS"]))
```



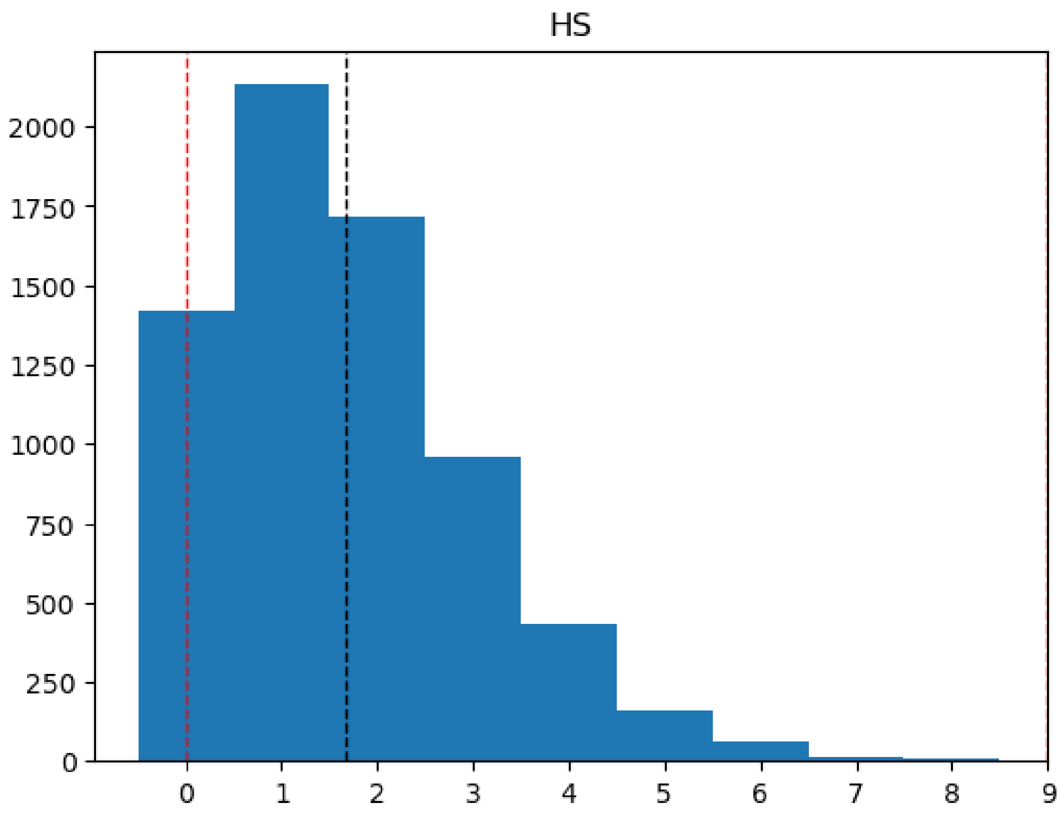
```
Média: 2.5931412032472503   Mínimo: 0.0   Máximo: 15.0
```

1.4 Questão 2

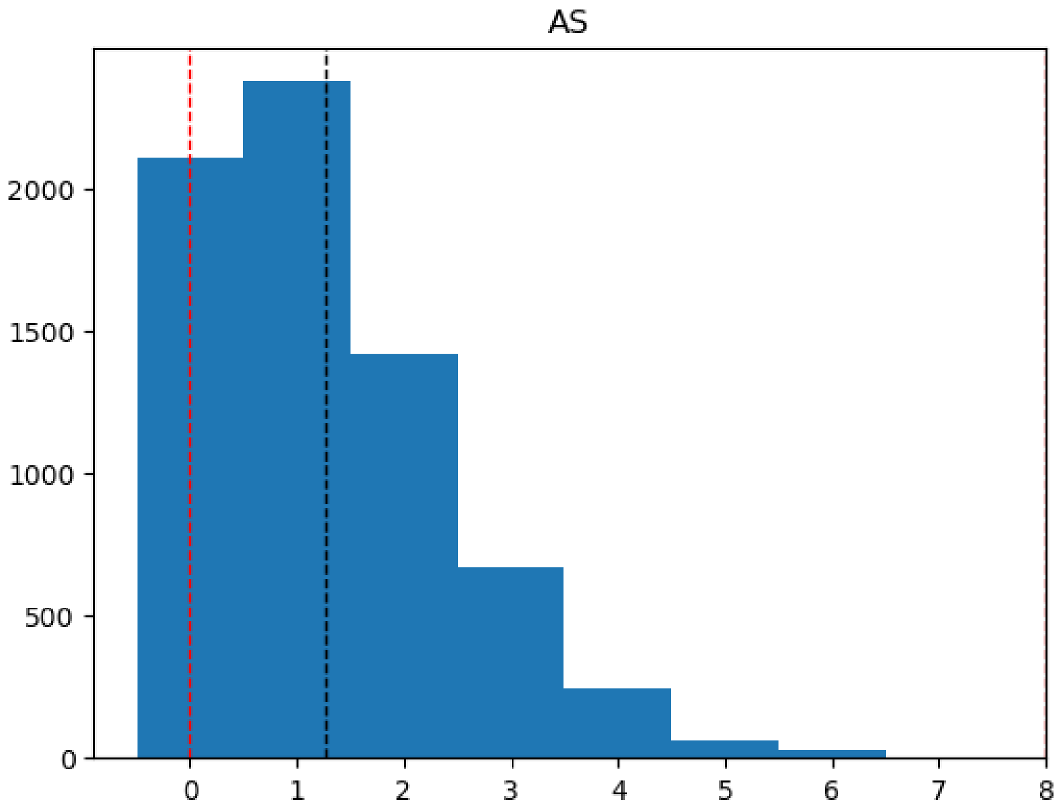
- Escolha uma temporada que já terminou, de alguma das ligas presentes no dataset.
- Realize os mesmos histogramas da questão 1, mas agora para a temporada escolhida.
- Quais as diferenças entre os histogramas da questão 1 e da questão 2? O que isso pode indicar sobre a qualidade ofensiva da liga escolhida vs. o todo?

[11]:
```
LEAGUE = "GER1"
SEASON = "00-01"
```

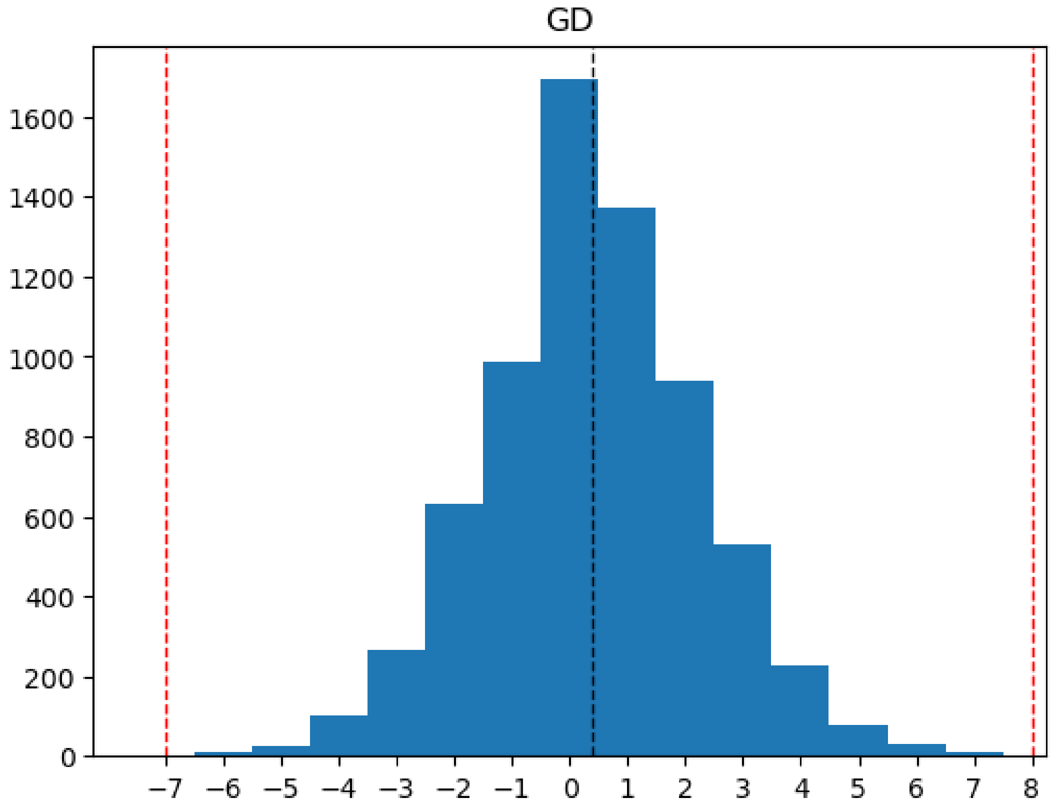
[12]:
```
for column in ["HS", "AS", "GD"]:
    histogram(df.query(f"Lge == '{LEAGUE}'")[column])
```



```
Média: 1.663479646530494   Mínimo: 0.0   Máximo: 9.0
```



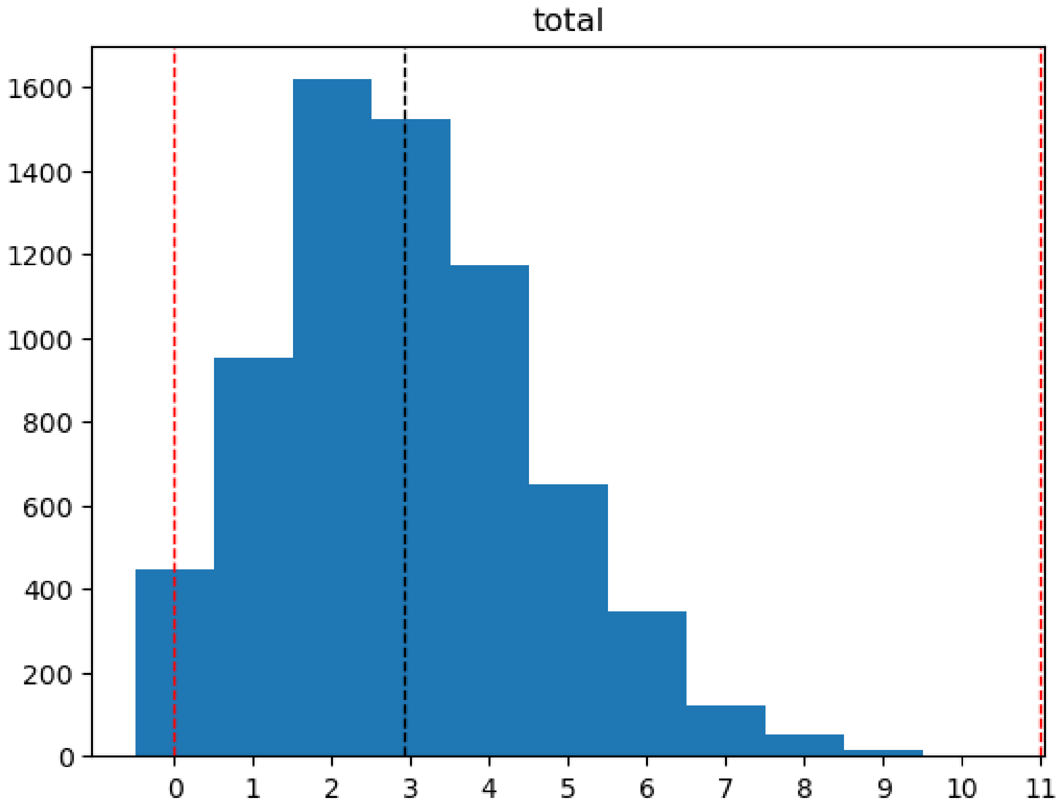
```
Média: 1.2623497030276691   Mínimo: 0.0   Máximo: 8.0
```



```
Média: 0.4011299435028249   Mínimo: -7   Máximo: 8
```

[13]:

```
df["total"] = abs(df["AS"]) + abs(df["HS"])
histogram(df.query(f"Lge == '{LEAGUE}'")["total"])
```



```
Média: 2.925829349558163   Mínimo: 0.0   Máximo: 11.0
```

Em média, na liga GER1, são feitos mais gols, em comparação com o total.

1.5 Questão 3

- À partir dos dados do campeonato em selecionado, crie um dataframe que corresponda à tabela de classificação ao fim da temporada contendo o nome dos times, nº de pontos, jogos, vitórias, empates, derrotas, gols pró, gols contra e saldo de gols. Ordena a classificação por pontos, vitórias, saldo de gols e gols pró.
- Faça o mesmo para apenas para a primeira metade de jogos.

[14]:

```python
def get_table(df: pd.DataFrame, league: str, season: str, ratio: float) -> pd.DataFrame:
    df_league = df
    if "Lge" in df:
        df_league = df.query(f"Lge == '{league}' and Sea == '{season}'")
    teams = df_league["HT"].unique()
    data: list[list] = []
    for team in teams:
        df_team: pd.DataFrame = df_league.query(f"HT == '{team}' or AT == '{team}'")
        if "Date" in df_team:
            df_team.sort_values(by=["Date"])
        played = ceil(ratio * len(df_team))

        # Ajusta dataframe para representar a porção mais recente dos jogos␣
↪jogados pelo time
        df_team = df_team.head(played)

        won = len(
            df_team.query(
                f"(HT == '{team}' and WDL == 'W') or (AT == '{team}' and WDL ==␣
↪'L')"
            )
        )
        drawn = len(
            df_team.query(
                f"(HT == '{team}' and WDL == 'D') or (AT == '{team}' and WDL ==␣
↪'D')"
            )
        )
        lost = len(
            df_team.query(
                f"(HT == '{team}' and WDL == 'L') or (AT == '{team}' and WDL ==␣
↪'W')"
            )
        )

        assert played == won + drawn + lost

        points = 3 * won + drawn

        gf = sum(df_team.query(f"HT == '{team}'")["HS"]) + sum(
            df_team.query(f"AT == '{team}'")["AS"]
        )
        ga = sum(df_team.query(f"HT == '{team}'")["AS"]) + sum(
            df_team.query(f"AT == '{team}'")["HS"]
        )
        gd = gf - ga

        data.append([team, played, won, drawn, lost, gf, ga, gd, points])

    df_table = pd.DataFrame(
        data,
        columns=["Team", "Matches", "Won", "Drawn", "Lost", "GF", "GA", "GD",␣
↪"Points"],
    )
    df_table = df_table.sort_values(by=["Points", "Won", "GD", "GF"],␣
↪ascending=False)
    return df_table
```

```
[15]: # Primeiro: Bayern Munich, Quarto: Leverkusen
      df_full_table = get_table(df, LEAGUE, SEASON, 1)
      df_full_table
```

```
[15]:                  Team  Matches  Won  Drawn  Lost    GF    GA    GD  Points
      1       Bayern Munich       34   19      6     9  62.0  37.0  25.0      63
      8          Schalke 04       34   18      8     8  65.0  35.0  30.0      62
      0            Dortmund       34   16     10     8  62.0  42.0  20.0      58
      5          Leverkusen       34   17      6    11  54.0  40.0  14.0      57
      12      Hertha Berlin       34   18      2    14  58.0  52.0   6.0      56
      2            Freiburg       34   15     10     9  54.0  37.0  17.0      55
      6       Werder Bremen       34   15      8    11  53.0  48.0   5.0      53
      4      Kaiserslautern       34   15      5    14  49.0  54.0  -5.0      50
      15          Wolfsburg       34   12     11    11  60.0  45.0  15.0      47
      16            FC Koln       34   12     10    12  59.0  52.0   7.0      46
      10        Munich 1860       34   12      8    14  43.0  55.0 -12.0      44
      17      Hansa Rostock       34   12      7    15  34.0  47.0 -13.0      43
      3         Hamburger SV       34   10     11    13  58.0  58.0   0.0      41
      9      Energie Cottbus       34   12      3    19  38.0  52.0 -14.0      39
      13       VfB Stuttgart       34    9     11    14  42.0  49.0  -7.0      38
      7   Eintracht Frankfurt       34   10      5    19  41.0  68.0 -27.0      35
      14         Unterhaching       34    8     11    15  35.0  59.0 -24.0      35
      11              Bochum       34    7      6    21  30.0  67.0 -37.0      27
```

```
[16]: # Apenas para a primeira metade dos jogos
      df_half = get_table(df, LEAGUE, SEASON, 0.5)
      df_half
```

```
[16]:                  Team  Matches  Won  Drawn  Lost    GF    GA    GD  Points
      8          Schalke 04       17   10      3     4  35.0  17.0  18.0      33
      5          Leverkusen       17    9      4     4  28.0  19.0   9.0      31
      1       Bayern Munich       17    9      3     5  34.0  19.0  15.0      30
      0            Dortmund       17    9      3     5  27.0  24.0   3.0      30
      12      Hertha Berlin       17    9      1     7  36.0  31.0   5.0      28
      4      Kaiserslautern       17    8      3     6  25.0  23.0   2.0      27
      16            FC Koln       17    7      4     6  31.0  27.0   4.0      25
      15          Wolfsburg       17    6      6     5  34.0  24.0  10.0      24
      2            Freiburg       17    6      5     6  25.0  21.0   4.0      23
      3         Hamburger SV       17    6      3     8  34.0  32.0   2.0      21
      17      Hansa Rostock       17    6      3     8  18.0  27.0  -9.0      21
      10        Munich 1860       17    5      6     6  22.0  30.0  -8.0      21
      7   Eintracht Frankfurt       17    6      2     9  22.0  29.0  -7.0      20
      6       Werder Bremen       17    5      5     7  23.0  29.0  -6.0      20
```

```
14         Unterhaching       17    5       5      7  19.0  27.0  -8.0       20
11               Bochum       17    5       3      9  16.0  33.0 -17.0       18
9        Energie Cottbus      17    5       2     10  19.0  31.0 -12.0       17
13          VfB Stuttgart     17    4       5      8  25.0  30.0  -5.0       17
```

[17]:

```python
df_half.head()
```

[17]:

```
              Team  Matches  Won  Drawn  Lost    GF    GA    GD  Points
8       Schalke 04       17   10      3     4  35.0  17.0  18.0      33
5       Leverkusen       17    9      4     4  28.0  19.0   9.0      31
1    Bayern Munich       17    9      3     5  34.0  19.0  15.0      30
0         Dortmund       17    9      3     5  27.0  24.0   3.0      30
12   Hertha Berlin       17    9      1     7  36.0  31.0   5.0      28
```

1.6 Questão 4

- Utilizando os jogos da liga escolhida, use regressão de Poisson para criar um modelo de previsão de resultados, como visto nos slides em sala e no Soccermatics. – https://soccermatics.readthedocs.io/en/latest/gallery/lesson5/plot_SimulateMatches.html
- Dê print no sumário do ajuste
- Simule a partida entre o 1º e o 4º colocado, onde o 1º joga em casa. Primeiro, apresente a quantidade esperada de gols de cada time. Em seguida, apresente um histograma com a probabilidade de diferentes placares entre os times.

[18]:

```python
df_sample = df.query(f"Lge == '{LEAGUE}' and Sea == '{SEASON}'")
df_sample
```

[18]:

```
       Sea   Lge        Date             HT            AT   HS   AS  GD WDL  \
0    00-01  GER1  11/08/2000       Dortmund  Hansa Rostock  1.0  0.0   1   W
1    00-01  GER1  12/08/2000  Bayern Munich  Hertha Berlin  4.0  1.0   3   W
2    00-01  GER1  12/08/2000       Freiburg  VfB Stuttgart  4.0  0.0   4   W
3    00-01  GER1  12/08/2000    Hamburger SV    Munich 1860  2.0  2.0   0   D
4    00-01  GER1  12/08/2000 Kaiserslautern         Bochum  0.0  1.0  -1   L
..     …     …          …              …       …   …   …  ..  ..
301  00-01  GER1  19/05/2001    Hamburger SV  Bayern Munich  1.0  1.0   0   D
302  00-01  GER1  19/05/2001 Kaiserslautern  Hertha Berlin  0.0  1.0  -1   L
303  00-01  GER1  19/05/2001      Leverkusen         Bochum  1.0  0.0   1   W
304  00-01  GER1  19/05/2001      Schalke 04   Unterhaching  5.0  3.0   2   W
305  00-01  GER1  19/05/2001   Werder Bremen  Hansa Rostock  3.0  0.0   3   W

     total
0      1.0
1      5.0
2      4.0
3      4.0
4      1.0
..     …
301    2.0
```

```
302    1.0
303    1.0
304    8.0
305    3.0

[306 rows x 10 columns]
```

[19]:

```python
goal_model_data = pd.concat(
    [
        df_sample[["HT", "AT", "HS"]]
        .assign(home=1)
        .rename(columns={"HT": "team", "AT": "opponent", "HS": "goals"}),
        df_sample[["AT", "HT", "AS"]]
        .assign(home=0)
        .rename(columns={"AT": "team", "HT": "opponent", "AS": "goals"}),
    ]
)

poisson_model = smf.glm(
    formula="goals ~ home + team + opponent",
    data=goal_model_data,
    family=sm.families.Poisson(),
).fit()
poisson_model.summary()
```

[19]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                  goals   No. Observations:                  612
Model:                            GLM   Df Residuals:                      576
Model Family:                 Poisson   Df Model:                           35
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -918.48
Date:                Mon, 10 Apr 2023   Deviance:                       724.18
Time:                        00:28:09   Pearson chi2:                     634.
No. Iterations:                     5   Pseudo R-squ. (CS):             0.1670
Covariance Type:            nonrobust
================================================================================
===================
                                      coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
-------------------
Intercept                           0.0696      0.215      0.324      0.746
-0.351       0.490
team[T.Bochum]                     -0.6928      0.223     -3.110      0.002
```

```
-1.129      -0.256
team[T.Dortmund]                    0.0059       0.180       0.033       0.974
-0.347       0.359
team[T.Eintracht Frankfurt]        -0.3781       0.202      -1.875       0.061
-0.773       0.017
team[T.Energie Cottbus]            -0.4734       0.206      -2.294       0.022
-0.878      -0.069
team[T.FC Koln]                    -0.0319       0.182      -0.175       0.861
-0.389       0.325
team[T.Freiburg]                   -0.1386       0.186      -0.743       0.457
-0.504       0.227
team[T.Hamburger SV]               -0.0418       0.183      -0.228       0.819
-0.401       0.317
team[T.Hansa Rostock]              -0.5907       0.214      -2.764       0.006
-1.009      -0.172
team[T.Hertha Berlin]              -0.0490       0.183      -0.268       0.789
-0.408       0.310
team[T.Kaiserslautern]             -0.2159       0.191      -1.128       0.259
-0.591       0.159
team[T.Leverkusen]                 -0.1350       0.186      -0.724       0.469
-0.500       0.230
team[T.Munich 1860]                -0.3458       0.199      -1.740       0.082
-0.735       0.044
team[T.Schalke 04]                  0.0450       0.178       0.253       0.800
-0.303       0.394
team[T.Unterhaching]               -0.5476       0.212      -2.586       0.010
-0.963      -0.133
team[T.VfB Stuttgart]              -0.3765       0.200      -1.881       0.060
-0.769       0.016
team[T.Werder Bremen]              -0.1443       0.187      -0.770       0.441
-0.512       0.223
team[T.Wolfsburg]                  -0.0234       0.181      -0.129       0.897
-0.379       0.332
opponent[T.Bochum]                  0.5578       0.205       2.719       0.007
0.156       0.960
opponent[T.Dortmund]                0.1272       0.226       0.563       0.573
-0.315       0.570
opponent[T.Eintracht Frankfurt]     0.5858       0.205       2.863       0.004
0.185       0.987
opponent[T.Energie Cottbus]         0.3131       0.215       1.454       0.146
-0.109       0.735
opponent[T.FC Koln]                 0.3380       0.215       1.569       0.117
-0.084       0.760
opponent[T.Freiburg]               -0.0094       0.233      -0.040       0.968
-0.466       0.447
opponent[T.Hamburger SV]            0.4465       0.211       2.118       0.034
0.033       0.860
```

```
opponent[T.Hansa Rostock]           0.2071      0.220      0.941      0.347
-0.224       0.638
opponent[T.Hertha Berlin]           0.3368      0.215      1.563      0.118
-0.085       0.759
opponent[T.Kaiserslautern]          0.3639      0.214      1.702      0.089
-0.055       0.783
opponent[T.Leverkusen]              0.0688      0.228      0.301      0.763
-0.379       0.517
opponent[T.Munich 1860]             0.3752      0.213      1.762      0.078
-0.042       0.793
opponent[T.Schalke 04]             -0.0522      0.236     -0.221      0.825
-0.515       0.411
opponent[T.Unterhaching]            0.4362      0.210      2.077      0.038
0.025       0.848
opponent[T.VfB Stuttgart]           0.2582      0.218      1.184      0.237
-0.169       0.686
opponent[T.Werder Bremen]           0.2505      0.219      1.143      0.253
-0.179       0.680
opponent[T.Wolfsburg]               0.1941      0.222      0.873      0.383
-0.242       0.630
home                                0.4231      0.068      6.197      0.000
0.289       0.557
================================================================================
===================
"""
```

[20]:

```
def get_teams(pos: Tuple[int, int], table: pd.DataFrame) -> Tuple[str, str]:
    return table.iloc[[pos[0]]]["Team"].item(), table.iloc[[pos[1]]]["Team"].
 ↪item()
```

[21]:

```
home_team, away_team = get_teams((0, 3), df_full_table)
home_team, away_team
```

[21]:

```
('Bayern Munich', 'Leverkusen')
```

[22]:

```
def predict_match(home_team: str, away_team: str, verbose: bool):
    home_score_rate = poisson_model.predict(
        pd.DataFrame(
            data={"team": home_team, "opponent": away_team, "home": 1},␣
 ↪index=[1]
        )
    )
    away_score_rate = poisson_model.predict(
        pd.DataFrame(
            data={"team": away_team, "opponent": home_team, "home": 0},␣
 ↪index=[1]
        )
    )
    if verbose:
        print(
            home_team
            + " against "
            + away_team
            + " expect to score: "
            + str(home_score_rate)
        )
        print(
            away_team
            + " against "
            + home_team
            + " expect to score: "
            + str(away_score_rate)
        )

    # Lets just get a result
    home_goals = np.random.poisson(home_score_rate)
    away_goals = np.random.poisson(away_score_rate)
    home_result = home_goals[0]
    away_result = away_goals[0]
    # home_state = "W" if home_result > away_result else "L"
    # if home_result == away_result:
    #     home_state = "D"
    home_state = None
    if float(home_score_rate) - float(away_score_rate) > 0.5:
        home_state = "W"
    elif float(home_score_rate) - float(away_score_rate) < -0.5:
        home_state = "L"
    else:
        home_state = "D"

    if verbose:
        print(home_team + ": " + str(home_result))
        print(away_team + ": " + str(away_result))

    return [home_team, away_team, home_result, away_result, home_state]
```

[23]:

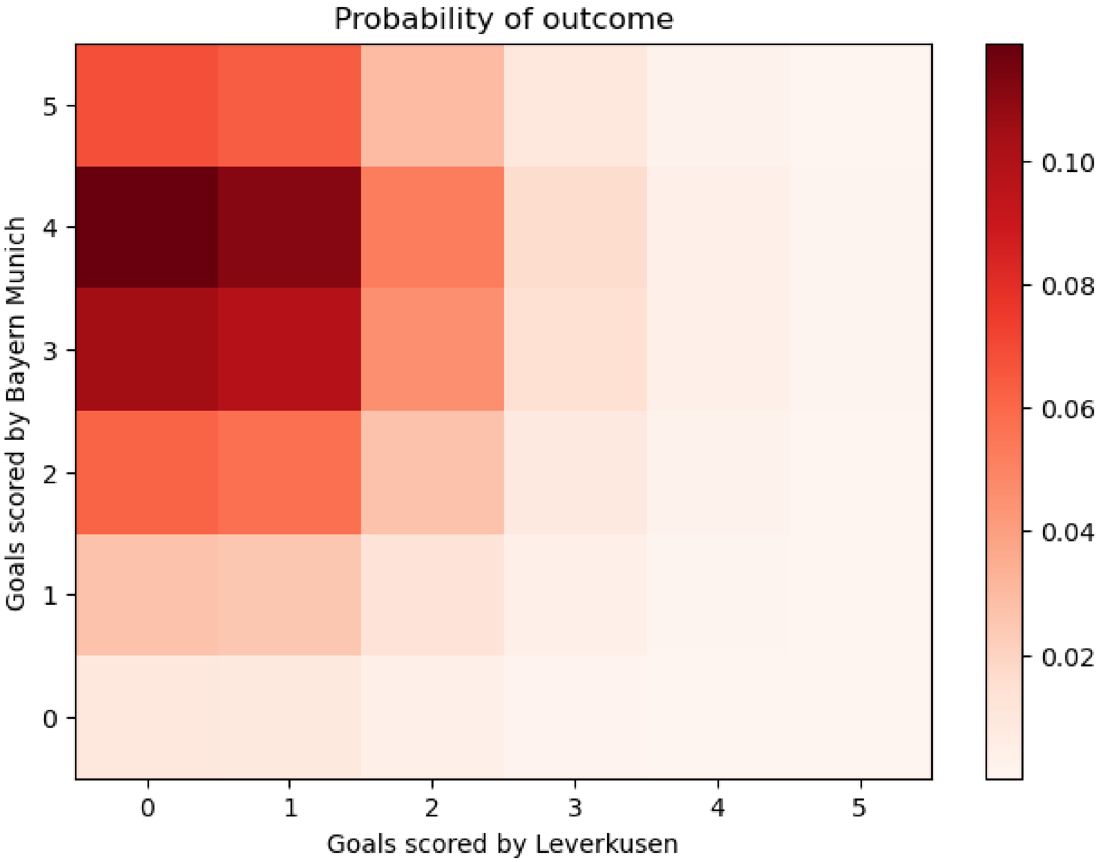
```
predict_match(home_team, away_team, True)
```

```
Bayern Munich against Leverkusen expect to score: 1    1.753296
dtype: float64
Leverkusen against Bayern Munich expect to score: 1    0.936653
dtype: float64
Bayern Munich: 2
Leverkusen: 0
```

```
[23]: ['Bayern Munich', 'Leverkusen', 2, 0, 'W']
```

```
[24]: # Code to caluclate the goals for the match.
def simulate_match(foot_model, homeTeam, awayTeam, max_goals=10):
    home_goals_avg = foot_model.predict(
        pd.DataFrame(
            data={"team": homeTeam, "opponent": awayTeam, "home": 1}, index=[1]
        )
    ).values[0]
    away_goals_avg = foot_model.predict(
        pd.DataFrame(
            data={"team": awayTeam, "opponent": homeTeam, "home": 0}, index=[1]
        )
    ).values[0]
    team_pred = [
        [poisson.pmf(i, team_avg) for i in range(0, max_goals + 1)]
        for team_avg in [home_goals_avg, away_goals_avg]
    ]
    return np.outer(np.array(team_pred[0]), np.array(team_pred[1]))


def match_histogram(poisson_model, home_team: str, away_team: str):
    # Fill in the matrix
    max_goals = 5
    score_matrix = simulate_match(poisson_model, home_team, away_team,␣
 ↪max_goals)

    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    pos = ax.imshow(
        score_matrix,
        extent=[-0.5, max_goals + 0.5, -0.5, max_goals + 0.5],
        aspect="auto",
        cmap=plt.cm.Reds,
    )
    fig.colorbar(pos, ax=ax)
    ax.set_title("Probability of outcome")
    plt.xlim((-0.5, 5.5))
    plt.ylim((-0.5, 5.5))
    plt.tight_layout()
    ax.set_xlabel("Goals scored by " + away_team)
    ax.set_ylabel("Goals scored by " + home_team)
    plt.show()

    # Home, draw, away probabilities
    homewin = np.sum(np.tril(score_matrix, -1))
    draw = np.sum(np.diag(score_matrix))
    awaywin = np.sum(np.triu(score_matrix, 1))

    return (homewin, draw, awaywin)
```

[25]:
```
match_histogram(poisson_model, home_team, away_team)
```



[25]:
```
(0.5569346601684995, 0.23440095655136858, 0.19903294131326035)
```

1.7 Questão 5

- Utilize o modelo treinado para simular os placares esperados de todos os jogos da temporada.
- Construa uma tabela de classificação em cima dos resultados esperados. Considere que jogos com uma diferença esperada de gols < 0.5 é um empate.
- Compare a tabela real com a simulada. Onde estão as principais diferenças entre elas? E similaridades? O que isso pode indicar em termos de o que modelo subestima e superestima sobre a qualidade dos times?

[26]:
```
def get_table_adjusted(df: pd.DataFrame, league: str, season: str):
    teams = df["HT"].unique()
    data: list[list] = []
    for team in teams:
        df_team: pd.DataFrame = df.query(f"HT == '{team}' or AT == '{team}'")
        played = len(df_team)

        won = len(
            df_team.query(
                f"(HT == '{team}' and WDL == 'W') or (AT == '{team}' and WDL ==␣
↪'L')"
            )
        )
        drawn = len(
            df_team.query(
                f"(HT == '{team}' and WDL == 'D') or (AT == '{team}' and WDL ==␣
↪'D')"
            )
        )
        lost = len(
            df_team.query(
                f"(HT == '{team}' and WDL == 'L') or (AT == '{team}' and WDL ==␣
↪'W')"
            )
        )

        assert played == won + drawn + lost

        points = 3 * won + drawn

        data.append([team, played, won, drawn, lost, points])

    df_table = pd.DataFrame(
        data,
        columns=["Team", "Matches", "Won", "Drawn", "Lost", "Points"],
    )
    df_table = df_table.sort_values(by=["Points", "Won"], ascending=False)
    return df_table
```

[27]:

```
def championship(league: str, season: str):
    data: list[list] = []
    df_league = df.query(f"Lge == '{league}' and Sea == '{season}'")
    teams = df_league["HT"].unique()
    for x in teams:
        for y in teams:
            if x != y:
                match = predict_match(x, y, False)
                data.append(match)
    df_simulation_games = pd.DataFrame(
        data,
        columns=["HT", "AT", "HS", "AS", "WDL"],
    )
    return get_table_adjusted(df_simulation_games, league, season)
```

```
[28]: df_table_simulation = championship(LEAGUE, SEASON)
```

```
[29]: # df_table_simulation.to_csv("simulation.csv", index=False)
```

```
[30]: df_table_simulation
```

```
[30]:                  Team  Matches  Won  Drawn  Lost  Points
      8           Schalke 04       34   23     11     0      80
      1        Bayern Munich       34   19     14     1      71
      2             Freiburg       34   18     13     3      67
      0             Dortmund       34   18     12     4      66
      15           Wolfsburg       34   16     13     5      61
      5           Leverkusen       34   14     15     5      57
      6        Werder Bremen       34   12     13     9      49
      12       Hertha Berlin       34   12     13     9      49
      16             FC Koln       34   12     13     9      49
      3          Hamburger SV       34   12     12    10      48
      4       Kaiserslautern       34    8     16    10      40
      13        VfB Stuttgart       34    6     17    11      35
      10          Munich 1860       34    5     16    13      31
      9       Energie Cottbus       34    3     17    14      26
      17        Hansa Rostock       34    3     17    14      26
      14         Unterhaching       34    2     12    20      18
      7    Eintracht Frankfurt       34    2     11    21      17
      11              Bochum       34    0      7    27       7
```

```
[31]: # df_full_table.to_csv("real.csv", index=False)
      df_full_table
```

```
[31]:                  Team  Matches  Won  Drawn  Lost    GF    GA    GD  Points
      1        Bayern Munich       34   19      6     9  62.0  37.0  25.0      63
      8           Schalke 04       34   18      8     8  65.0  35.0  30.0      62
      0             Dortmund       34   16     10     8  62.0  42.0  20.0      58
      5           Leverkusen       34   17      6    11  54.0  40.0  14.0      57
      12       Hertha Berlin       34   18      2    14  58.0  52.0   6.0      56
      2             Freiburg       34   15     10     9  54.0  37.0  17.0      55
      6        Werder Bremen       34   15      8    11  53.0  48.0   5.0      53
      4       Kaiserslautern       34   15      5    14  49.0  54.0  -5.0      50
      15           Wolfsburg       34   12     11    11  60.0  45.0  15.0      47
      16             FC Koln       34   12     10    12  59.0  52.0   7.0      46
      10          Munich 1860       34   12      8    14  43.0  55.0 -12.0      44
      17        Hansa Rostock       34   12      7    15  34.0  47.0 -13.0      43
      3         Hamburger SV       34   10     11    13  58.0  58.0   0.0      41
```

```
9       Energie Cottbus       34   12      3     19  38.0  52.0 -14.0      39
13        VfB Stuttgart       34    9     11     14  42.0  49.0  -7.0      38
7   Eintracht Frankfurt       34   10      5     19  41.0  68.0 -27.0      35
14         Unterhaching       34    8     11     15  35.0  59.0 -24.0      35
11               Bochum       34    7      6     21  30.0  67.0 -37.0      27
```

1.8 Comparação

Gerando algumas simulações, é possível perceber que o modelo tende a exagerar os extremos. Isso fica expecialmente claro ao se olhar a tabela de pontos: existem casos em que o primeiro time fica com 80 pontos e o último com 8 (quando, na realidade, essa variação fica faixa 63-27). Também é possível perceber essa discrepância ao se notar que o conjunto os times extremos se mantém, enquanto há maior variabilidade nos times mais próximos da média. Isto é, o mesmo conjunto de 4 times vencedores (ou perdedores) se mantém, e a ordem outros muda em maior intensidade. No entanto, essa análise é rasa, seria necessário aplicar métodos estatísticos para se ter uma ideia da real efetividade do modelo. Por exemplo, seria possível gerar uma grande quantidade de simulações e fazer uma média.