# 22110189-es114-dn3

April 25, 2023

[ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import numpy as np
from sklearn.cluster import KMeans
import plotly
import plotly.graph_objs as go

df1 = pd.read_csv('/content/AusOpen-men-2013.csv')
df2 = pd.read_csv('/content/AusOpen-women-2013.csv')
df3 = pd.read_csv('/content/FrenchOpen-men-2013.csv')
df4 = pd.read_csv('/content/USOpen-men-2013.csv')
df5 = pd.read_csv('/content/USOpen-women-2013.csv')
df6 = pd.read_csv('/content/FrenchOpen-women-2013.csv')
df7 = pd.read_csv('/content/Wimbledon-men-2013.csv')
df8 = pd.read_csv("/content/Wimbledon-women-2013.csv")

df1 = df1.fillna(0)
df2 = df2.fillna(0)
df3 = df3.fillna(0)
df4 = df4.fillna(0)
df5 = df4.fillna(0)
df6 = df6.fillna(0)
df7 = df7.fillna(0)
df8 = df8.fillna(0)
```

## 1 Q 1.

Suppose there is a match between 2 players, can we predict who is going to win.

---

[ ]:

```
player1_df = df1.groupby('Player1').size().reset_index(name='Matches').
  ↪rename(columns={'Player1': 'Player'})
player2_df = df1.groupby('Player2').size().reset_index(name='Matches').
  ↪rename(columns={'Player2': 'Player'})
```

```
player_matches_df = pd.concat([player1_df, player2_df], ignore_index=True).
 ↪groupby('Player').sum().reset_index()

print(player_matches_df.to_markdown())
```

```
|     | Player                 |   Matches |
|----:|:-----------------------|----------:|
|   0 | Adrian Mannarino       |         2 |
|   1 | Albert Montanes        |         1 |
|   2 | Albert Ramos           |         1 |
|   3 | Alejandro Falla        |         2 |
|   4 | Alejandro Gonzalez     |         1 |
|   5 | Aleksandr Nedovyesov   |         1 |
|   6 | Alex Bogomolov Jr.     |         1 |
|   7 | Alexandr Dolgopolov    |         2 |
|   8 | Aljaz Bedene           |         1 |
|   9 | Andreas Seppi          |         2 |
|  10 | Andrey Golubev         |         1 |
|  11 | Andy Murray            |         5 |
|  12 | Benjamin Becker        |         1 |
|  13 | Benoit Paire           |         3 |
|  14 | Bernard Tomic          |         1 |
|  15 | Blaz Kavcic            |         2 |
|  16 | Blaz Rola              |         2 |
|  17 | Bradley Klahn          |         1 |
|  18 | Carlos Berlocq         |         1 |
|  19 | Damir Dzumhur          |         3 |
|  20 | Daniel Brands          |         1 |
|  21 | Daniel Gimeno-Traver   |         1 |
|  22 | David Ferrer           |         5 |
|  23 | David Guez             |         1 |
|  24 | Denis Istomin          |         3 |
|  25 | Denis Kudla            |         1 |
|  26 | Di Wu                  |         1 |
|  27 | Dmitry Tursunov        |         2 |
|  28 | Dominic Thiem          |         2 |
|  29 | Donald Young           |         3 |
|  30 | Dudi Sela              |         1 |
|  31 | Dusan Lajovic          |         2 |
|  32 | Edouard Roger-Vasselin |         3 |
|  33 | Ernests Gulbis         |         2 |
|  34 | Fabio Fognini          |         4 |
|  35 | Federico Delbonis      |         1 |
|  36 | Feliciano Lopez        |         3 |
|  37 | Fernando Verdasco      |         2 |
|  38 | Filippo Volandri       |         1 |
|  39 | Florian Mayer          |         4 |
|  40 | Frank Dancevic         |         1 |
|  41 | Gael Monfils           |         3 |
|  42 | Gilles Simon           |         3 |
|  43 | Go Soeda               |         1 |
|  44 | Grigor Dimitrov        |         5 |
|  45 | Guillermo Garcia-Lopez |         2 |
|  46 | Horacio Zeballos       |         1 |
|  47 | Igor Sijsling          |         1 |
|  48 | Ivan Dodig             |         2 |
|  49 | Ivo Karlovic           |         1 |
|  50 | Jack Sock              |         2 |
|  51 | James Duckworth        |         1 |
|  52 | Jan Hajek              |         1 |
|  53 | Jan-Lennard Struff     |         1 |
|  54 | Jarkko Nieminen        |         2 |
|  55 | Jeremy Chardy          |         3 |
|  56 | Jerzy Janowicz         |         3 |
|  57 | Jesse Huta Galung      |         1 |
|  58 | Jimmy Wang             |         1 |
|  59 | Jiri Vesely            |         1 |
|  60 | Jo-Wilfried Tsonga     |         4 |
|  61 | Joao Sousa             |         1 |
|  62 | John Isner             |         1 |
|  63 | Jordan Thompson        |         1 |
|  64 | Juan Martin Del Potro  |         2 |
|  65 | Juan Monaco            |         1 |
|  66 | Julian Reister         |         1 |
|  67 | Julien Benneteau       |         2 |
|  68 | Kei Nishikori          |         4 |
|  69 | Kenny De Schepper      |         2 |
|  70 | Kevin Anderson         |         4 |
|  71 | Leonardo Mayer         |         2 |
|  72 | Lleyton Hewitt         |         1 |
|  73 | Lucas Pouille          |         1 |
|  74 | Lukas Lacko            |         1 |
|  75 | Lukas Rosol            |         1 |
|  76 | Lukasz Kubot           |         1 |
|  77 | Marcel Granollers      |         1 |
|  78 | Marcos Baghdatis       |         1 |
|  79 | Marin Cilic            |         2 |
|  80 | Marinko Matosevic      |         1 |
|  81 | Martin Klizan          |         3 |
|  82 | Matthew Ebden          |         2 |
|  83 | Michael Berrer         |         2 |
|  84 | Michael Llodra         |         1 |
|  85 | Michael Russell        |         1 |
|  86 | Michal Przysiezny      |         2 |
|  87 | Mikhail Kukushkin      |         1 |
|  88 | Mikhail Youzhny        |         2 |
|  89 | Milos Raonic           |         3 |
|  90 | Nick Kyrgios           |         2 |
|  91 | Nicolas Mahut          |         1 |
|  92 | Nikolay Davydenko      |         2 |
|  93 | Novak Djokovic         |         5 |
|  94 | Pablo Andujar          |         2 |
|  95 | Pablo Carreno Busta    |         1 |
|  96 | Peter Gojowczyk        |         1 |
|  97 | Radek Stepanek         |         1 |
|  98 | Rafael Nadal           |         7 |
|  99 | Rhyne Williams         |         1 |
| 100 | Ricardas Berankis      |         1 |
| 101 | Richard Gasquet        |         3 |
| 102 | Roberto Bautista Agut  |         4 |
| 103 | Robin Haase            |         1 |
| 104 | Roger Federer          |         6 |
| 105 | Ryan Harrison          |         1 |
| 106 | Sam Querrey            |         3 |
| 107 | Samuel Groth           |         1 |
| 108 | Santiago Giraldo       |         1 |
| 109 | Sergiy Stakhovsky      |         1 |
| 110 | Somdev Devvarman       |         1 |
| 111 | Stanislas Wawrinka     |         6 |
| 112 | Stephane Robert        |         4 |
| 113 | Steve Johnson          |         1 |
| 114 | Teymuraz Gabashvili    |         3 |
| 115 | Thanasi Kokkinakis     |         2 |
| 116 | Thomaz Bellucci        |         2 |
| 117 | Tim Smyczek            |         1 |
| 118 | Tobias Kamke           |         1 |
| 119 | Tomas Berdych          |         6 |
| 120 | Tommy Haas             |         1 |
| 121 | Tommy Robredo          |         4 |
| 122 | Vasek Pospisil         |         2 |
| 123 | Victor Hanescu         |         2 |
| 124 | Vincent Millot         |         2 |
| 125 | Wayne Odesnik          |         1 |
| 126 | Yen-Hsun Lu            |         2 |
| 127 | Ze Zhang               |         1 |
```

we had taken the best players in termes of maximum number of matches played so that data is also clear

[ ]:

```
Rogerdf = pd.DataFrame()
Rafaeldf = pd.DataFrame()

for index, row in df1.iterrows():
    if row['Player1'] == 'Roger Federer':
        Rogerdf = Rogerdf.append({'FSP': row['FSP.1'],
                              'FSW': row['FSW.1'],
                              'SSP': row['SSP.1'],
                              'SSW': row['SSW.1'],
                              'ACE': row['ACE.1'],
                              'DBF': row['DBF.1'],
                              'WNR': row['WNR.1'],
                              'UFE': row['UFE.1'],
                              'BPC': row['BPC.1'],
                              'BPW': row['BPW.1'],
                              'NPA': row['NPA.1'],
                              'NPW': row['NPW.1'],
                              'TPW': row['TPW.1'],
                              'ST1': row['ST1.1'],
                              'ST2': row['ST2.1'],
                              'ST3': row['ST3.1'],
                              'ST4': row['ST4.1'],
                              'ST5': row['ST5.1']}, ignore_index=True)

    elif row['Player2'] == 'Roger Federer':
        Rogerdf = Rogerdf.append({'FSP': row['FSP.2'],
                              'FSW': row['FSW.2'],
                              'FSP': row['FSP.2'],
                              'FSW': row['FSW.2'],
                              'SSP': row['SSP.2'],
                              'SSW': row['SSW.2'],
                              'ACE': row['ACE.2'],
                              'DBF': row['DBF.2'],
                              'WNR': row['WNR.2'],
                              'UFE': row['UFE.2'],
                              'BPC': row['BPC.2'],
                              'BPW': row['BPW.2'],
                              'NPA': row['NPA.2'],
                              'NPW': row['NPW.2'],
                              'TPW': row['TPW.2'],
                              'ST1': row['ST1.2'],
                              'ST2': row['ST2.2'],
                              'ST3': row['ST3.2'],
                              'ST4': row['ST4.2'],
                              'ST5': row['ST5.2']}, ignore_index=True)



for index, row in df1.iterrows():
    if row['Player1'] == 'Rafael Nadal':
        Rafaeldf = Rafaeldf.append({'FSP': row['FSP.1'],
                              'FSW': row['FSW.1'],
                              'SSP': row['SSP.1'],
                              'SSW': row['SSW.1'],
                              'ACE': row['ACE.1'],
                              'DBF': row['DBF.1'],
                              'WNR': row['WNR.1'],
                              'UFE': row['UFE.1'],
                              'BPC': row['BPC.1'],
                              'BPW': row['BPW.1'],
                              'NPA': row['NPA.1'],
                              'NPW': row['NPW.1'],
                              'TPW': row['TPW.1'],
                              'ST1': row['ST1.1'],
                              'ST2': row['ST2.1'],
                              'ST3': row['ST3.1'],
                              'ST4': row['ST4.1'],
                              'ST5': row['ST5.1']}, ignore_index=True)

    elif row['Player2'] == 'Rafael Nadal':
        Rafaeldf = Rafaeldf.append({'FSP': row['FSP.2'],
                              'FSW': row['FSW.2'],
                              'FSP': row['FSP.2'],
                              'FSW': row['FSW.2'],
                              'SSP': row['SSP.2'],
                              'SSW': row['SSW.2'],
                              'ACE': row['ACE.2'],
                              'DBF': row['DBF.2'],
                              'WNR': row['WNR.2'],
                              'UFE': row['UFE.2'],
                              'BPC': row['BPC.2'],
                              'BPW': row['BPW.2'],
                              'NPA': row['NPA.2'],
                              'NPW': row['NPW.2'],
                              'TPW': row['TPW.2'],
                              'ST1': row['ST1.2'],
                              'ST2': row['ST2.2'],
                              'ST3': row['ST3.2'],
                              'ST4': row['ST4.2'],
                              'ST5': row['ST5.2']}, ignore_index=True)
```

```
<ipython-input-751-ee56749c875f>:26: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:26: FutureWarning:
```

```
The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:26: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:26: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:26: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:26: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
```

```
<ipython-input-751-ee56749c875f>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-751-ee56749c875f>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
```

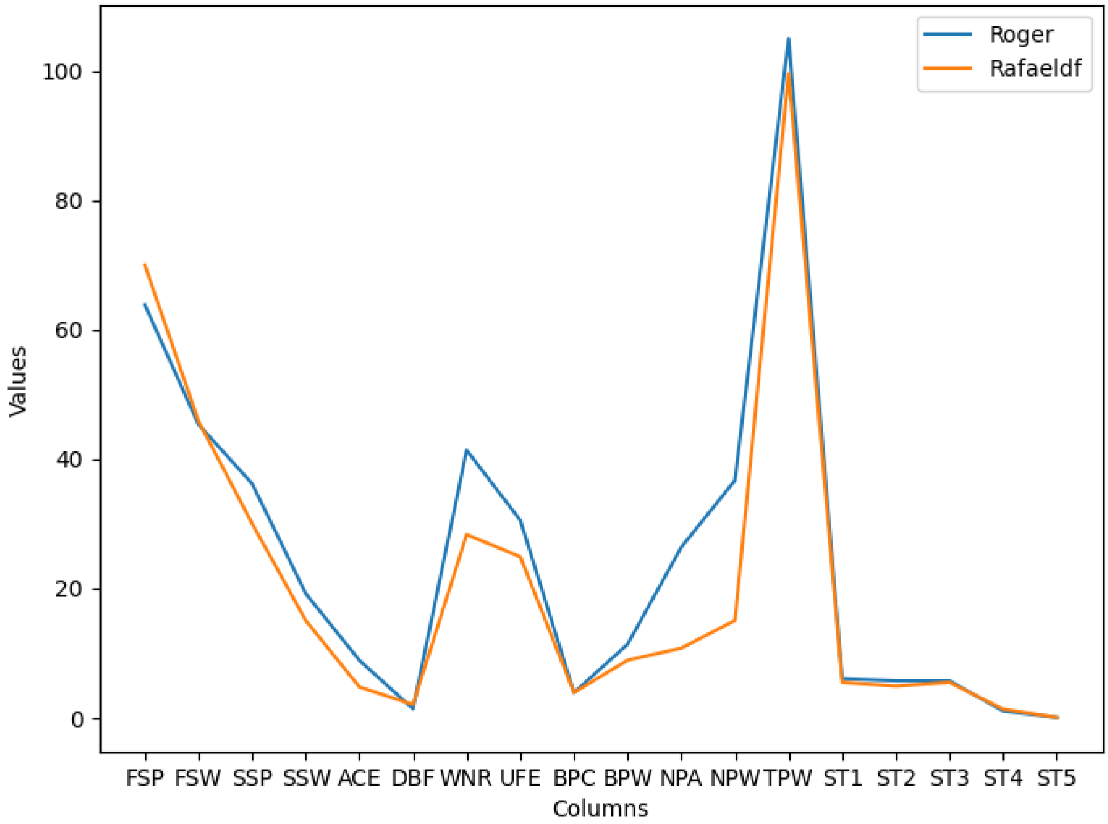
[ ]:

```
mean_row = Rogerdf.mean(axis=0)
Rogerdf.loc['mean'] = mean_row

mean_row = Rafaeldf.mean(axis=0)
Rafaeldf.loc['mean'] = mean_row

display(Rogerdf)
display(Rafaeldf)
```

```
            FSP        FSW        SSP        SSW        ACE        DBF  \
0     58.000000  40.000000  42.000000  19.000000  11.000000  5.000000
1     76.000000  45.000000  24.000000   9.000000  11.000000  0.000000
2     62.000000  39.000000  38.000000  16.000000   6.000000  1.000000
3     57.000000  42.000000  43.000000  25.000000   7.000000  1.000000
4     64.000000  61.000000  36.000000  28.000000  10.000000  0.000000
5     66.000000  45.000000  34.000000  18.000000   8.000000  1.000000
mean  63.833333  45.333333  36.166667  19.166667   8.833333  1.333333

            WNR   UFE       BPC        BPW        NPA        NPW    TPW  ST1  \
0     30.000000  17.0  4.000000  17.000000  18.000000  22.000000   96.0  6.0
1     52.000000  35.0  6.000000  11.000000  23.000000  29.000000  103.0  6.0
2     35.000000  18.0  5.000000  14.000000  11.000000  20.000000   98.0  6.0
3     43.000000  21.0  3.000000   7.000000  34.000000  41.000000  100.0  6.0
4     54.000000  42.0  4.000000  17.000000  49.000000  66.000000  147.0  6.0
5     34.000000  50.0  1.000000   2.000000  23.000000  42.000000   86.0  6.0
mean  41.333333  30.5  3.833333  11.333333  26.333333  36.666667  105.0  6.0

           ST2       ST3  ST4  ST5
0     6.000000  6.000000  0.0  0.0
1     6.000000  7.000000  0.0  0.0
2     6.000000  6.000000  0.0  0.0
3     7.000000  6.000000  0.0  0.0
4     6.000000  6.000000  6.0  0.0
5     3.000000  3.000000  0.0  0.0
mean  5.666667  5.666667  1.0  0.0
       FSP        FSW    SSP    SSW        ACE   DBF        WNR        UFE  \
0     62.0  14.000000  38.0   6.0   5.000000  1.0  13.000000   5.000000
1     74.0  45.000000  26.0  11.0   7.000000  1.0  39.000000  19.000000
2     67.0  38.000000  33.0  20.0   2.000000  0.0  21.000000  18.000000
3     71.0  67.000000  29.0  14.0  12.000000  1.0  36.000000  28.000000
4     73.0  66.000000  27.0  22.0   3.000000  7.0  42.000000  47.000000
5     65.0  41.000000  35.0  22.0   3.000000  1.0  28.000000  25.000000
6     78.0  50.000000  22.0  10.0   1.000000  3.0  19.000000  32.000000
mean  70.0  45.857143  30.0  15.0   4.714286  2.0  28.285714  24.857143

           BPC        BPW        NPA   NPW         TPW       ST1       ST2  \
0     1.000000   1.000000   5.000000   5.0   30.000000  6.000000  0.000000
1     5.000000  10.000000  11.000000  14.0   91.000000  6.000000  6.000000
2     6.000000  12.000000   9.000000  13.0  107.000000  6.000000  6.000000
3     5.000000   8.000000  10.000000  19.0  129.000000  7.000000  7.000000
4     4.000000  11.000000  32.000000  36.0  147.000000  3.000000  7.000000
5     4.000000  14.000000   3.000000  10.0  105.000000  7.000000  6.000000
6     2.000000   6.000000   5.000000   8.0   88.000000  3.000000  2.000000
mean  3.857143   8.857143  10.714286  15.0   99.571429  5.428571  4.857143

           ST3       ST4  ST5
0     0.000000  0.000000  0.0
1     6.000000  0.000000  0.0
2     6.000000  0.000000  0.0
3     7.000000  0.000000  0.0
4     7.000000  6.000000  0.0
5     6.000000  0.000000  0.0
6     6.000000  3.000000  0.0
mean  5.428571  1.285714  0.0
```

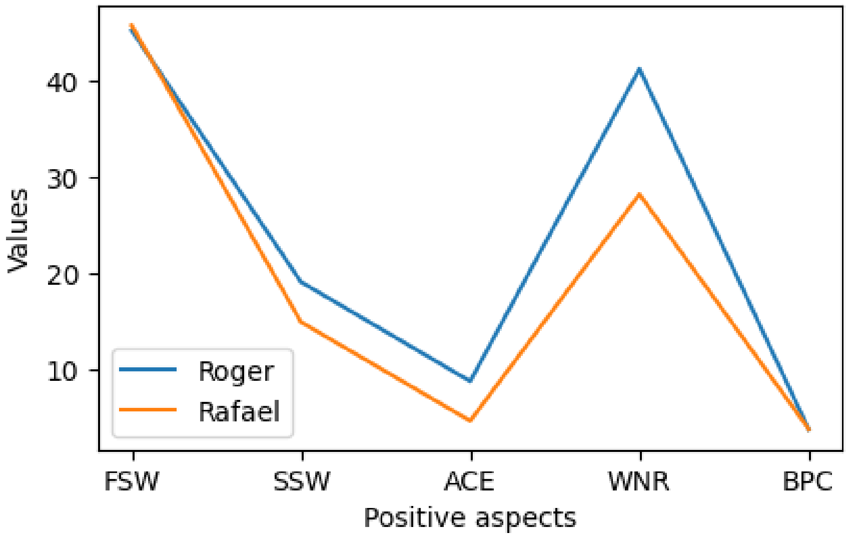
[ ]:

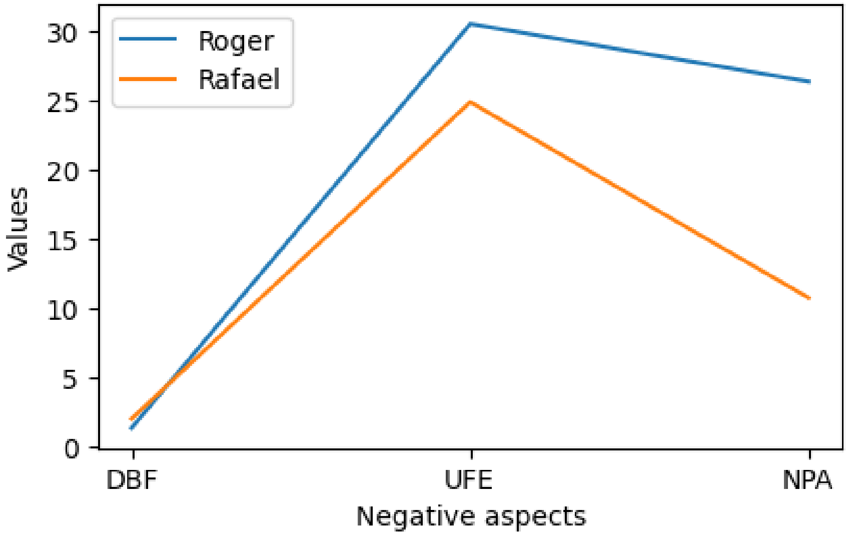
```
fig, ax = plt.subplots(figsize=(8,6))
x = Rogerdf.columns
y1 = Rogerdf.iloc[6]
y2 = Rafaeldf.iloc[7]
ax.plot(x, y1, label='Roger')
ax.plot(x, y2, label='Rafaeldf')
ax.set_xlabel('Columns')
ax.set_ylabel('Values')
ax.legend()
plt.show()
```



[ ]:
```
fig, ax = plt.subplots(figsize=(5,3))
x1 = ['FSW', 'SSW', 'ACE', 'WNR', 'BPC']
y1_1 = Rogerdf.iloc[6, [1, 3, 4, 6, 8]]
y1_2 = Rafaeldf.iloc[7, [1, 3, 4, 6, 8]]
ax.plot(x1, y1_1, label='Roger')
ax.plot(x1, y1_2, label='Rafael')
ax.set_xlabel('Positive aspects')
ax.set_ylabel('Values')
ax.legend()
plt.show()

fig, ax = plt.subplots(figsize=(5,3))
x2 = ['DBF', 'UFE', 'NPA']
y2_1 = Rogerdf.iloc[6, [5, 7, 10]]
y2_2 = Rafaeldf.iloc[7, [5, 7, 10]]
ax.plot(x2, y2_1, label='Roger')
ax.plot(x2, y2_2, label='Rafael')
ax.set_xlabel('Negative aspects')
ax.set_ylabel('Values')
ax.legend()
```

```
plt.show()
```





through the plot it is not clear as Roger has made positve points also more and also negative points therefore we need to make formula and one with more score will be the best one and will have more probability of wining

[ ]:
```
def formula(FSW,SSW,ACE,BPC,TPW,WNR,DBF,UFE,NPA):
    return␣
  ↪(1*FSW)+(2*SSW)+(3*ACE)+(2*BPC)+(2*TPW)+(1*WNR)-((3*DBF)+(3*UFE)+(2*NPA))
```

```
print('Score of Roger is',formula(Rogerdf['FSW']['mean'],␣
 ↪Rogerdf['SSW']['mean'], Rogerdf['ACE']['mean'], Rogerdf['BPC']['mean'],␣
 ↪Rogerdf['TPW']['mean'], Rogerdf['WNR']['mean'], Rogerdf['DBF']['mean'],␣
 ↪Rogerdf['UFE']['mean'], Rogerdf['NPA']['mean']))
print('Score of Rafael is',formula(Rafaeldf['FSW']['mean'],␣
 ↪Rafaeldf['SSW']['mean'], Rafaeldf['ACE']['mean'], Rafaeldf['BPC']['mean'],␣
 ↪Rafaeldf['TPW']['mean'], Rafaeldf['WNR']['mean'], Rafaeldf['DBF']['mean'],␣
 ↪Rafaeldf['UFE']['mean'], Rafaeldf['NPA']['mean']))
```

```
Score of Roger is 221.00000000000003
Score of Rafael is 223.1428571428571
```

```
[ ]: scale = MinMaxScaler()
     scale.fit(Rogerdf)
     Rogerdf = scale.transform(Rogerdf)

     scale.fit(Rafaeldf)
     Rafaeldf = scale.transform(Rafaeldf)

     print('Score of Roger is',formula(Rogerdf[6][1], Rogerdf[6][3], Rogerdf[6][4],␣
      ↪Rogerdf[6][8], Rogerdf[6][12], Rogerdf[6][6], Rogerdf[6][5], Rogerdf[6][7],␣
      ↪Rogerdf[6][10]))
     print('Score of Rafael is',formula(Rafaeldf[7][1], Rafaeldf[7][3],␣
      ↪Rafaeldf[7][4], Rafaeldf[7][8], Rafaeldf[7][12], Rafaeldf[7][6],␣
      ↪Rafaeldf[7][5], Rafaeldf[7][7], Rafaeldf[7][10]))
```

```
Score of Roger is 2.45227033057059
Score of Rafael is 2.790741199759231
```

## 2 Q 2.

Form a strategy for a player to play against a specific player.

---

```
[ ]: player1_df = df2.groupby('Player1').size().reset_index(name='Matches').
      ↪rename(columns={'Player1': 'Player'})
     player2_df = df2.groupby('Player2').size().reset_index(name='Matches').
      ↪rename(columns={'Player2': 'Player'})

     player_matches_df = pd.concat([player1_df, player2_df], ignore_index=True).
      ↪groupby('Player').sum().reset_index()

     print(player_matches_df.to_markdown())
```

```
|     | Player                      |   Matches |
|----:|:----------------------------|----------:|
|   0 | Agnieszka Radwanska         |         6 |
|   1 | Ajla Tomljanovic           |         2 |
|   2 | Alexandra Cadantu          |         1 |
|   3 | Alison Riske               |         3 |
|   4 | Alison Van Uytvanck        |         1 |
|   5 | Alize Cornet               |         3 |
|   6 | Alla Kudryavtseva          |         2 |
|   7 | Ana Ivanovic               |         5 |
|   8 | Ana Konjuh                 |         1 |
|   9 | Anabel Medina Garrigues    |         1 |
|  10 | Anastasia Pavlyuchenkova   |         3 |
|  11 | Andrea Petkovic            |         1 |
|  12 | Angelique Kerber           |         4 |
|  13 | Anna Schmiedlova           |         2 |
|  14 | Anna Tatishvili            |         1 |
|  15 | Annika Beck                |         2 |
|  16 | Ashleigh Barty             |         1 |
|  17 | Ayumi Morita               |         2 |
|  18 | Barbora Zahlavova Strycova |         2 |
|  19 | Belinda Bencic             |         2 |
|  20 | Bethanie Mattek-Sands      |         1 |
|  21 | Bojana Jovanovski          |         2 |
|  22 | Camila Giorgi              |         2 |
|  23 | Carina Witthoeft           |         1 |
|  24 | Carla Suarez Navarro       |         3 |
|  25 | Caroline Garcia            |         1 |
|  26 | Caroline Wozniacki         |         3 |
|  27 | Casey Dellacqua            |         4 |
|  28 | Chanelle Scheepers         |         1 |
|  29 | Christina McHale           |         2 |
|  30 | Daniela Hantuchova         |         3 |
|  31 | Dinah Pfizenmaier          |         1 |
|  32 | Dominika Cibulkova         |         7 |
|  33 | Donna Vekic                |         1 |
|  34 | Ekaterina Makarova         |         4 |
|  35 | Elena Vesnina              |         1 |
|  36 | Elina Svitolina            |         3 |
|  37 | Eugenie Bouchard           |         6 |
|  38 | Flavia Pennetta            |         5 |
|  39 | Francesca Schiavone        |         1 |
|  40 | Galina Voskoboeva          |         2 |
|  41 | Garbine Muguruza           |         4 |
|  42 | Hao Chen Tang              |         1 |
|  43 | Heather Watson             |         1 |
|  44 | Irina Falconi              |         2 |
|  45 | Irina-Camelia Begu         |         1 |
|  46 | Jana Cepelova              |         1 |
|  47 | Jarmila Gajdosova          |         1 |
|  48 | Jelena Jankovic            |         4 |
|  49 | Jie Zheng                  |          3 |
|  50 | Johanna Larsson            |          1 |
|  51 | Julia Glushko              |          1 |
|  52 | Julia Goerges              |          2 |
|  53 | Kaia Kanepi                |          1 |
|  54 | Karin Knapp                |          2 |
|  55 | Karolina Pliskova          |          2 |
|  56 | Katarzyna Piter            |          1 |
|  57 | Katerina Siniakova         |          1 |
|  58 | Kiki Bertens               |          1 |
|  59 | Kimiko Date-Krumm          |          1 |
|  60 | Kirsten Flipkens           |          2 |
|  61 | Klara Zakopalova           |          1 |
|  62 | Kristina Mladenovic        |          1 |
|  63 | Kurumi Nara                |          3 |
|  64 | Lara Arruabarrena          |          1 |
|  65 | Laura Robson               |          1 |
|  66 | Lauren Davis               |          3 |
|  67 | Lesia Tsurenko             |          1 |
|  68 | Lourdes Dominguez Lino     |          1 |
|  69 | Lucie Hradecka             |          2 |
|  70 | Lucie Safarova             |          3 |
|  71 | Luksika Kumkhum            |          2 |
|  72 | Madison Keys               |          2 |
|  73 | Magdalena Rybarikova       |          2 |
|  74 | Mandy Minella              |          2 |
|  75 | Maria Sharapova            |          4 |
|  76 | Mariana Duque-Marino       |          1 |
|  77 | Marina Erakovic            |          2 |
|  78 | Mirjana Lucic-Baroni       |          1 |
|  79 | Misaki Doi                 |          1 |
|  80 | Mona Barthel               |          3 |
|  81 | Monica Niculescu           |          3 |
|  82 | Monica Puig                |          2 |
|  83 | Na Li                      |          7 |
|  84 | Nadiya Kichenok            |          1 |
|  85 | Olga Govortsova            |          2 |
|  86 | Olivia Rogowska            |          2 |
|  87 | Patricia Mayr-Achleitner   |          1 |
|  88 | Paula Ormaechea            |          1 |
|  89 | Pauline Parmentier         |          1 |
|  90 | Petra Kvitova              |          1 |
|  91 | Petra Martic               |          1 |
|  92 | Polona Hercog              |          1 |
|  93 | Roberta Vinci              |          1 |
|  94 | Sabine Lisicki             |          2 |
|  95 | Sachia Vickery             |          1 |
|  96 | Samantha Stosur            |          3 |
|  97 | Sara Errani                |         1 |
|  98 | Serena Williams            |         4 |
|  99 | Shahar Peer                |         1 |
| 100 | Shuai Peng                 |         1 |
| 101 | Shuai Zhang                |         1 |
| 102 | Silvia Soler-Espinosa      |         1 |
| 103 | Simona Halep               |         5 |
| 104 | Sloane Stephens            |         4 |
| 105 | Sorana Cirstea             |         1 |
| 106 | Stefanie Voegele           |         2 |
| 107 | Storm Sanders              |         1 |
| 108 | Su-Wei Hsieh               |         1 |
| 109 | Svetlana Kuznetsova        |         1 |
| 110 | Tadeja Majeric             |         1 |
| 111 | Teliana Pereira            |         1 |
| 112 | Timea Babos                |         1 |
| 113 | Tsvetana Pironkova         |         2 |
| 114 | Vania King                 |         1 |
| 115 | Varvara Lepchenko          |         2 |
| 116 | Venus Williams             |         1 |
| 117 | Vera Zvonareva             |         1 |
| 118 | Vesna Dolonc               |         2 |
| 119 | Victoria Azarenka          |         5 |
| 120 | Virginie Razzano           |         2 |
| 121 | Yanina Wickmayer           |         2 |
| 122 | Yaroslava Shvedova         |         1 |
| 123 | Ying-Ying Duan             |         1 |
| 124 | Yulia Putintseva           |         1 |
| 125 | Yung-Jan Chan              |         1 |
| 126 | Yvonne Meusburger          |         3 |
| 127 | Zarina Diyas               |         3 |
```

[ ]:

```
Player = pd.DataFrame()
Opponent = pd.DataFrame()

for index, row in df2.iterrows():
    if row['Player1'] == 'Dominika Cibulkova':
        Player = Player.append({'FSP': row['FSP.1'],
                              'FSW': row['FSW.1'],
                              'SSP': row['SSP.1'],
                              'SSW': row['SSW.1'],
                              'ACE': row['ACE.1'],
                              'DBF': row['DBF.1'],
                              'WNR': row['WNR.1'],
                              'UFE': row['UFE.1'],
                              'BPC': row['BPC.1'],
                              'BPW': row['BPW.1'],
                              'NPA': row['NPA.1'],
                              'NPW': row['NPW.1'],
                              'TPW': row['TPW.1'],
                              'ST1': row['ST1.1'],
                              'ST2': row['ST2.1'],
                              'ST3': row['ST3.1'],
                              'ST4': row['ST4.1'],
                              'ST5': row['ST5.1']}, ignore_index=True)

    elif row['Player2'] == 'Dominika Cibulkova':
        Player = Player.append({'FSP': row['FSP.2'],
                              'FSW': row['FSW.2'],
                              'FSP': row['FSP.2'],
                              'FSW': row['FSW.2'],
                              'SSP': row['SSP.2'],
                              'SSW': row['SSW.2'],
                              'ACE': row['ACE.2'],
                              'DBF': row['DBF.2'],
                              'WNR': row['WNR.2'],
                              'UFE': row['UFE.2'],
                              'BPC': row['BPC.2'],
                              'BPW': row['BPW.2'],
                              'NPA': row['NPA.2'],
                              'NPW': row['NPW.2'],
                              'TPW': row['TPW.2'],
                              'ST1': row['ST1.2'],
                              'ST2': row['ST2.2'],
                              'ST3': row['ST3.2'],
                              'ST4': row['ST4.2'],
                              'ST5': row['ST5.2']}, ignore_index=True)



for index, row in df2.iterrows():
    if row['Player1'] == 'Na Li':
        Opponent = Opponent.append({'FSP': row['FSP.1'],
                              'FSW': row['FSW.1'],
                              'SSP': row['SSP.1'],
                              'SSW': row['SSW.1'],
                              'ACE': row['ACE.1'],
                              'DBF': row['DBF.1'],
                              'WNR': row['WNR.1'],
                              'UFE': row['UFE.1'],
                              'BPC': row['BPC.1'],
                              'BPW': row['BPW.1'],
                              'NPA': row['NPA.1'],
                              'NPW': row['NPW.1'],
                              'TPW': row['TPW.1'],
                              'ST1': row['ST1.1'],
                              'ST2': row['ST2.1'],
                              'ST3': row['ST3.1'],
                              'ST4': row['ST4.1'],
                              'ST5': row['ST5.1']}, ignore_index=True)

    elif row['Player2'] == 'Na Li':
        Opponent = Opponent.append({'FSP': row['FSP.2'],
                              'FSW': row['FSW.2'],
                              'FSP': row['FSP.2'],
                              'FSW': row['FSW.2'],
                              'SSP': row['SSP.2'],
                              'SSW': row['SSW.2'],
                              'ACE': row['ACE.2'],
                              'DBF': row['DBF.2'],
                              'WNR': row['WNR.2'],
                              'UFE': row['UFE.2'],
                              'BPC': row['BPC.2'],
                              'BPW': row['BPW.2'],
                              'NPA': row['NPA.2'],
                              'NPW': row['NPW.2'],
                              'TPW': row['TPW.2'],
                              'ST1': row['ST1.2'],
                              'ST2': row['ST2.2'],
                              'ST3': row['ST3.2'],
                              'ST4': row['ST4.2'],
                              'ST5': row['ST5.2']}, ignore_index=True)
```

```
<ipython-input-758-4887aece3ccb>:26: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:26: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:26: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:6: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
```

```
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:26: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:6: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:26: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:51: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:71: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.

<ipython-input-758-4887aece3ccb>:51: FutureWarning:
```

```
The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
```

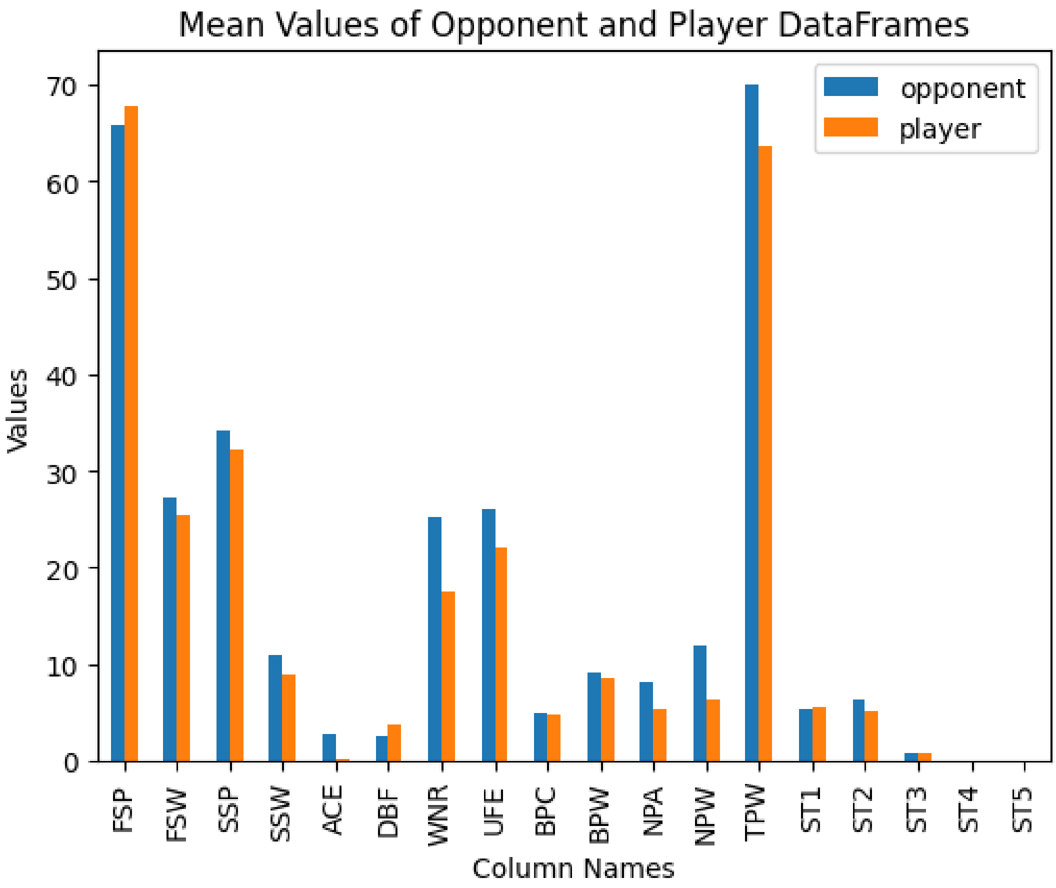
[ ]:

```
Opponent = Opponent.fillna(0)
Player = Player.fillna(0)

display(Player)
display(Opponent)
```

```
    FSP   FSW   SSP   SSW  ACE  DBF   WNR   UFE  BPC   BPW   NPA   NPW   TPW  \
0  61.0  33.0  39.0  14.0  0.0  6.0  21.0  30.0  3.0   7.0   6.0   7.0  72.0
1  68.0  21.0  32.0   8.0  0.0  2.0  17.0  13.0  6.0   9.0   4.0   5.0  55.0
2  69.0  21.0  31.0   8.0  1.0  1.0  20.0  20.0  5.0   9.0   7.0   9.0  56.0
3  69.0  27.0  31.0  13.0  1.0  5.0  16.0  28.0  7.0  14.0   2.0   2.0  87.0
4  77.0  25.0  23.0   5.0  0.0  3.0  17.0  16.0  5.0   9.0   3.0   4.0  54.0
5  64.0  22.0  36.0  10.0  0.0  2.0  21.0  20.0  6.0   9.0  12.0  14.0  63.0
6  67.0  29.0  33.0   5.0  0.0  7.0  11.0  28.0  2.0   3.0   3.0   4.0  58.0

   ST1  ST2  ST3  ST4  ST5
0  6.0  6.0  0.0  0.0  0.0
1  6.0  6.0  0.0  0.0  0.0
2  6.0  6.0  0.0  0.0  0.0
3  3.0  6.0  6.0  0.0  0.0
4  6.0  6.0  0.0  0.0  0.0
5  6.0  6.0  0.0  0.0  0.0
6  6.0  0.0  0.0  0.0  0.0
    FSP   FSW   SSP   SSW  ACE  DBF   WNR   UFE  BPC   BPW   NPA   NPW   TPW  \
0  76.0  24.0  24.0   7.0  3.0  0.0  20.0  18.0  5.0  12.0  13.0  19.0  61.0
1  60.0  27.0  40.0  12.0  4.0  2.0  30.0  31.0  5.0   7.0   2.0   5.0  67.0
2  60.0  43.0  40.0  18.0  5.0  6.0  17.0  50.0  4.0  10.0   8.0   9.0  99.0
3  80.0  26.0  20.0   4.0  1.0  1.0  18.0  14.0  5.0   7.0   8.0  14.0  55.0
4  64.0  23.0  36.0  10.0  1.0  1.0  23.0  17.0  5.0   8.0   7.0  10.0  62.0
5  61.0  27.0  39.0  11.0  4.0  5.0  35.0  23.0  6.0  10.0  11.0  14.0  71.0
6  60.0  21.0  40.0  15.0  2.0  3.0  34.0  30.0  5.0  10.0   8.0  13.0  75.0

   ST1  ST2  ST3  ST4  ST5
0  6.0  6.0  0.0  0.0  0.0
1  6.0  7.0  0.0  0.0  0.0
2  1.0  7.0  6.0  0.0  0.0
3  6.0  6.0  0.0  0.0  0.0
4  6.0  6.0  0.0  0.0  0.0
5  6.0  6.0  0.0  0.0  0.0
6  7.0  6.0  0.0  0.0  0.0
```

[ ]:

```
mean_opponent = Opponent.mean()
mean_player = Player.mean()

df_mean = pd.DataFrame({'opponent': mean_opponent, 'player': mean_player})

df_mean.plot(kind='bar')
plt.xlabel('Column Names')
plt.ylabel('Values')
plt.title('Mean Values of Opponent and Player DataFrames')
plt.show()
```

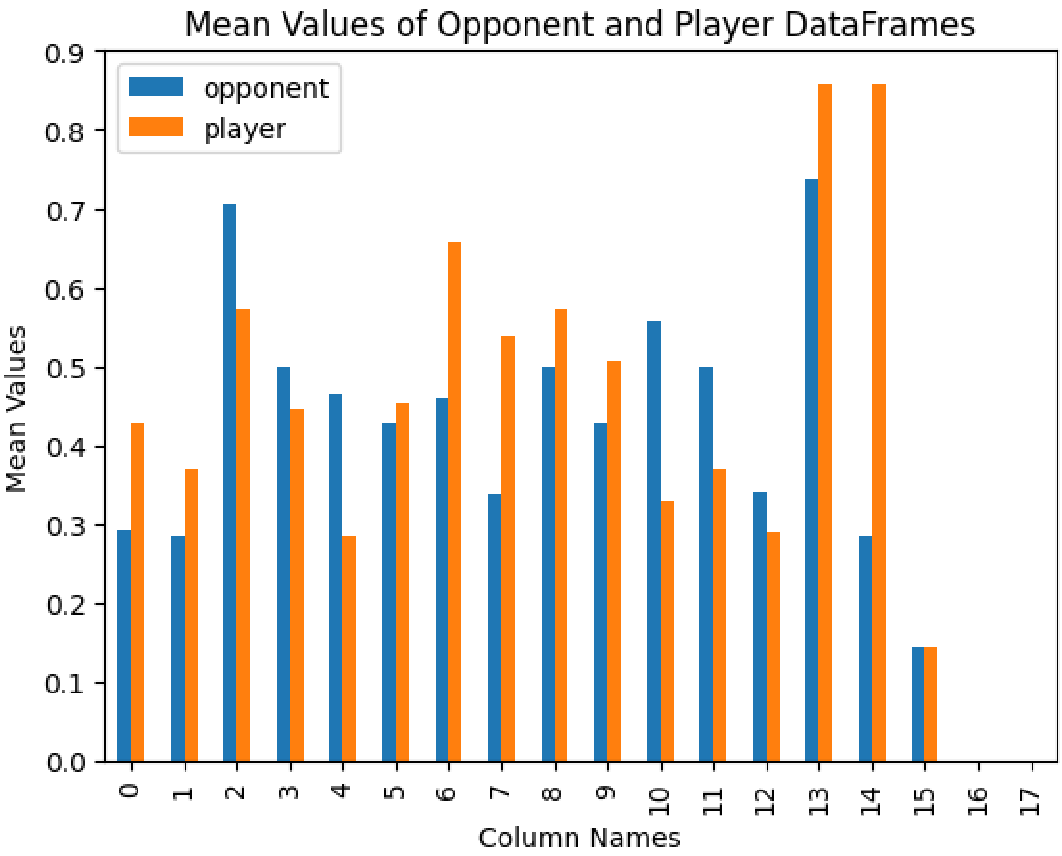


[ ]:

```
scale = MinMaxScaler()
scale.fit(Opponent)
Opponent = scale.transform(Opponent)

scale.fit(Player)
Player = scale.transform(Player)
```

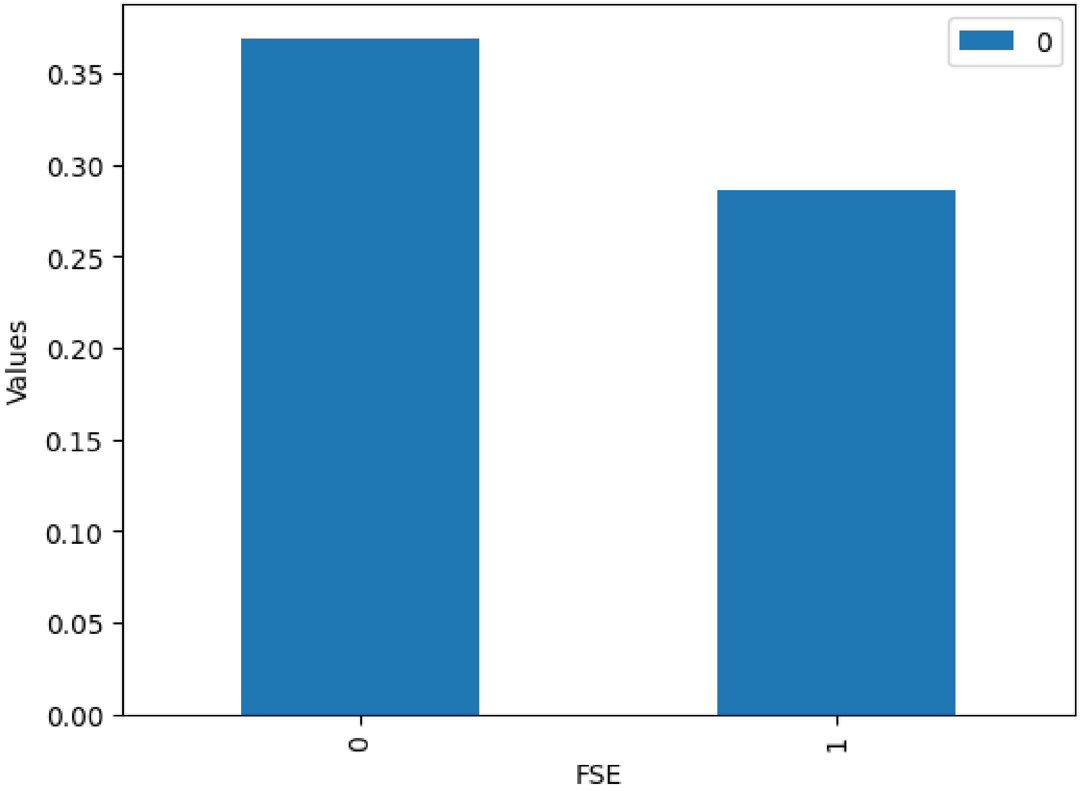
[ ]:

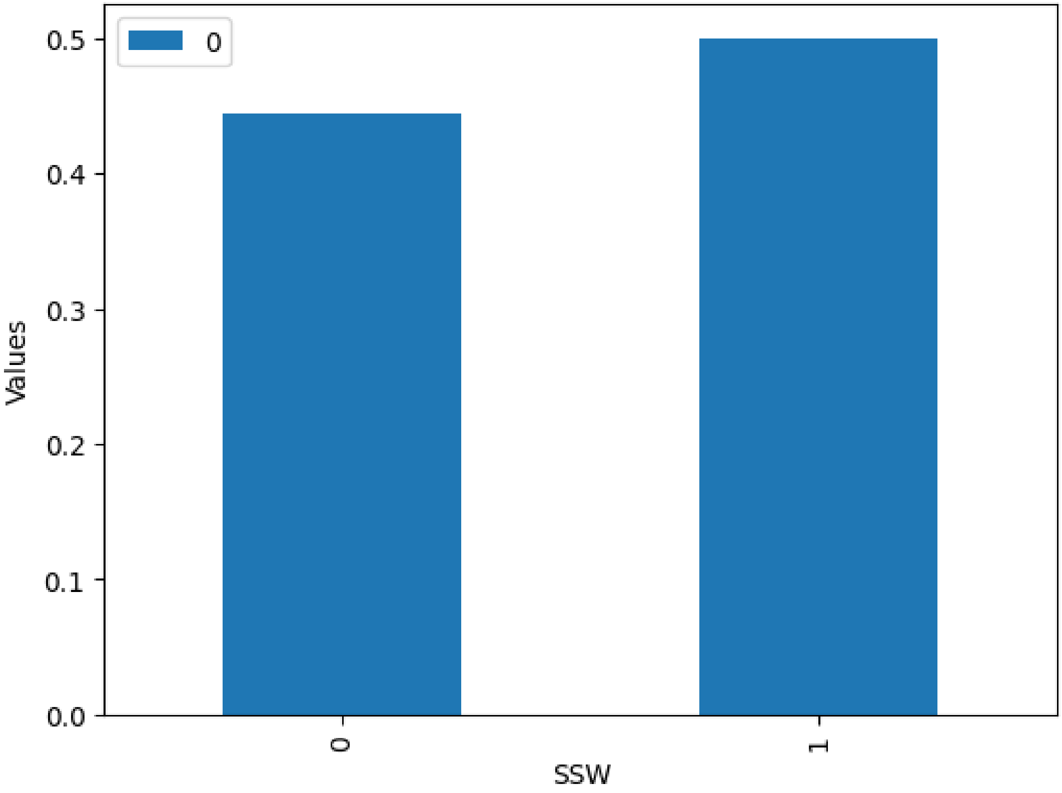
```
mean_opponent = np.mean(Opponent, axis=0)
mean_player = np.mean(Player, axis=0)

df_mean = pd.DataFrame({'opponent': mean_opponent, 'player': mean_player})

df_mean.plot(kind='bar')
plt.xlabel('Column Names')
plt.ylabel('Mean Values')
plt.title('Mean Values of Opponent and Player DataFrames')
plt.show()
```

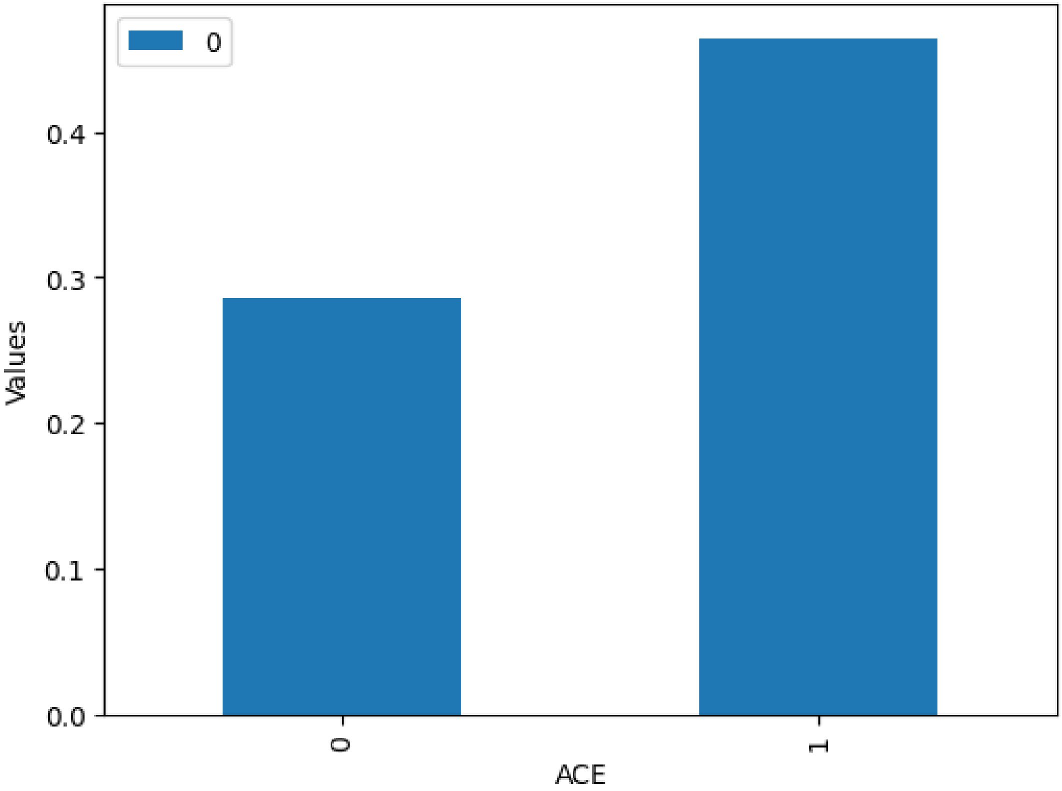


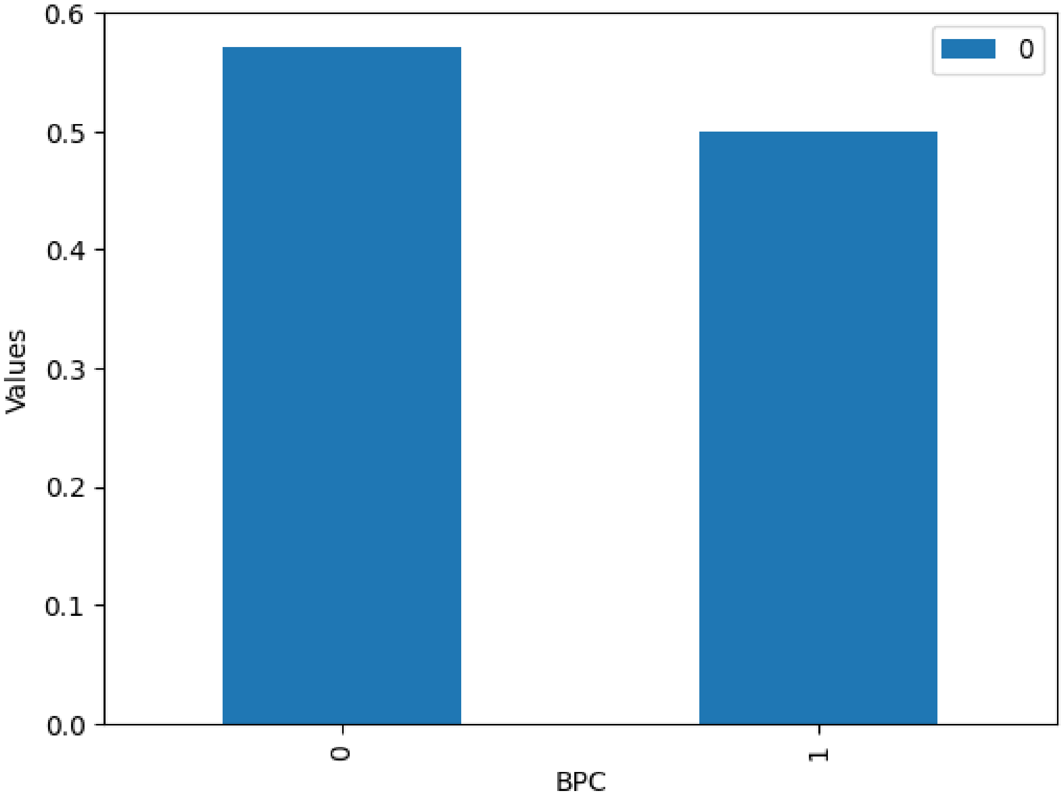
[ ]:

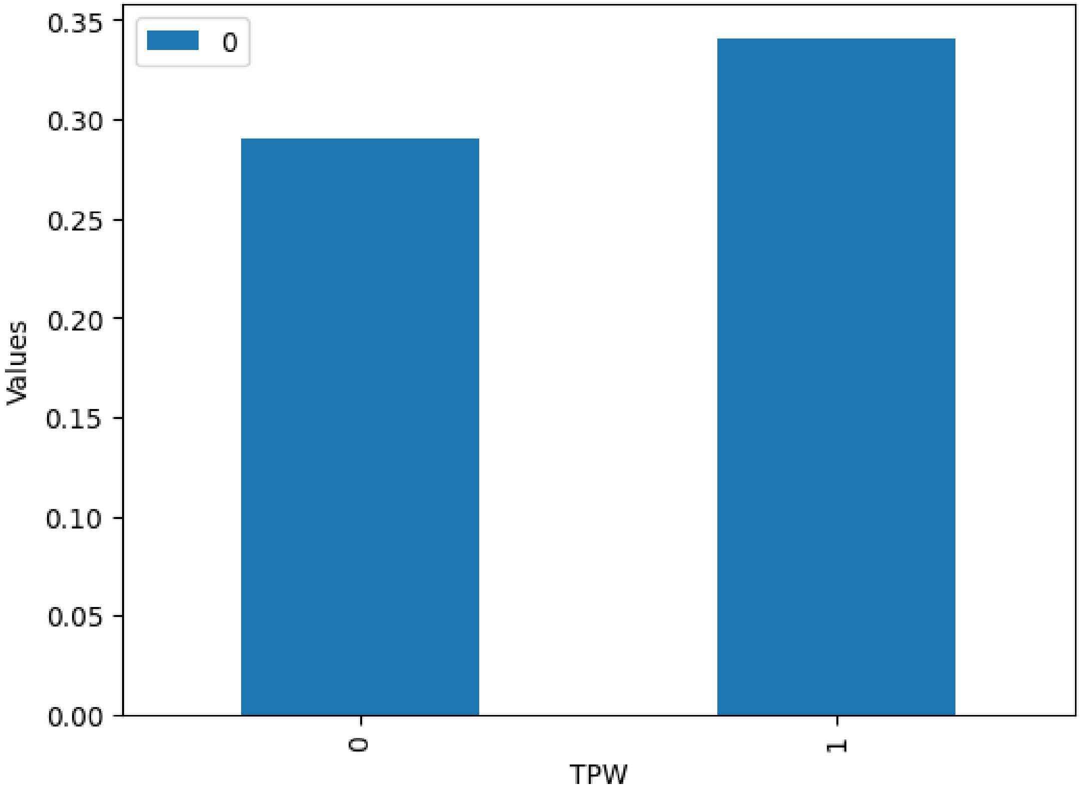
```
indices = [1, 3, 4, 8, 12, 6]
names = ['FSE', 'SSW', 'ACE', 'BPC', 'TPW', 'WNR']
count = 0
for i in indices:
    a = np.mean(Player[:, i])
    b = np.mean(Opponent[:, i])
    df = pd.DataFrame({a,b})
    df.plot(kind='bar')
    plt.xlabel(names[count])
    plt.ylabel('Values')
    plt.show()
    count = count + 1

indices = [5, 7, 10]
names = ['DBF', 'UFE', 'NPA']
count = 0
for i in indices:
    a = np.mean(Player[:, i])
    b = np.mean(Opponent[:, i])
    df = pd.DataFrame({a,b})
    df.plot(kind='bar')
    plt.xlabel(names[count])
    plt.ylabel('Values')
    plt.show()
    count = count + 1
```

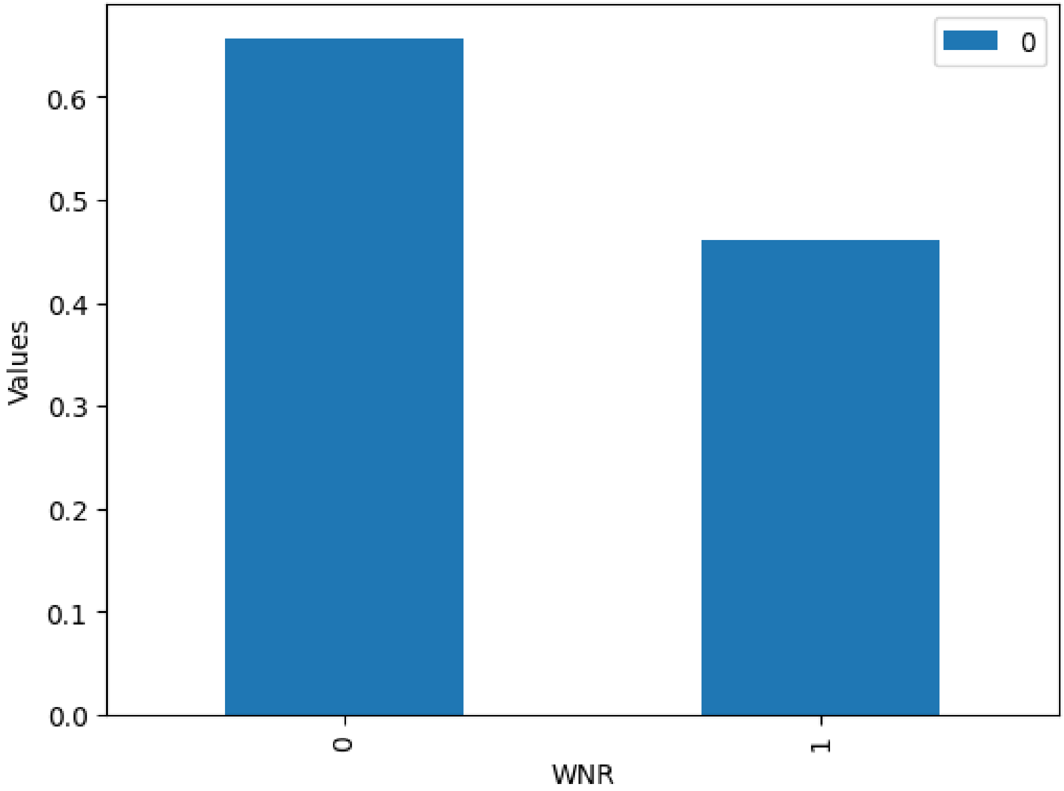


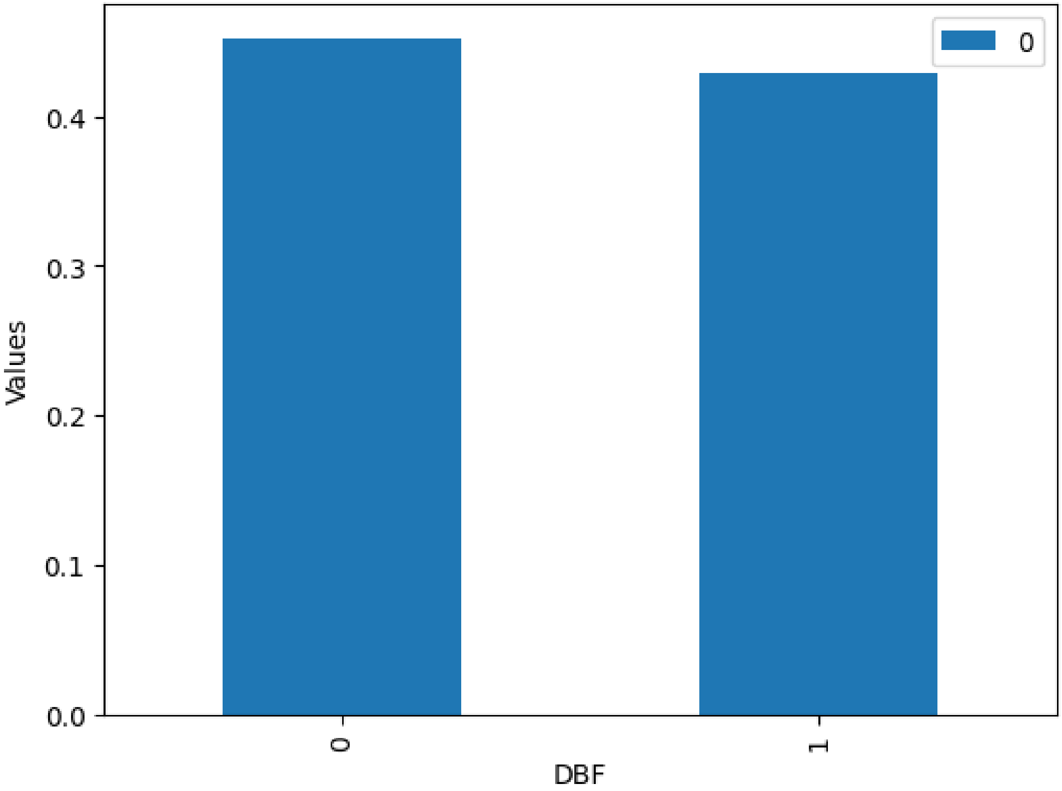

0.5
0.4
0.3
0.2
0.1
0.0
Values
0
1
SSW
0

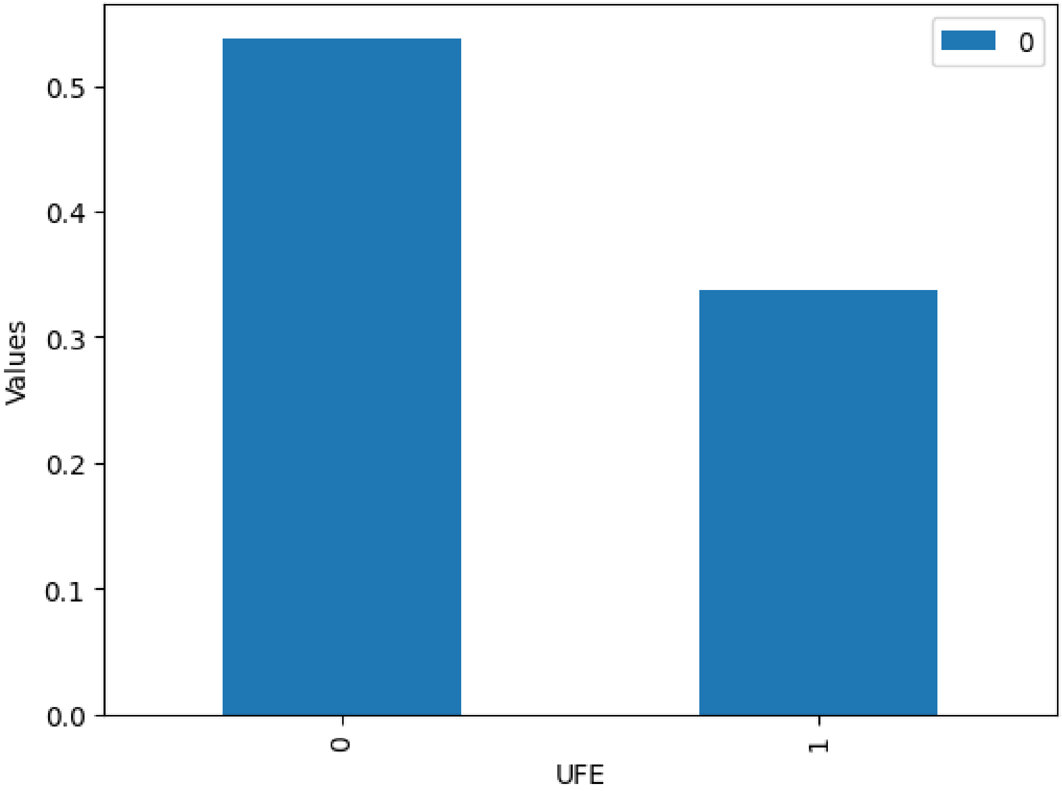

0
Values
0.4
0.3
0.2
0.1
0.0
0
1
ACE

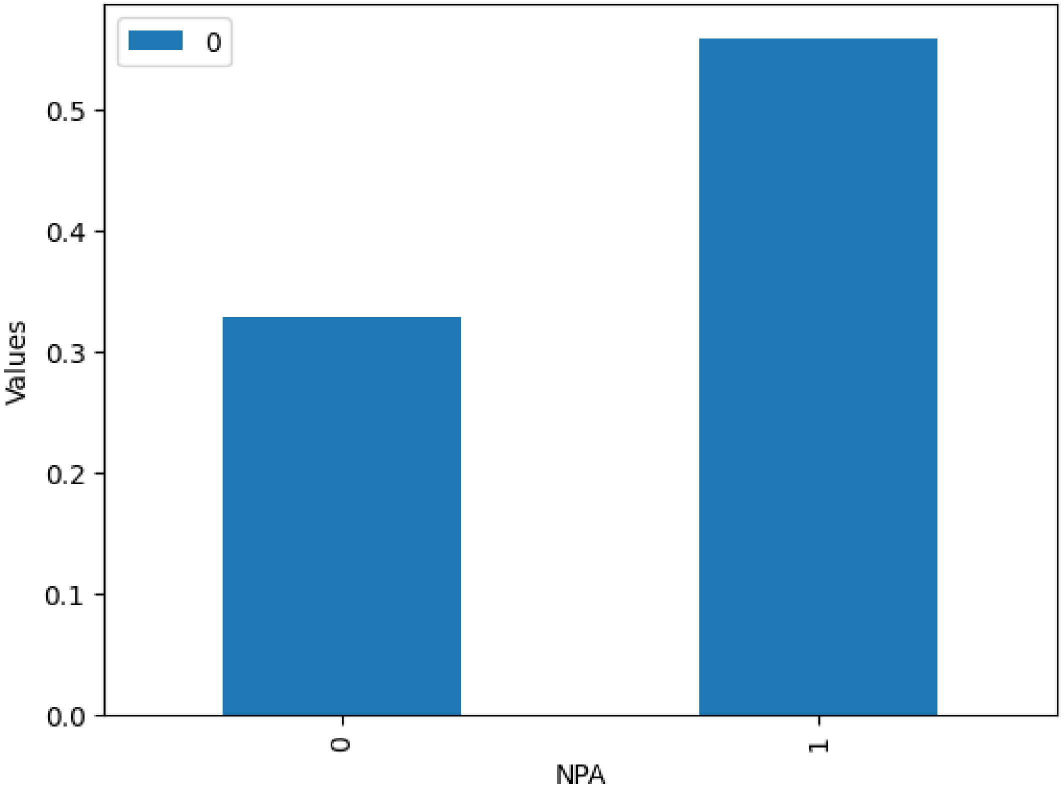

0.6
0.5
0.4
0.3
0.2
0.1
0.0
Values
0
1
BPC
0


0
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Values
0
1
TPW


0.6
0.5
0.4
0.3
0.2
0.1
0.0
Values
0
1
WNR
0


0
0.4
0.3
Values
0.2
0.1
0.0
0
1
DBF


0
0.5
0.4
0.3
0.2
0.1
0.0
Values
0
1
UFE



```
[ ]:
```

```
[ ]:
```

## 3 Q 3.

How can we make recommendations to a viewer based on preferable matches of that specific viewer.

---

```
[ ]: df3
```

```
[ ]:
                 Player1                  Player2  Round  Result  FNL.1  \
0    Pablo Carreno-Busta            Roger Federer      1       0      0
1       Somdev Devvarman  Daniel Munoz-De La Nava      1       1      3
2           Tobias Kamke            Paolo Lorenzi      1       1      3
3       Julien Benneteau        Ricardas Berankis      1       1      3
4            Lukas Lacko              Sam Querrey      1       0      0
..                   …                      …    …     …    …
120         Rafael Nadal       Stanislas Wawrinka      5       1      3
121       Novak Djokovic               Tommy Haas      5       1      3
```

```
122         David Ferrer       Jo-Wilfried Tsonga      6       1      3
123       Novak Djokovic             Rafael Nadal      6       0      2
124         Rafael Nadal             David Ferrer      7       1      3

     FNL.2  FSP.1  FSW.1  SSP.1  SSW.1  …  BPC.2  BPW.2  NPA.2  NPW.2  \
0        3     62     27     38     11  …      7      7     14     18
1        0     62     54     38     22  …      1     16     22     25
2        2     62     53     38     15  …     10     18     19     27
3        1     72     87     28     19  …      4     13     33     43
4        3     52     31     48     22  …      4      7     12     13
..     …    …    …    …    …  …    …    …    …    …
120      0     75     40     25     11  …      1      5     16     30
121      0     64     41     36     22  …      2      2      2     17
122      0     60     35     40     23  …      2      5      7     16
123      3     67     76     33     30  …      8     16     15     26
124      0     70     43     30     11  …      3     12     10     14

     TPW.2  ST1.2  ST2.2  ST3.2  ST4.2  ST5.2
0       88      6      6    6.0    0.0    0.0
1      106      3      3    5.0    0.0    0.0
2      139      3      3    6.0    6.0    3.0
3      149      6      3    7.0    6.0    0.0
4       93      6      6    6.0    0.0    0.0
..     …    …    …    …    …    …
120     64      2      3    1.0    0.0    0.0
121     84      3      6    5.0    0.0    0.0
122     84      1      6    2.0    0.0    0.0
123    177      6      3    6.0    6.0    9.0
124     72      3      2    3.0    0.0    0.0

[125 rows x 42 columns]
```

```
[ ]: new_df = df3.iloc[:, 2:].copy()
```

k means

```
[ ]: num_clusters = 5
     km = KMeans(num_clusters)
     predicted = km.fit_predict(new_df)
     predicted
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
```

[ ]:
```
array([0, 1, 4, 2, 1, 3, 2, 4, 0, 4, 3, 2, 1, 3, 2, 3, 0, 4, 3, 4, 1, 4,
       4, 3, 1, 1, 3, 2, 0, 3, 3, 2, 0, 4, 0, 1, 1, 4, 4, 4, 0, 3, 3, 1,
       1, 1, 3, 4, 3, 4, 4, 1, 1, 3, 3, 4, 0, 1, 4, 0, 1, 2, 1, 0, 2, 1,
       3, 0, 2, 1, 3, 0, 4, 4, 4, 1, 2, 4, 4, 0, 1, 4, 3, 4, 3, 4, 3, 2,
       4, 2, 3, 0, 1, 3, 0, 4, 4, 3, 1, 1, 1, 2, 1, 4, 4, 4, 3, 2, 1, 3,
       4, 3, 0, 2, 2, 3, 3, 4, 3, 0, 3, 3, 3, 2, 3], dtype=int32)
```

dividing data

[ ]:
```
listx1 = df3.values.tolist()

count=0
for i in predicted:
  listx1[count].append(i)
  count = count + 1

lis1=[]
lis2=[]
lis3=[]
lis4=[]
lis5=[]
j=0
i=0

for i in listx1:
  if i[42]==0:
    lis1.append(listx1[j])
  elif i[42]==1:
    lis2.append(listx1[j])
  elif i[42]==2:
    lis3.append(listx1[j])
  elif i[42]==3:
    lis4.append(listx1[j])
  elif i[42]==4:
    lis5.append(listx1[j])
  j= j + 1

arr1 = np.array(lis1)
arr2 = np.array(lis2)
arr3 = np.array(lis3)
arr4 = np.array(lis4)
arr5 = np.array(lis5)

df = pd.DataFrame(arr2)
print(df)
```

```
                         0                        1  2  3  4  5   6   7   8  \
0         Somdev Devvarman  Daniel Munoz-De La Nava  1  1  3  0  62  54  38
```

```
1              Lukas Lacko             Sam Querrey  1  0  0  3  52  31  48
2          Benjamin Becker           Jeremy Chardy  1  0  0  3  66  48  34
3          Illya Marchenko          Kevin Anderson  1  0  0  3  60  39  40
4    Andreas Haider-Maurer         Nicolas Almagro  1  0  1  3  65  50  35
5             Martin Alund  Edouard Roger-Vasselin  1  0  1  3  57  31  43
6            Florian Mayer           Denis Istomin  1  0  1  2  59  40  41
7             Albert Ramos         Jerzy Janowicz  1  0  0  3  61  41  39
8             Benoit Paire        Marcos Baghdatis  1  1  2  1  46  40  54
9             Andreas Beck           Fabio Fognini  1  0  0  3  55  37  45
10               Pere Riba             Lukas Rosol  1  0  0  3  60  38  40
11          Carlos Berlocq             John Isner  1  0  0  3  73  45  27
12       Andrey Kuznetsov          Ryan Harrison  1  0  0  3  71  48  29
13           Bernard Tomic         Victor Hanescu  1  0  0  3  63  38  37
14          Alex Kuznetsov          Lucas Pouille  1  0  0  3  63  33  37
15          Novak Djokovic           David Goffin  1  1  3  0  69  53  31
16               Jan Hajek            Sam Querrey  2  0  0  3  66  36  34
17   Roberto Bautista Agut          Jeremy Chardy  2  0  0  3  79  42  21
18  Edouard Roger-Vasselin        Nicolas Almagro  2  0  0  3  57  30  43
19           Denis Istomin      Nikolay Davydenko  2  0  0  3  68  41  32
20           Lucas Pouille        Grigor Dimitrov  2  0  0  3  60  31  40
21        Feliciano Lopez           David Ferrer  3  0  0  3  63  37  37
22            Milos Raonic         Kevin Anderson  3  0  0  3  57  49  43
23           Andreas Seppi        Nicolas Almagro  3  0  0  3  51  34  49
24       Nikolay Davydenko        Richard Gasquet  3  0  0  3  76  46  24
25          Victor Hanescu   Philipp Kohlschreiber  3  0  0  3  73  31  27
```

```
     9  …  33  34  35   36 37 38   39   40   41 42
0   22  …  16  22  25  106  3  3  5.0  0.0  0.0  1
1   22  …   7  12  13   93  6  6  6.0  0.0  0.0  1
2   13  …  11  10  16  108  6  6  7.0  0.0  0.0  1
3   25  …  12  12  14  115  6  7  6.0  0.0  0.0  1
4   20  …   7   6  12  113  4  6  6.0  6.0  0.0  1
5   17  …  19  16  21  111  6  4  6.0  6.0  0.0  1
6   15  …  11  17  26  101  4  6  7.0  0.0  0.0  1
7   25  …   7  19  22  113  7  7  6.0  0.0  0.0  1
8   29  …  14  13  25  104  6  6  3.0  0.0  0.0  1
9   25  …  13   9  11  125  6  7  6.0  0.0  0.0  1
10  14  …   7   9  12   97  6  6  6.0  0.0  0.0  1
11  10  …   9  16  27   97  6  6  6.0  0.0  0.0  1
12  11  …   8  12  18  111  6  6  7.0  0.0  0.0  1
13  17  …   5  14  25  103  7  7  2.0  0.0  0.0  1
14  13  …  11  10  14  109  6  7  6.0  0.0  0.0  1
15  20  …   2  15  26   99  6  4  5.0  0.0  0.0  1
16  16  …  12   9  15  107  6  7  6.0  0.0  0.0  1
17   8  …   8  21  28  101  6  7  6.0  0.0  0.0  1
18  16  …  12  11  14   95  6  6  6.0  0.0  0.0  1
19  19  …  14  15  21  102  6  7  6.0  0.0  0.0  1
20  14  …  10  17  19  103  6  7  6.0  0.0  0.0  1
```

```
21  14  …  19  14  20  111  6  7  6.0  0.0  0.0  1
22  29  …  12  10  12  126  7  7  6.0  0.0  0.0  1
23  18  …  13  12  14  107  7  6  6.0  0.0  0.0  1
24   9  …  10  17  31  101  6  6  6.0  0.0  0.0  1
25   8  …   9  19  32  105  6  7  6.0  0.0  0.0  1

[26 rows x 43 columns]
```

# 4 Q 4.

How can we choose players for a match so that the match becomes interesting and competitive.

---

```
[ ]: player1_df = df4.groupby('Player1').agg({
         'BPW.1': 'mean',
         'NPA.1': 'mean',
         'NPW.1': 'mean',
         'TPW.1': 'mean',
         'ST1.1': 'mean',
         'ST2.1': 'mean',
         'ST3.1': 'mean',
         'ST4.1': 'mean',
         'ST5.1': 'mean',
         'FSP.2': 'mean',
         'FSW.2': 'mean',
         'SSP.2': 'mean',
         'SSW.2': 'mean',
         'ACE.2': 'mean',
         'DBF.2': 'mean',
         'WNR.2': 'mean',
         'UFE.2': 'mean',
         'BPC.2': 'mean',
         'BPW.2': 'mean',
         'NPA.2': 'mean',
         'NPW.2': 'mean',
         'TPW.2': 'mean',
         'ST1.2': 'mean',
         'ST2.2': 'mean',
         'ST3.2': 'mean',
         'ST4.2': 'mean',
         'ST5.2': 'mean'
     }).reset_index().rename(columns={'Player1': 'Player'})

     player2_df = df4.groupby('Player2').agg({
         'BPW.1': 'mean',
         'NPA.1': 'mean',
         'NPW.1': 'mean',
         'TPW.1': 'mean',
         'ST1.1': 'mean',
         'ST2.1': 'mean',
         'ST3.1': 'mean',
         'ST4.1': 'mean',
         'ST5.1': 'mean',
         'FSP.2': 'mean',
         'FSW.2': 'mean',
         'SSP.2': 'mean',
         'SSW.2': 'mean',
         'ACE.2': 'mean',
         'DBF.2': 'mean',
         'WNR.2': 'mean',
         'UFE.2': 'mean',
         'BPC.2': 'mean',
         'BPW.2': 'mean',
         'NPA.2': 'mean',
         'NPW.2': 'mean',
         'TPW.2': 'mean',
         'ST1.2': 'mean',
         'ST2.2': 'mean',
         'ST3.2': 'mean',
         'ST4.2': 'mean',
         'ST5.2': 'mean'
     }).reset_index().rename(columns={'Player2': 'Player'})

     player_matches_df = pd.concat([player1_df, player2_df], ignore_index=True).
      ↪groupby('Player').sum().reset_index()
```

[ ]:
```
new_df = player_matches_df.iloc[:, 2:].copy()

new_clusters = 5
km = KMeans(new_clusters)
predicted3d = km.fit_predict(new_df)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
```

[ ]:
```
num_clusters = 5
km = KMeans(num_clusters)
predicted = km.fit_predict(new_df)
listx1 = player_matches_df.values.tolist()
```

```
count=0
for i in predicted:
  listx1[count].append(i)
  count = count + 1

lis1=[]
lis2=[]
lis3=[]
lis4=[]
lis5=[]
j=0
i=0

for i in listx1:
  if i[28]==0:
    lis1.append(listx1[j])
  elif i[28]==1:
    lis2.append(listx1[j])
  elif i[28]==2:
    lis3.append(listx1[j])
  elif i[28]==3:
    lis4.append(listx1[j])
  elif i[28]==4:
    lis5.append(listx1[j])
  j= j + 1

arr1 = np.array(lis1)
arr2 = np.array(lis2)
arr3 = np.array(lis3)
arr4 = np.array(lis4)
arr5 = np.array(lis5)

fig = go.Figure()

fig.add_trace(go.Scatter3d(x=arr1[:, 0], y=arr1[:, 1], z=arr1[:, 2],␣
 ↪marker=dict(size=5), mode='markers', name='cluster 1'))
fig.add_trace(go.Scatter3d(x=arr2[:, 0], y=arr2[:, 1], z=arr2[:, 2],␣
 ↪marker=dict(size=5), mode='markers', name='cluster 2'))
fig.add_trace(go.Scatter3d(x=arr3[:, 0], y=arr3[:, 1], z=arr3[:, 2],␣
 ↪marker=dict(size=5), mode='markers', name='cluster 3'))
fig.add_trace(go.Scatter3d(x=arr4[:, 0], y=arr4[:, 1], z=arr4[:, 2],␣
 ↪marker=dict(size=5), mode='markers', name='cluster 4'))
fig.add_trace(go.Scatter3d(x=arr5[:, 0], y=arr5[:, 1], z=arr5[:, 2],␣
 ↪marker=dict(size=5), mode='markers', name='cluster 5'))
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning:
```

```
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
```

[ ]:

```
from sklearn.decomposition import PCA

pc = PCA(n_components = 3)
reduced_df = pc.fit_transform(new_df)
```

[ ]:

```
listx1 = reduced_df.tolist()

count=0
for i in predicted3d:
  listx1[count].append(i)
  count = count + 1

lis1=[]
lis2=[]
lis3=[]
lis4=[]
lis5=[]
j=0
i=0

for i in listx1:
  if i[3]==0:
    lis1.append(listx1[j])
  elif i[3]==1:
    lis2.append(listx1[j])
  elif i[3]==2:
    lis3.append(listx1[j])
  elif i[3]==3:
    lis4.append(listx1[j])
  elif i[3]==4:
    lis5.append(listx1[j])
  j=j+1

arr1 = np.array(lis1)
arr2 = np.array(lis2)
arr3 = np.array(lis3)
arr4 = np.array(lis4)
arr5 = np.array(lis5)

fig = go.Figure()
```

```
fig.add_trace(go.Scatter3d(x=arr1[:, 0], y=arr1[:, 1], z=arr1[:, 2],␣
 ↪marker=dict(size=5), mode='markers', name='cluster 1'))
fig.add_trace(go.Scatter3d(x=arr2[:, 0], y=arr2[:, 1], z=arr2[:, 2],␣
 ↪marker=dict(size=5), mode='markers', name='cluster 2'))
fig.add_trace(go.Scatter3d(x=arr3[:, 0], y=arr3[:, 1], z=arr3[:, 2],␣
 ↪marker=dict(size=5), mode='markers', name='cluster 3'))
fig.add_trace(go.Scatter3d(x=arr4[:, 0], y=arr4[:, 1], z=arr4[:, 2],␣
 ↪marker=dict(size=5), mode='markers', name='cluster 4'))
fig.add_trace(go.Scatter3d(x=arr5[:, 0], y=arr5[:, 1], z=arr5[:, 2],␣
 ↪marker=dict(size=5), mode='markers', name='cluster 5'))

fig.show()
```

## 5 Q 5.

Can we group tennis players based on their playing style, and then use this information to predict which group is most likely to succeed against another group in a match?

---

[ ]:

```
player1 = df5.groupby('Player1').agg({
    'BPW.1': 'mean',
    'NPA.1': 'mean',
    'NPW.1': 'mean',
    'TPW.1': 'mean',
    'ST1.1': 'mean',
    'ST2.1': 'mean',
    'ST3.1': 'mean',
    'ST4.1': 'mean',
    'ST5.1': 'mean',
    'FSP.2': 'mean',
    'FSW.2': 'mean',
    'SSP.2': 'mean',
    'SSW.2': 'mean',
    'ACE.2': 'mean',
    'DBF.2': 'mean',
    'WNR.2': 'mean',
    'UFE.2': 'mean',
    'BPC.2': 'mean',
    'BPW.2': 'mean',
    'NPA.2': 'mean',
    'NPW.2': 'mean',
    'TPW.2': 'mean',
    'ST1.2': 'mean',
    'ST2.2': 'mean',
    'ST3.2': 'mean',
    'ST4.2': 'mean',
    'ST5.2': 'mean'
}).reset_index().rename(columns={'Player1': 'Player'})

player2 = df5.groupby('Player2').agg({
    'BPW.1': 'mean',
    'NPA.1': 'mean',
    'NPW.1': 'mean',
    'TPW.1': 'mean',
    'ST1.1': 'mean',
    'ST2.1': 'mean',
    'ST3.1': 'mean',
    'ST4.1': 'mean',
    'ST5.1': 'mean',
    'FSP.2': 'mean',
    'FSW.2': 'mean',
    'SSP.2': 'mean',
    'SSW.2': 'mean',
    'ACE.2': 'mean',
    'DBF.2': 'mean',
    'WNR.2': 'mean',
    'UFE.2': 'mean',
    'BPC.2': 'mean',
    'BPW.2': 'mean',
    'NPA.2': 'mean',
    'NPW.2': 'mean',
    'TPW.2': 'mean',
    'ST1.2': 'mean',
    'ST2.2': 'mean',
    'ST3.2': 'mean',
    'ST4.2': 'mean',
    'ST5.2': 'mean'
}).reset_index().rename(columns={'Player2': 'Player'})

player_matches = pd.concat([player1, player2], ignore_index=True).
 ↪groupby('Player').sum().reset_index()
# print(player_matches.to_markdown())
```

[ ]:

```
new_df = player_matches.iloc[:, 2:].copy()

new_clusters = 5
km = KMeans(new_clusters)
predicted = km.fit_predict(new_df)
predicted
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning:
```

```
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
```

```
[ ]: array([3, 2, 2, 2, 1, 0, 3, 4, 3, 4, 4, 4, 2, 3, 1, 0, 1, 4, 1, 2, 1, 3,
            2, 1, 4, 3, 3, 2, 2, 3, 3, 4, 3, 2, 1, 1, 2, 4, 1, 4, 2, 2, 2, 4,
            1, 3, 1, 4, 4, 3, 1, 3, 1, 1, 4, 0, 3, 4, 2, 4, 3, 2, 1, 2, 2, 4,
            4, 2, 1, 2, 4, 0, 1, 4, 2, 0, 3, 1, 2, 3, 2, 2, 4, 3, 0, 1, 2, 2,
            4, 4, 2, 2, 2, 2, 2, 4, 3, 2, 2, 2, 3, 4, 2, 1, 2, 4, 2, 4, 2, 1,
            1, 4, 1, 0, 3, 1, 2, 2, 2, 0, 4, 2, 2, 3, 4, 4, 4, 3], dtype=int32)
```

```
[ ]: player_matches['predicted'] = predicted
```

```
[ ]: player_matches['TPW'] = player_matches['TPW.1'] + player_matches['TPW.2']
```

```
[ ]: sorted_player_matches = player_matches.sort_values(by='TPW', ascending=False)
```

```
[ ]: f20 = sorted_player_matches.head(20)
     display(f20)
```

```
                    Player      BPW.1      NPA.1      NPW.1       TPW.1  \
5        Alex Bogomolov Jr.  37.000000  60.000000  82.000000  364.000000
119            Tim Smyczek  30.500000  55.500000  78.500000  313.000000
15          Bernard Tomic  26.000000  44.000000  63.000000  282.000000
29              Dudi Sela  31.000000   0.000000   0.000000  257.000000
75      Marcel Granollers  24.000000  44.500000  69.500000  276.000000
71         Lleyton Hewitt  29.000000  39.000000  64.000000  272.500000
8     Andreas Haider-Maurer  27.000000   0.000000   0.000000  250.000000
26           Denis Kudla  19.000000  12.000000  15.000000  226.000000
55       Janko Tipsarevic  20.666667  34.666667  57.000000  235.000000
0        Adrian Mannarino  16.000000  37.500000  57.000000  256.500000
60             Joao Sousa  23.500000  11.000000  15.000000  261.000000
84        Mikhail Youzhny  22.250000  39.000000  57.000000  242.000000
113     Stanislas Wawrinka  21.000000  37.500000  55.750000  244.500000
32         Evgeny Donskoy  22.000000   3.000000   7.000000  229.000000
56        Jarkko Nieminen  20.000000   0.000000   0.000000  224.000000
96    Philipp Kohlschreiber   4.333333  37.666667  59.333333  203.666667
49             Ivan Dodig  23.000000  21.500000  35.500000  222.000000
21          Daniel Evans  21.500000  36.000000  55.000000  235.500000
83      Mikhail Kukushkin  14.500000   7.500000  13.500000  205.500000
76       Marcos Baghdatis  26.000000  30.500000  38.000000  236.500000

         ST1.1      ST2.1      ST3.1      ST4.1  ST5.1  …  NPA.2  \
5    11.000000  12.000000  10.000000  11.000000   10.0  …   49.0
119   9.500000  12.000000  12.000000   8.500000    6.5  …   51.5
15    9.000000   9.000000  12.000000   9.000000    3.0  …   60.0
29   10.000000   7.000000   8.000000   7.000000    4.0  …    0.0
75    5.500000  10.000000  12.000000   9.000000   11.5  …   57.5
```

```
71   10.500000   9.000000  10.500000  10.000000    6.5  …   48.0
8    10.000000   7.000000  11.000000   6.000000    4.0  …    0.0
26    8.000000   8.000000  10.000000   5.000000    0.0  …   12.0
55   10.333333  11.666667   6.666667   6.666667    0.0  …   41.0
0    13.000000  12.000000  10.000000   6.500000    0.0  …   19.5
60    9.000000  10.500000  12.000000   6.500000    8.0  …    5.0
84   10.500000   6.250000  10.500000   8.250000    7.0  …   44.5
113   8.750000  11.250000   8.750000   6.250000    3.0  …   60.5
32    9.500000   9.000000  10.500000   2.500000    3.0  …   12.0
56   11.000000   8.000000   8.000000   3.000000    4.0  …    0.0
96    9.666667   7.666667   7.333333   3.666667    0.0  …   38.0
49    8.000000   9.500000  10.500000   6.000000    3.0  …   29.0
21   12.500000  10.500000  11.000000   5.000000    0.0  …   46.5
83    9.000000   9.000000  11.500000   2.000000    0.0  …    7.5
76   12.000000  12.000000  12.000000   7.000000    0.0  …   32.0

          NPW.2       TPW.2  ST1.2      ST2.2  ST3.2      ST4.2  ST5.2  \
5      86.000000  350.000000   10.0   9.000000   8.00  11.000000   13.0
119    85.000000  299.500000    9.0  10.000000   4.00   9.500000    9.5
15     81.000000  298.000000    7.0   9.000000  11.00  13.000000    6.0
29      0.000000  289.000000   13.0  12.000000  12.00   5.000000    6.0
75    100.500000  266.500000   12.0   9.500000   2.50   9.000000   11.5
71     79.500000  267.500000    8.0  13.000000   9.50  10.000000    3.0
8       0.000000  265.000000    9.0  12.000000   8.00   7.000000    6.0
26     24.000000  281.000000   13.0  13.000000  12.00   7.000000    0.0
55     51.666667  271.333333   12.0   9.333333  13.00   9.000000    0.0
0      33.000000  244.500000    9.5   9.000000  11.00   7.000000    0.0
60     10.500000  233.500000    6.5   7.000000   6.50  10.000000    5.0
84     77.500000  249.750000    8.5  11.000000  12.25   5.500000    5.0
113    92.000000  243.000000   11.0   9.000000  10.75   3.500000    2.0
32     15.500000  255.500000   11.5  11.500000  11.50   6.000000    1.5
56      0.000000  256.000000    8.0  13.000000   9.00   6.000000    6.0
96     49.333333  249.000000   12.0  11.000000  12.00  10.333333    0.0
49     49.500000  223.500000    9.5  12.000000   6.00   4.000000    6.0
21     71.000000  207.500000    7.5   7.500000   8.50   5.500000    0.0
83      9.500000  224.500000   11.0  12.000000  11.50   3.000000    0.0
76     51.500000  190.000000    6.0   4.500000   8.50   6.000000    0.0

     predicted         TPW
5            0  714.000000
119          0  612.500000
15           0  580.000000
29           3  546.000000
75           0  542.500000
71           0  540.000000
8            3  515.000000
26           3  507.000000
55           0  506.333333
```

```
0            3  501.000000
60           3  494.500000
84           0  491.750000
113          0  487.500000
32           3  484.500000
56           3  480.000000
96           3  452.666667
49           3  445.500000
21           3  443.000000
83           3  430.000000
76           3  426.500000

[20 rows x 30 columns]
```

## 6 Q 6.

Plot an 6d plot for common entries?

---

```
[ ]: import plotly.graph_objs as go
     from plotly.offline import iplot

     df6['FNL'] = df6['FNL.1'] + df6['FNL.1']

     markersize = df6['FNL']*10
     markercolor = df6['Round']

     plots = go.Scatter3d(x=df6['Player1'], y=df6['Player2'], z=df6['Result'],
         mode='markers',
         marker=dict(size=markersize, color=markercolor, colorscale='Blues',␣
      ↪opacity=0.8))

     fig = go.Figure(data=[plots])

     iplot(fig)
```

## 7 Q 7.

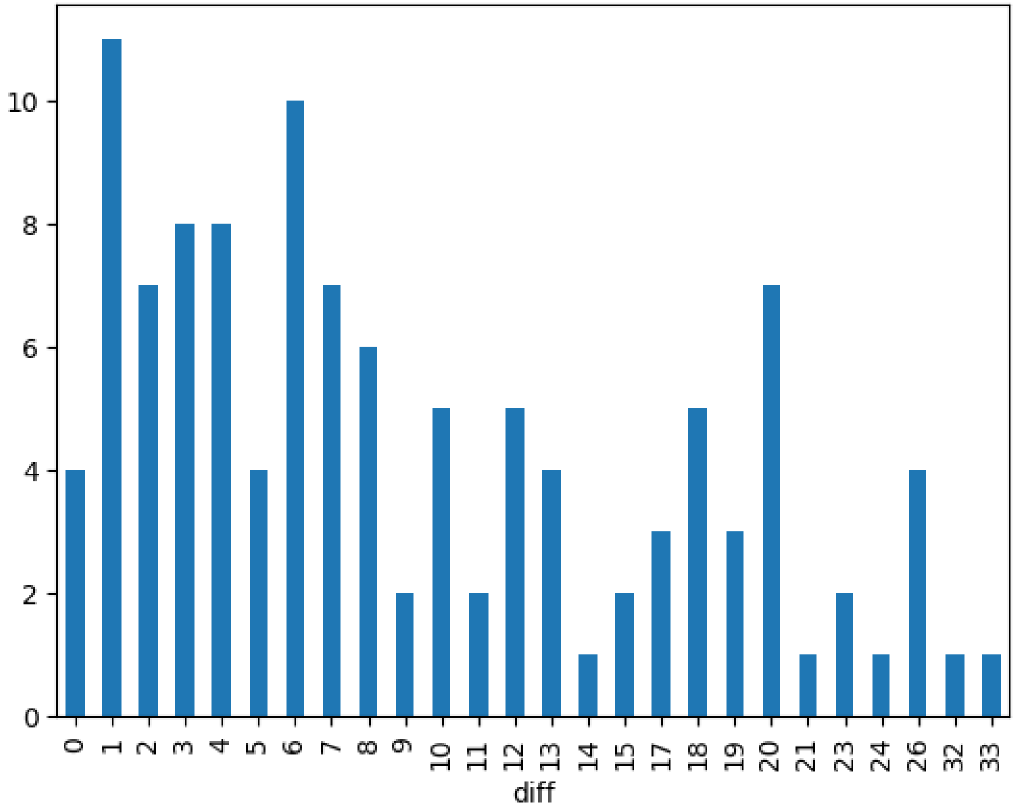
Can we calculate the number of tough matches and easiest matches?

---

```
[ ]: df7["diff"]=(df7["NPW.1"] - df7["NPW.2"]).abs()

     data=df7.groupby("diff")
     data=data["diff"]
     data=data.count()
```

```
data.plot(kind="bar")
```

```
[ ]: <Axes: xlabel='diff'>
```



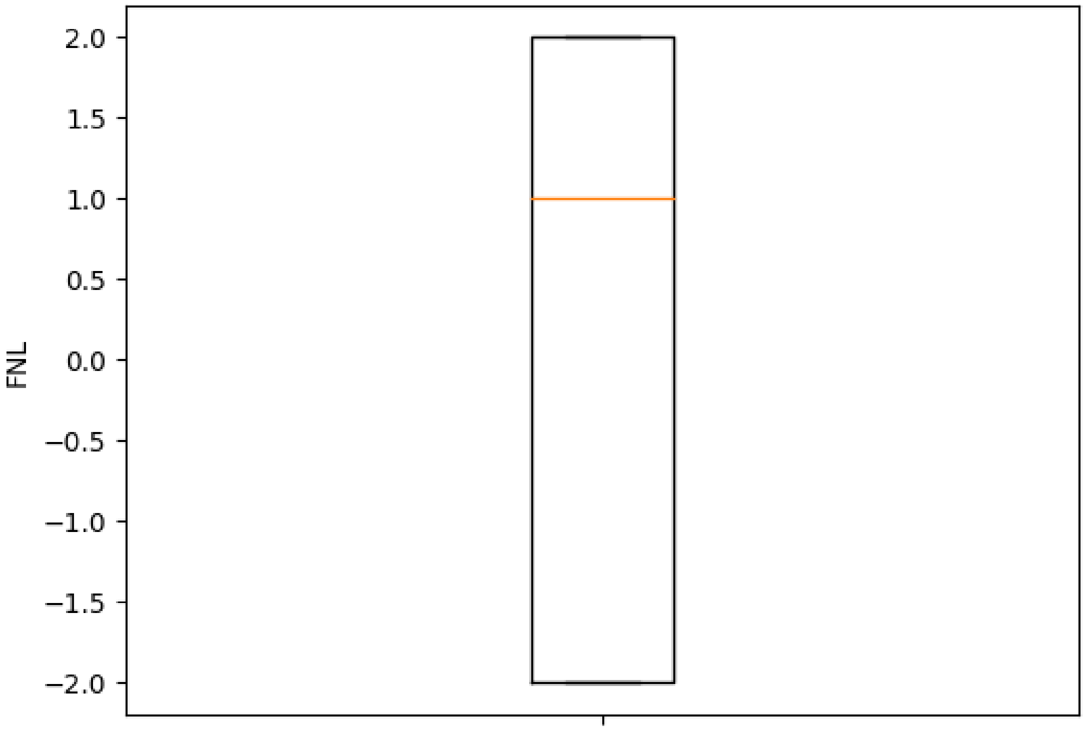
## 8 Q 8.

Is there a significant difference in the number of games won by the first player versus second player in a match?

---

```
[ ]: # df8.columns
```

```
[ ]: df8['FNL'] = df8['FNL.1'] - df8['FNL.2']
     fig, ax = plt.subplots()
     ax.boxplot(df8['FNL'])
     ax.set_ylabel('FNL')
     ax.set_xticklabels([''])
     plt.show()
```


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
FNL