Tweaking xGOT

By Siddharth Singh

In the dynamic and ever-evolving world of football (soccer), understanding player performance is critical to both fans and professionals alike. From casual enthusiasts to scouts and managers, statistical analysis has become an integral part of evaluating talent and making strategic decisions. With the 2022-2023 season offering a plethora of data, this project dives into a comprehensive analysis of players' performance across the top five European football leagues.

Utilizing the extensive dataset from fbref.com, the analysis presented in this project focuses on players who have scored more than 15 goals during the season. A key metric, known as the 'Expected Goals on Target' (xGOT), has been manually calculated by me to shed light on the efficiency and accuracy of these top-scoring players. The xGOT values serve as a powerful tool to evaluate how many goals a player is expected to score based on the quality of their shots on target.

By investigating this specific metric in conjunction with other statistical data, this study aims to uncover insights that may contribute to a deeper understanding of player performance, strategy development, and talent identification. Whether you're a football aficionado or a professional in the sport's industry, this analysis offers a fresh perspective on the factors that define success on the pitch in the 2022-2023 season, mainly around the statistic ‘Expected goals on target’.

In the analysis of football statistics, two vital models help us dissect the intricacies of goal-scoring opportunities: Expected Goals (xG) and Expected Goals on Target (xGOT).

Expected Goals (xG):

The xG model quantifies the quality of chances that a team creates. It represents a pre-shot measurement, evaluating the likelihood of a goal before the shot is taken. Factors such as the player's position, the angle of the shot, the type of pass leading to the shot, and the proximity of defenders can influence the xG value. Essentially, xG provides a statistical expression of how promising a scoring chance is.

Expected Goals on Target (xGOT):

Building on the foundation laid by xG, the xGOT model takes the analysis a step further by focusing exclusively on shots that are on target. As a post-shot model, xGOT measures the probability of a goal happening after the shot is taken.

The uniqueness of xGOT lies in its consideration of both the original xG of the shot and the goalmouth location where the shot ended up. Unlike xG, which primarily assesses the chance of a shot becoming a goal, xGOT also evaluates how well the shot was executed in terms of placement. Shots that end up in the corners are given more credit than shots aimed straight at the center of the goal. Since off-target shots inherently have a 0% chance of resulting in a goal, xGOT focuses only on on-target attempts.

Shooting Goals Added (SGA):

One practical application of these models is the calculation of Shooting Goals Added (SGA). SGA represents the difference between xGOT and xG for a player. If a player's xGOT exceeds their xG, it indicates that they are executing higher quality shots relative to the quality of chances they've attempted. Essentially, SGA allows us to gauge how well a player is shooting over a specific time frame.

Now that we are familiar with the stats used in the project, lets finally get to plotting some graphs!

In [20]:
```
#importing the neccessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import squarify

%matplotlib inline
```

In [21]:
```
#quick look at the dataset

xgot_unfiltered_df = pd.read_csv(r'C:\Users\siddh\OneDrive\Desktop\project dataset xgot.csv')
xgot_unfiltered_df.head()
```

Out[21]:

	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	90s	Gls	...	Dist	FK	PK	PKatt	xG	npxG	npxG/Sh	G-xG	np:G-xG	xGOT
0	1	Erling Haaland	no NOR	FW	Manchester City	eng Premier League	22	2000	30.8	36	...	12.6	0.0	7	7	28.4	23.1	0.20	7.6	5.9	29.0
1	8	Folarin Balogun	us USA	FW	Reims	fr Ligue 1	21	2001	33.3	21	...	15.2	2.0	6	7	26.6	21.0	0.18	-5.6	-6.0	23.6
2	3	Kylian Mbappé	fr FRA	FW	Paris S-G	fr Ligue 1	23	1998	31.3	29	...	15.9	1.0	3	5	26.3	22.2	0.15	2.7	3.8	26.7
3	7	Robert Lewandowski	pl POL	FW	Barcelona	es La Liga	33	1988	31.6	23	...	13.8	5.0	0	1	24.3	23.5	0.18	-1.3	-0.5	22.8
4	4	Alexandre Lacazette	fr FRA	FW	Lyon	fr Ligue 1	31	1991	32.5	27	...	14.4	8.0	6	8	24.1	17.8	0.17	2.9	3.2	23.1

5 rows × 27 columns

In [22]:
```
xgot_unfiltered_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 27 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Rk       99 non-null     int64
 1   Player   99 non-null     object
 2   Nation   99 non-null     object
 3   Pos      99 non-null     object
 4   Squad    99 non-null     object
 5   Comp     99 non-null     object
 6   Age      99 non-null     int64
 7   Born     99 non-null     int64
 8   90s      99 non-null     float64
 9   Gls      99 non-null     int64
 10  Sh       99 non-null     int64
 11  SoT      99 non-null     int64
 12  SoT%     99 non-null     float64
 13  Sh/90    99 non-null     float64
 14  SoT/90   99 non-null     float64
 15  G/Sh     99 non-null     float64
 16  G/SoT    99 non-null     float64
 17  Dist     98 non-null     float64
 18  FK       98 non-null     float64
 19  PK       99 non-null     int64
 20  PKatt    99 non-null     int64
 21  xG       98 non-null     float64
 22  npxG     98 non-null     float64
 23  npxG/Sh  98 non-null     float64
 24  G-xG     98 non-null     float64
 25  np:G-xG  98 non-null     float64
 26  xGOT     34 non-null     float64
dtypes: float64(14), int64(8), object(5)
```

To streamline the analysis and avoid monotonous manual data extraction, I have filtered the dataset to focus exclusively on players who scored more than 15 goals in the top five European leagues during the 2022-2023 season. This targeted approach enables us to concentrate on the most relevant data, saving time and providing a clear insight into the performance of the season's top goal-scorers.

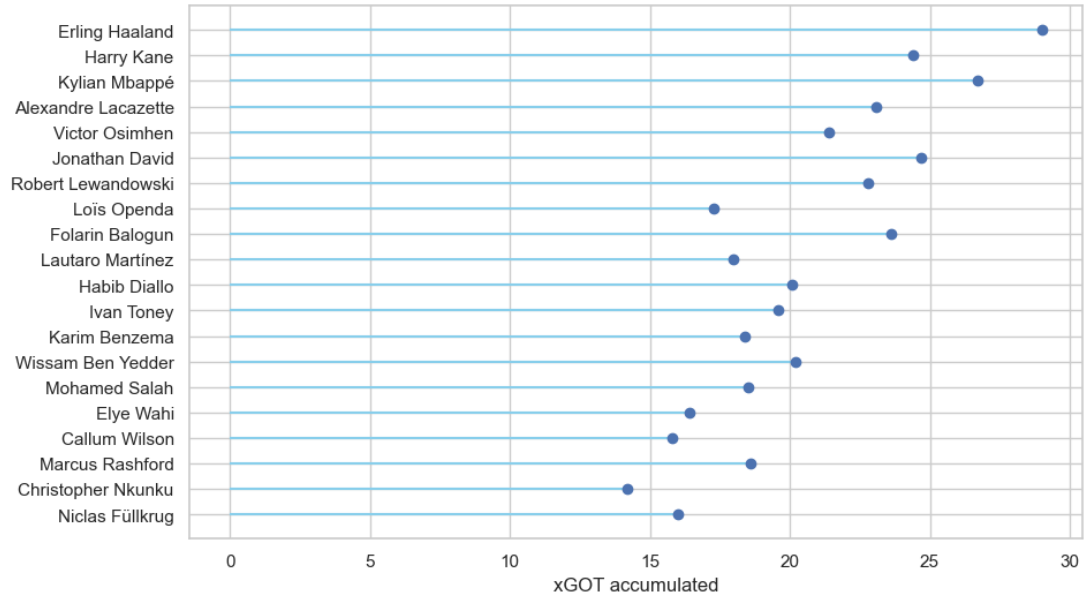
In [23]:
```
#filtering the data
xgot_df = xgot_unfiltered_df[xgot_unfiltered_df['Gls'] >= 15]

# Display the filtered DataFrame
xgot_df
```

Out[23]:

	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	90s	Gls	...	Dist	FK	PK	PKatt	xG	npxG	npxG/Sh	G-xG	np:G-xG	xGOT
0	1	Erling Haaland	no NOR	FW	Manchester City	eng Premier League	22	2000	30.8	36	...	12.6	0.0	7	7	28.4	23.1	0.20	7.6	5.9	29.0
1	8	Folarin Balogun	us USA	FW	Reims	fr Ligue 1	21	2001	33.3	21	...	15.2	2.0	6	7	26.6	21.0	0.18	-5.6	-6.0	23.6
2	3	Kylian Mbappé	fr FRA	FW	Paris S-G	fr Ligue 1	23	1998	31.3	29	...	15.9	1.0	3	5	26.3	22.2	0.15	2.7	3.8	26.7
3	7	Robert Lewandowski	pl POL	FW	Barcelona	es La Liga	33	1988	31.6	23	...	13.8	5.0	0	1	24.3	23.5	0.18	-1.3	-0.5	22.8
4	4	Alexandre Lacazette	fr FRA	FW	Lyon	fr Ligue 1	31	1991	32.5	27	...	14.4	8.0	6	8	24.1	17.8	0.17	2.9	3.2	23.1
5	6	Jonathan David	ca CAN	FW	Lille	fr Ligue 1	22	2000	35.2	24	...	15.8	0.0	10	11	23.6	15.1	0.16	0.4	-1.1	24.7
6	15	Mohamed Salah	eg EGY	FW	Liverpool	eng Premier League	30	1992	36.6	19	...	15.1	2.0	2	4	21.7	18.5	0.15	-2.7	-1.5	18.5
7	2	Harry Kane	eng ENG	FW	Tottenham	eng Premier League	29	1993	37.8	30	...	16.0	3.0	5	6	21.5	16.7	0.13	8.5	8.3	24.4
8	14	Karim Benzema	fr FRA	FW	Real Madrid	es La Liga	34	1987	22.6	19	...	15.9	6.0	7	8	21.5	14.9	0.15	-2.5	-2.9	18.4
9	5	Victor Osimhen	ng NGA	FW	Napoli	it Serie A	23	1998	28.5	26	...	12.8	1.0	2	3	21.3	18.9	0.14	4.7	5.1	21.4
10	12	Ivan Toney	eng ENG	FW	Brentford	eng Premier League	26	1996	32.8	20	...	16.5	9.0	6	7	18.7	13.2	0.15	1.3	0.8	19.6
11	10	Loïs Openda	be BEL	FW	Lens	fr Ligue 1	22	2000	28.0	21	...	13.1	0.0	1	1	18.5	17.7	0.17	2.5	2.3	17.3
12	9	Lautaro	ar ARG	FW	Inter	it Serie A	24	1997	28.6	21	...	15.1	0.0	1	2	18.1	16.5	0.14	2.9	3.5	18.0

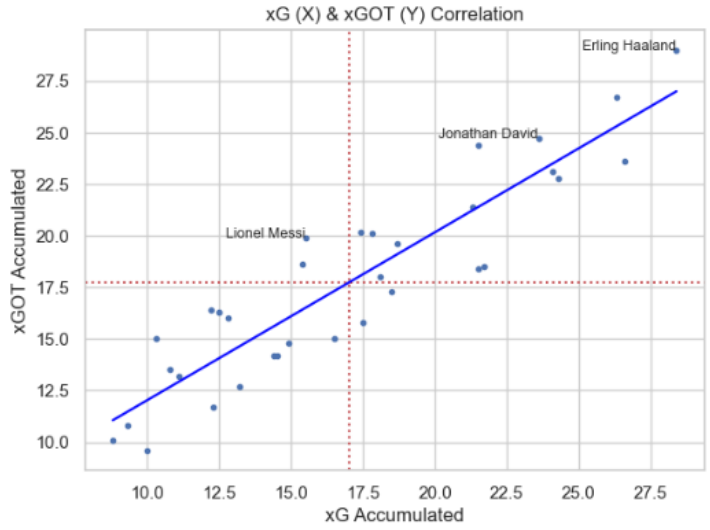
Let’s now look at the players with the most xGOT accumulated arranged in ascending order of goals scored in the 22-23 league season.

```
In [24]: plt.figure(figsize=(10, 6))

         # Sort the DataFrame by the 'Gls' column in descending order
         xgot_df_sortedbyxGOT = xgot_df.sort_values('Gls', ascending=False)

         # Then use slicing to select the top 20 rows
         xgot_df_sortedbyxGOT_top20 = xgot_df_sortedbyxGOT[:20]

         # Plotting the data
         plt.hlines(y=np.arange(1,21), xmin=0, xmax=xgot_df_sortedbyxGOT_top20['xGOT'][:20], color="skyblue")
         plt.plot(xgot_df_sortedbyxGOT_top20['xGOT'][:20], np.arange(1,21), "o")
         plt.yticks(np.arange(1,21), xgot_df_sortedbyxGOT_top20["Player"][:20])
         plt.gca().invert_yaxis()  # This line inverts the y-axis so the player with the most goals is at the top
         plt.xlabel('xGOT accumulated')
         plt.show()
```



The graph reveals intriguing disparities between players in terms of xGOT accumulation. Interestingly, some players have managed to score more goals despite accumulating a lower xGOT value. This unexpected finding hints at complexities in the relationship between expected and actual performance, a topic we will delve into further in the upcoming plots, but first lets compare these 2 expected statistics and find their correlation and what factors they depend on.

In [25]:
```
# Fit a linear regression line to the 'xG' and 'xGOT' data
slope, intercept = np.polyfit(xgot_df['xG'], xgot_df['xGOT'], 1)

# Generate the x values for the trendline
x_values = np.linspace(xgot_df['xG'].min(), xgot_df['xG'].max(), 100)

# Calculate the corresponding y values using the slope and intercept
y_values = slope * x_values + intercept

# Now you can create your scatterplot with this new DataFrame
fig, ax = plt.subplots()
fig.set_size_inches(7, 5)

plt.plot(xgot_df['xG'], xgot_df['xGOT'], ".")
plt.axvline(xgot_df['xG'].mean(), linestyle=':', color='r')  # Vertical line at the mean of 'Gls'
plt.axhline(y=xgot_df['xGOT'].mean(), linestyle=':', color='r')  # Horizontal line at the mean of 'MP'

plt.plot(x_values, y_values, "-", color="blue")  # Plot the trendline

ax.set_title("xG (X) & xGOT (Y) Correlation")
ax.set_xlabel("xG Accumulated")
ax.set_ylabel("xGOT Accumulated")

highlighted_players = ["Erling Haaland", "Lionel Messi" , "Jonathan David"]

for index, row in xgot_df.iterrows():
    if row['Player'] in highlighted_players:
        plt.text(row['xG'], row['xGOT'], row['Player'], fontsize=9, ha='right')

plt.show()
```



The proximity to the goal is a crucial factor in determining both xGOT and xG values, reflecting the underlying complexity of scoring predictions. Typically, a shot taken a mere 3 meters from the goal is expected to register a higher xGOT compared to a shot from 19 meters out. Exceptions, however, are part and parcel of the game, showcasing the multifaceted aspects of these statistical models. A player such as Erling Haaland, renowned as a "fox in the box" and a classic number 9 with goal poaching instincts, often finds himself in close proximity to the goal. This positioning allows him to accumulate higher xG and xGOT values. Factors such as the shot's placement in the corner or its speed further contribute to these higher values, offering a nuanced view of a striker's efficiency and threat.

It's essential to recognize that for certain players, such as Jonathan David, or those who frequently take penalties, achieving an xGOT value of 1.0 (the maximum) is not uncommon. This can skew the model's accuracy, as these values may be artificially inflated due to penalty shots, which typically have a high likelihood of success. To provide a more nuanced and accurate analysis, we must adjust for this by isolating xGOT values derived from open play. This adjustment involves removing the xGOT contributions from penalties and can be calculated using the following formula:

xGOTop (Open Play) = xGOT – {PK (Goals Scored from the Penalty Spot) * 0.975 (estimated xGOT average)}

By implementing this correction, we aim to offer a more genuine reflection of a player's xGOT performance, focusing solely on actions from open play and thereby minimizing the distortion caused by penalty kicks.

In [26]:
```
xgot_unfiltered_df['xGOTop'] = xgot_unfiltered_df['xGOT'] - xgot_unfiltered_df['PK'] * 0.975
xgot_unfiltered_df['xGOT_difference'] = xgot_unfiltered_df['xGOT'] - xgot_unfiltered_df['xGOTop']
```

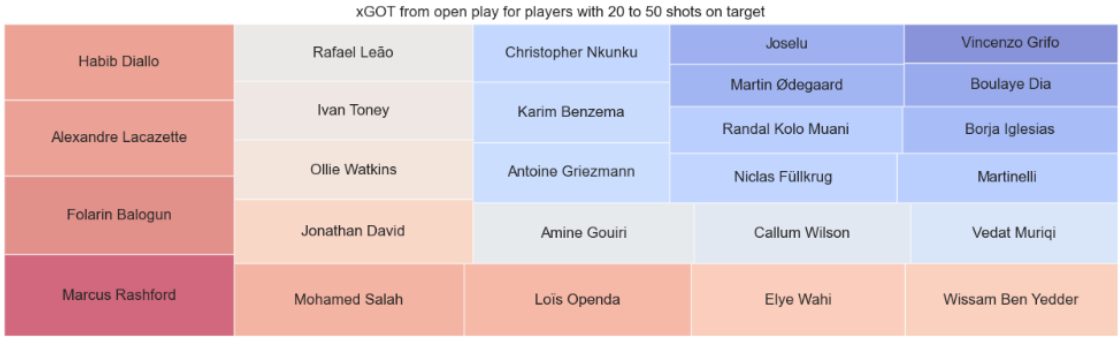
In [27]:
```
xgot_df = xgot_unfiltered_df[xgot_unfiltered_df['Gls'] >= 15]
xgot_df
```

Out[27]:

	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	90s	Gls	...	PK	PKatt	xG	npxG	npxG/Sh	G-xG	np:G-xG	xGOT	xGOTop	xGOT_d
0	1	Erling Haaland	no NOR	FW	Manchester City	eng Premier League	22	2000	30.8	36	...	7	7	28.4	23.1	0.20	7.6	5.9	29.0	22.175	
1	8	Folarin Balogun	us USA	FW	Reims	fr Ligue 1	21	2001	33.3	21	...	6	7	26.6	21.0	0.18	-5.6	-6.0	23.6	17.750	
2	3	Kylian Mbappé	fr FRA	FW	Paris S-G	fr Ligue 1	23	1998	31.3	29	...	3	5	26.3	22.2	0.15	2.7	3.8	26.7	23.775	
3	7	Robert Lewandowski	pl POL	FW	Barcelona	es La Liga	33	1988	31.6	23	...	0	1	24.3	23.5	0.18	-1.3	-0.5	22.8	22.800	
4	4	Alexandre Lacazette	fr FRA	FW	Lyon	fr Ligue 1	31	1991	32.5	27	...	6	8	24.1	17.8	0.17	2.9	3.2	23.1	17.250	
5	6	Jonathan David	ca CAN	FW	Lille	fr Ligue 1	22	2000	35.2	24	...	10	11	23.6	15.1	0.16	0.4	-1.1	24.7	14.950	
6	15	Mohamed Salah	eg EGY	FW	Liverpool	eng Premier League	30	1992	36.6	19	...	2	4	21.7	18.5	0.15	-2.7	-1.5	18.5	16.550	
7	2	Harry Kane	eng ENG	FW	Tottenham	eng Premier League	29	1993	37.8	30	...	5	6	21.5	16.7	0.13	8.5	8.3	24.4	19.525	
8	14	Karim Benzema	fr FRA	FW	Real Madrid	es La Liga	34	1987	22.6	19	...	7	8	21.5	14.9	0.15	-2.5	-2.9	18.4	11.575	
9	5	Victor Osimhen	ng NGA	FW	Napoli	it Serie A	23	1998	28.5	26	...	2	3	21.3	18.9	0.14	4.7	5.1	21.4	19.450	
			eng			eng															

With our refined stat in hand, it's time to explore further the players' performance in terms of expected goals from the target, offering a more detailed and refined understanding of their capabilities on the pitch.

In [28]:
```
import matplotlib
import matplotlib.pyplot as plt
import squarify

# Filter to include only players with shots between 50 and 75 in the English Premier League
xGOTop_treemap = xgot_df[(xgot_df["SoT"] >= 20) & (xgot_df["SoT"] <= 50) & (xgot_df["xGOTop"] > 0)]

# Sort the data by xGOTop in descending order
xGOTop_treemap = xGOTop_treemap.sort_values(by='xGOTop', ascending=False)

# Utilize matplotlib to scale our numbers between the min and max, then assign this scale to our values.
norm = matplotlib.colors.Normalize(vmin=min(xGOTop_treemap.xGOTop), vmax=max(xGOTop_treemap.xGOTop))
colors = [matplotlib.cm.coolwarm(norm(value)) for value in xGOTop_treemap.xGOTop]  # You can change the color map here

# Create our plot and resize it.
fig = plt.gcf()
ax = fig.add_subplot()
fig.set_size_inches(16, 4.5)

# Use squarify to plot our data, label it, and add colors. We add an alpha layer to ensure black labels show through
squarify.plot(label=xGOTop_treemap.Player, sizes=xGOTop_treemap.xGOTop, color=colors, alpha=.6)
plt.title("xGOT from open play for players with 20 to 50 shots on target")

plt.axis('off')
plt.show()
```

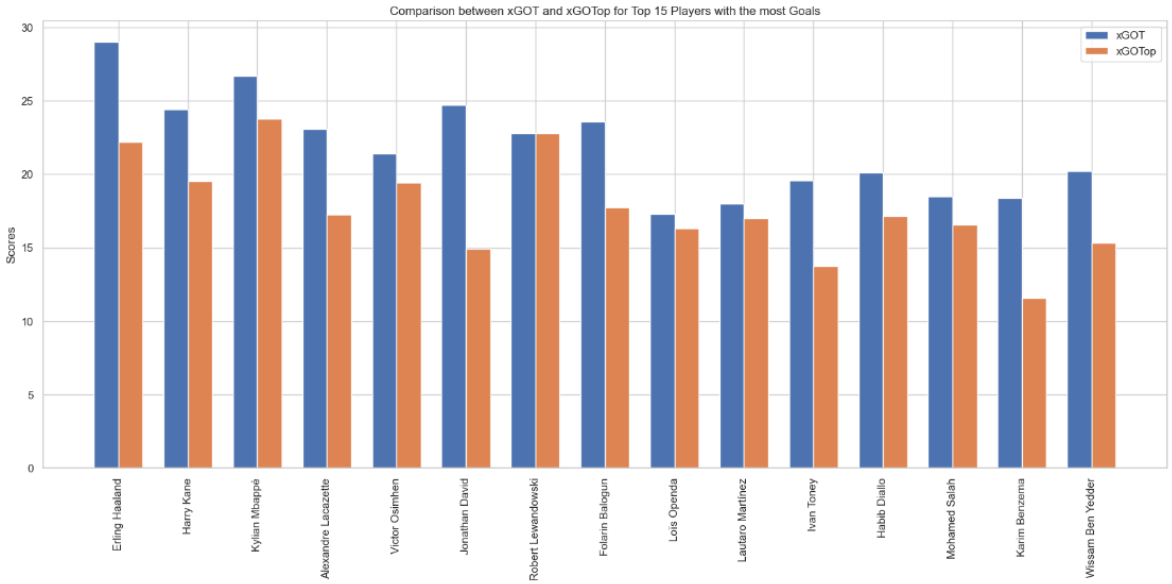


Observe how certain players in the top right of our analysis occupy a more confined space, despite falling within a similar "shot on target" range as others. This phenomenon can be attributed to their lower xGOTop value, as these players are often midfielders or occupy positions across the field rather than just the confinements of the penalty box, resulting in a smaller xGOT value.

Consider Grifo, who had the longest average shot distance from the goal and consequently appears far to the right in our analysis. While this correlation often holds true, there are always anomalies. Take Rashford and Callum Wilson as an illustration. With an average shot distance from goal at 16.4 for Rashford and a mere 12.4 for Wilson, one might expect Rashford's xGOT to be lesser. Surprisingly, Rashford's xGOT from open play surpassed Wilson's, a testament to his superior shot quality, a topic we'll delve into shortly.

Let's cast a concluding glance at the disparity between xGOTop and xGOT values for specific players. This examination will help us pinpoint those who have notably benefitted from their roles as habitual penalty takers for their respective teams, consequently leading to an augmented accumulation of xGOT.

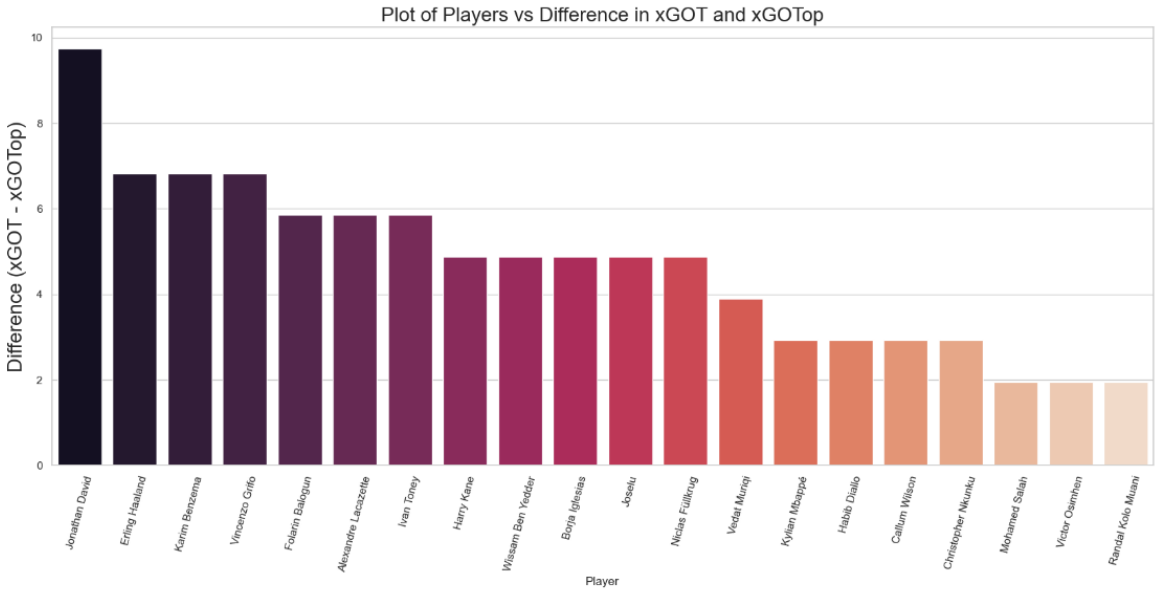
In [29]:
```
import numpy as np
import matplotlib.pyplot as plt

# Assuming xgot_df is your DataFrame containing player statistics
# Get the top 15 players with the most goals (Gls)
top_players = xgot_df.nlargest(15, 'Gls')

# Extract player names for labeling the x-axis
labels = top_players['Player']

# Create an array of x-values for the bar positions
x = np.arange(len(labels))

# Set the width of the bars
width = 0.35

# Create a new figure and axis for the bar plot
fig, ax = plt.subplots()

# Create the first set of bars for xGOT values
rects1 = ax.bar(x - width/2, top_players['xGOT'], width, label='xGOT')

# Create the second set of bars for xGOTop values
rects2 = ax.bar(x + width/2, top_players['xGOTop'], width, label='xGOTop')

# Set labels and title for the plot
ax.set_ylabel('Scores')
ax.set_title('Comparison between xGOT and xGOTop for Top 15 Players with the most Goals')

# Set the tick positions and labels on the x-axis
ax.set_xticks(x)
ax.set_xticklabels(labels, rotation=90)

# Add a legend to the plot
ax.legend()

# Display the plot
plt.show()
```



From the graph, we can discern that Jonathan David, Karim Benzema, and Erling Haaland exhibited the greatest disparity between overall xGOT and xGOT from open play among the top goal scorers of last season. This distinction largely stems from their proficiency and efficiency as penalty kick takers. Take note of Robert Lewandowski's bar graphs, which are identical in height, reflecting his tally of 0 penalty goals, a contrast to other players on the list. Our next visualization focuses on a bar plot, highlighting

players who have the maximum difference between these two statistics. This will allow us to isolate the players who benefitted the most from an inflation in xGOT.

In [30]:
```
# Import necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming xgot_df is your DataFrame containing player statistics
# Get the top 20 players with the most difference between xGOT and xGOTop
top_difference_players = xgot_df.nlargest(20, 'xGOT_difference')

# Set the theme and style for the plot using seaborn
sns.set_theme(style="whitegrid", color_codes=True)

# Create a bar plot using seaborn
# x-axis: Player names, y-axis: Difference between xGOT and xGOTop
# Use 'rocket' palette for color variation
ax = sns.barplot(x='Player', y='xGOT_difference', data=top_difference_players, palette='rocket')

# Set y-axis label
ax.set_ylabel("Difference (xGOT - xGOTop)", fontsize=20)

# Rotate x-axis labels for better readability
plt.xticks(rotation=75)

# Set the figure size
plt.rcParams["figure.figsize"] = (20, 8)

# Set the title for the plot
plt.title('Plot of Players vs Difference in xGOT and xGOTop', fontsize=20)

# Display the plot
plt.show()
```



With xGOT now correlated to various factors and its underlying elements thoroughly examined, we're poised to delve deeper into a specialized metric we've previously mentioned: 'Shot Quality,' represented as xGOTop (Open Play) / Shots on Target. Since xGOT from open play quantifies the likelihood of an on-target shot resulting in a goal, dividing this by the total shots on target reveals the intrinsic quality of a shot. And since shots on target don't encompass penalty kicks, we can confidently divide our xGOTop by shots on target without needing to subtract the penalty kicks from the equation.

A heightened shot quality value may primarily manifest due to a player's superior shooting skills, encompassing precision and power when faced with the goal. Alternatively, it could be the result of a low shots-on-target figure coupled with an inflated xGOT from open play. Both scenarios underscore the same underlying theme: the player's remarkable skill in transforming opportunities into goals, a testament to their shooting prowess.

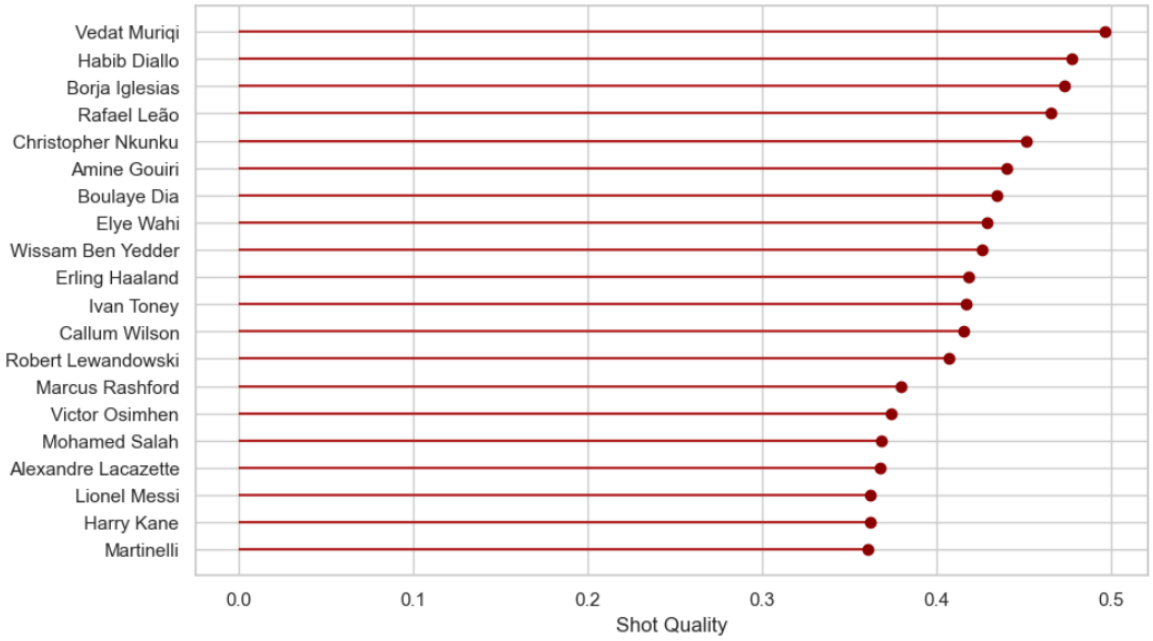
In [31]:
```
xgot_unfiltered_df['Shot_Quality'] = xgot_unfiltered_df['xGOTop'] / xgot_unfiltered_df['SoT']
xgot_df = xgot_unfiltered_df[xgot_unfiltered_df['Gls'] >= 15]
xgot_df
```

Out[31]:

	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	90s	Gls	...	PKatt	xG	npxG	npxG/Sh	G-xG	np:G-xG	xGOT	xGOTop	xGOT_differe
0	1	Erling Haaland	no NOR	FW	Manchester City	eng Premier League	22	2000	30.8	36	...	7	28.4	23.1	0.20	7.6	5.9	29.0	22.175	6
1	8	Folarin Balogun	us USA	FW	Reims	fr Ligue 1	21	2001	33.3	21	...	7	26.6	21.0	0.18	-5.6	-6.0	23.6	17.750	5
2	3	Kylian Mbappé	fr FRA	FW	Paris S-G	fr Ligue 1	23	1998	31.3	29	...	5	26.3	22.2	0.15	2.7	3.8	26.7	23.775	2
3	7	Robert Lewandowski	pl POL	FW	Barcelona	es La Liga	33	1988	31.6	23	...	1	24.3	23.5	0.18	-1.3	-0.5	22.8	22.800	0
4	4	Alexandre Lacazette	fr FRA	FW	Lyon	fr Ligue 1	31	1991	32.5	27	...	8	24.1	17.8	0.17	2.9	3.2	23.1	17.250	5
5	6	Jonathan David	ca CAN	FW	Lille	fr Ligue 1	22	2000	35.2	24	...	11	23.6	15.1	0.16	0.4	-1.1	24.7	14.950	9
6	15	Mohamed Salah	eg EGY	FW	Liverpool	eng Premier League	30	1992	36.6	19	...	4	21.7	18.5	0.15	-2.7	-1.5	18.5	16.550	1
7	2	Harry Kane	eng ENG	FW	Tottenham	eng Premier League	29	1993	37.8	30	...	6	21.5	16.7	0.13	8.5	8.3	24.4	19.525	4
8	14	Karim Benzema	fr FRA	FW	Real Madrid	es La Liga	34	1987	22.6	19	...	8	21.5	14.9	0.15	-2.5	-2.9	18.4	11.575	6
9	5	Victor Osimhen	ng NGA	FW	Napoli	it Serie A	23	1998	28.5	26	...	3	21.3	18.9	0.14	4.7	5.1	21.4	19.450	1
10	12	Ivan Toney	eng ENG	FW	Brentford	eng Premier League	26	1996	32.8	20	...	7	18.7	13.2	0.15	1.3	0.8	19.6	13.750	5
11	10	Loïs Openda	be BEL	FW	Lens	fr Ligue 1	22	2000	28.0	21	...	1	18.5	17.7	0.17	2.5	2.3	17.3	16.325	0
12	9	Lautaro Martínez	ar ARG	FW	Inter	it Serie A	24	1997	28.6	21	...	2	18.1	16.5	0.14	2.9	3.5	18.0	17.025	0

Lets now take a glance at players in the top 5 leagues in 22-23 season with the highest shot quality or xGOTop/Shot on Target

In [32]:
```python
import matplotlib.pyplot as plt
import numpy as np

# Set the figure size for the plot
plt.figure(figsize=(10, 6))

# Sort the DataFrame by the 'Shot_Quality' column in descending order
xgot_df_sortedbyShot_Quality = xgot_df.sort_values('Shot_Quality', ascending=False)

# Select the top 20 rows based on sorted 'Shot_Quality'
xgot_df_sortedbyShot_Quality_top20 = xgot_df_sortedbyShot_Quality[:20]

# Creating a horizontal bar plot
# Horizontal lines indicating player ranks, x-axis represents Shot Quality
# Color of lines: firebrick
plt.hlines(y=np.arange(1, 21), xmin=0, xmax=xgot_df_sortedbyShot_Quality_top20['Shot_Quality'][:20], color="firebrick")

# Plotting individual data points as circles
# x-axis: Shot Quality, y-axis: Player ranks
# Color of circles: darkred
plt.plot(xgot_df_sortedbyShot_Quality_top20['Shot_Quality'][:20], np.arange(1, 21), "o", color="darkred")

# Setting y-axis tick positions and labels
plt.yticks(np.arange(1, 21), xgot_df_sortedbyShot_Quality_top20["Player"][:20])

# Invert y-axis to have the player with the most goals at the top
plt.gca().invert_yaxis()

# Set x-axis label
plt.xlabel('Shot Quality')

plt.show()
```



This graph illustrates the top 20 players boasting the highest shot quality. A shot quality of 0.45, for example, can be interpreted as 45%, signifying the probability of an on-target shot resulting in a goal. This is determined by various factors, including the shot's position, speed, and placement. It's worth noting that the maximum figure in the plot nears 0.5. To gain a comprehensive understanding, it's imperative to also consider xGOT from open play divided by goals scored. This comparison will help us account for players who, despite taking an excessive number of shots compared to others, may end up with a lower shot quality value.

In analyzing this graph, it's noticeable that players such as Rashford, Harry Kane, Lewandowski, and Messi appear in the bottom half. Multiple reasons may contribute to this placement, including the possibility that they took a higher number of shots on target, thereby resulting in a lower shot quality value. To fully appreciate these values, it's essential to conduct an in-depth analysis considering various aspects like shots on target, xGOT from open play, and goals scored. Every player offers a unique narrative concerning their respective shot quality values. Here are some illustrative examples:

Vedat Muriqi: Positioned 22nd for xGOT from open play among 33 players and having the 5th lowest shots on target tally, Muriqi still achieved an impressive Shot Quality of 0.496.

Erling Haaland: Ranking 3rd highest for xGOT from open play and with the 5th highest shot count, Haaland demonstrated precision and deadliness in front of the goal. Despite these high values, his Shot Quality of 0.418 places him in the middle of the plot.

Kylian Mbappe (not in the graph): An intriguing case, Mbappe's 6th worst shot quality doesn't diminish his prowess as a ball striker. With the highest number of shots on target last season (17 more than the next player), his shot quality value appears deflated. Other factors, such as defenders' pressure and the solidity of the opposing defense, may also influence xGOT from open play. These complexities remind us that a higher shot quality value does not definitively signify a better striker than one with a lower value. A solution to this dilemma might be multi-season data, but currently, seasonal data remains quite limited.

In conclusion, interpreting shot quality requires a nuanced understanding, considering the multifaceted nature of the game and the unique characteristics of each player.

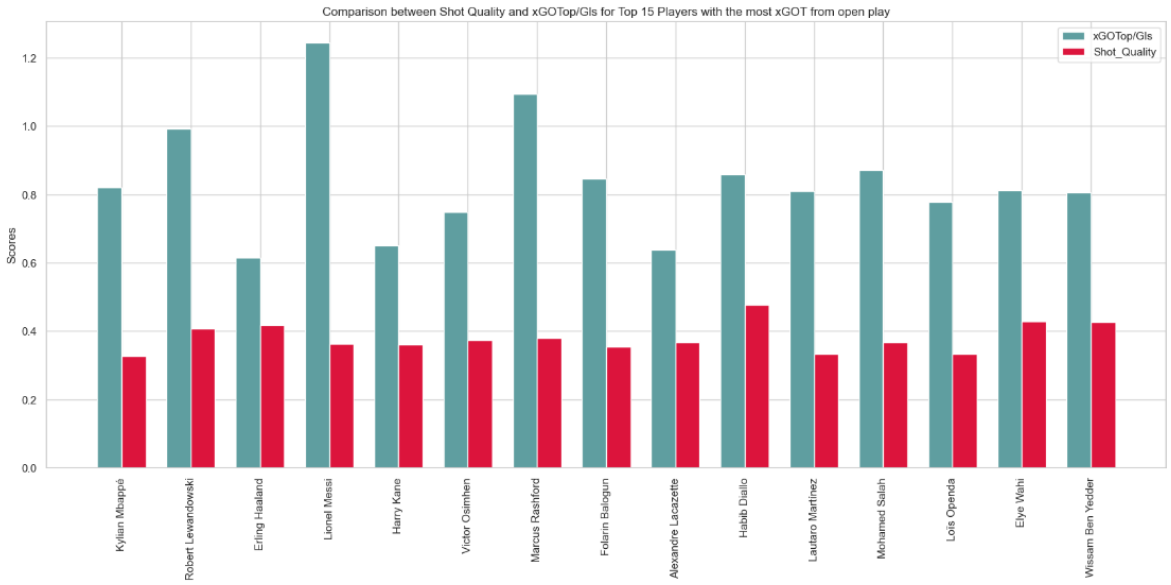
In [33]:
```
xgot_unfiltered_df['xGOTop/Gls'] = xgot_unfiltered_df['xGOTop'] / xgot_unfiltered_df['Gls']
xgot_df = xgot_unfiltered_df[xgot_unfiltered_df['Gls'] >= 15]
xgot_df
```

Out[33]:

	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	90s	Gls	...	xG	npxG	npxG/Sh	G-xG	np:G-xG	xGOT	xGOTop	xGOT_difference	S
0	1	Erling Haaland	no NOR	FW	Manchester City	eng Premier League	22	2000	30.8	36	...	28.4	23.1	0.20	7.6	5.9	29.0	22.175	6.825	
1	8	Folarin Balogun	us USA	FW	Reims	fr Ligue 1	21	2001	33.3	21	...	26.6	21.0	0.18	-5.6	-6.0	23.6	17.750	5.850	
2	3	Kylian Mbappé	fr FRA	FW	Paris S-G	fr Ligue 1	23	1998	31.3	29	...	26.3	22.2	0.15	2.7	3.8	26.7	23.775	2.925	
3	7	Robert Lewandowski	pl POL	FW	Barcelona	es La Liga	33	1988	31.6	23	...	24.3	23.5	0.18	-1.3	-0.5	22.8	22.800	0.000	
4	4	Alexandre Lacazette	fr FRA	FW	Lyon	fr Ligue 1	31	1991	32.5	27	...	24.1	17.8	0.17	2.9	3.2	23.1	17.250	5.850	
5	6	Jonathan David	ca CAN	FW	Lille	fr Ligue 1	22	2000	35.2	24	...	23.6	15.1	0.16	0.4	-1.1	24.7	14.950	9.750	
6	15	Mohamed Salah	eg EGY	FW	Liverpool	eng Premier League	30	1992	36.6	19	...	21.7	18.5	0.15	-2.7	-1.5	18.5	16.550	1.950	
7	2	Harry Kane	eng ENG	FW	Tottenham	eng Premier League	29	1993	37.8	30	...	21.5	16.7	0.13	8.5	8.3	24.4	19.525	4.875	
8	14	Karim Benzema	fr FRA	FW	Real Madrid	es La Liga	34	1987	22.6	19	...	21.5	14.9	0.15	-2.5	-2.9	18.4	11.575	6.825	
9	5	Victor Osimhen	ng NGA	FW	Napoli	it Serie A	23	1998	28.5	26	...	21.3	18.9	0.14	4.7	5.1	21.4	19.450	1.950	
10	12	Ivan Toney	eng ENG	FW	Brentford	eng Premier League	26	1996	32.8	20	...	18.7	13.2	0.15	1.3	0.8	19.6	13.750	5.850	
11	10	Loïs Openda	be BEL	FW	Lens	fr Ligue 1	22	2000	28.0	21	...	18.5	17.7	0.17	2.5	2.3	17.3	16.325	0.975	
12	9	Lautaro Martínez	ar ARG	FW	Inter	it Serie A	24	1997	28.6	21	...	18.1	16.5	0.14	2.9	3.5	18.0	17.025	0.975	

While the ratio of xGOT from open play to goals doesn't provide a straightforward insight, the division of xGOT from open play by shots offers a glimpse into shot quality based on placement, power, and precision. As xGOT aggregates from all shots on target, when calculating shot quality (using the formula xGOT from open play divided by shots on target), we obtain a percentage representing the likelihood of a goal stemming from shot quality. A high value in this metric can point to two scenarios: the player consistently delivers high-quality shots or the player shoots efficiently, capitalizing on fewer attempts. On the other hand, using xGOT divided by goals mainly serves a purpose when evaluating players who take an unusually large number of shots on target, like Mbappe, facilitating comparisons between the two metrics.

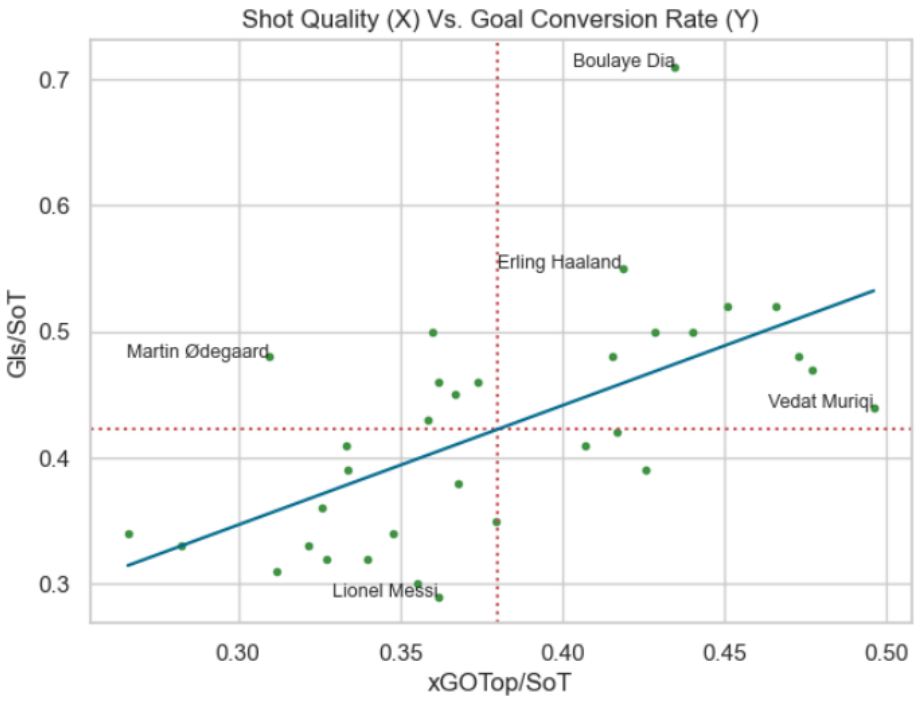
In [34]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Assuming xgot_df is your DataFrame containing player statistics
# Get the top 15 players with the highest xGOTop values
top_players = xgot_df.nlargest(15, 'xGOTop')

# Extract player names for labeling the x-axis
labels = top_players['Player']

# Create an array of x-values for the bar positions
x = np.arange(len(labels))

# Set the width of the bars
width = 0.35

# Create a new figure and axis for the bar plot
fig, ax = plt.subplots()

# Create the first set of bars for xGOTop/Gls values, with color '#5F9EA0'
rects1 = ax.bar(x - width/2, top_players['xGOTop/Gls'], width, label='xGOTop/Gls', color='#5F9EA0')

# Create the second set of bars for Shot Quality values, with color '#DC143C'
rects2 = ax.bar(x + width/2, top_players['Shot_Quality'], width, label='Shot_Quality', color='#DC143C')

# Set labels and title for the plot
ax.set_ylabel('Scores')
ax.set_title('Comparison between Shot Quality and xGOTop/Gls for Top 15 Players with the most xGOT from open play')

# Set the tick positions and labels on the x-axis
ax.set_xticks(x)
ax.set_xticklabels(labels, rotation=90)

# Add a legend to the plot
ax.legend()

plt.show()
```



Immediately, we observe that Messi's xGOTop/Gls ratio exceeds 1.0, distinguishing him from other players. This can be attributed to him netting 16 goals from an xGOT of 19.9 from open play. This discrepancy could be attributed to Messi either encountering more formidable goalkeepers or experiencing a stroke of misfortune in his attempts.

To delve deeper into this observation, we'll examine a scatterplot juxtaposing goal conversion rate (goals/shots on target) with shot quality. This will help us discern which players were fortunate or less so, or perhaps came up against more challenging goalkeepers.

In [35]:
```
# Fit a linear regression line to the 'xG' and 'xGOT' data
slope, intercept = np.polyfit(xgot_df['Shot_Quality'], xgot_df['G/SoT'], 1)

# Generate the x values for the trendline
x_values = np.linspace(xgot_df['Shot_Quality'].min(), xgot_df['Shot_Quality'].max(), 100)

# Calculate the corresponding y values using the slope and intercept
y_values = slope * x_values + intercept

# Now you can create your scatterplot with this new DataFrame
fig, ax = plt.subplots()
fig.set_size_inches(7, 5)

# Add a horizontal line at the mean of 'G/SoT' using a red dashed line
plt.plot(xgot_df['Shot_Quality'], xgot_df['G/SoT'], ".", color = '#3D9140')
plt.axvline(xgot_df['Shot_Quality'].mean(), linestyle=':', color='r')  # Vertical line at the mean of 'Gls'
plt.axhline(y=xgot_df['G/SoT'].mean(), linestyle=':', color='r')  # Horizontal line at the mean of 'MP'

# Plot the linear trendline using blue color
plt.plot(x_values, y_values, "-", color="#00688B")  # Plot the trendline

# Set title and labels for axes
ax.set_title("Shot Quality (X) Vs. Goal Conversion Rate (Y)")
ax.set_xlabel("xGOTop/SoT")
ax.set_ylabel("Gls/SoT")

# List of highlighted player names for annotation
highlighted_players = ["Boulaye Dia", "Martin Ødegaard", "Lionel Messi", "Erling Haaland", "Vedat Muriqi"]  # Replace with actual

# Add annotations for highlighted players
for index, row in xgot_df.iterrows():
    if row['Player'] in highlighted_players:
        plt.text(row['Shot_Quality'], row['G/SoT'], row['Player'], fontsize=9, ha='right')

plt.show()
```



From the graph, four players distinctly stand out: Dia, Haaland, Odegaard, Messi, and Muriqi, each occupying unique positions.

Dia boasts the highest goal conversion rate among all the players, indicating that nearly 75% of his shots turned into goals. While his shot quality was commendable, it wasn't the highest. This suggests he might have enjoyed a touch of luck, especially when compared to players like Muriqi who had higher xGOT per shot on target.

Ødegaard, not typically renowned for his shooting prowess, had a goal conversion rate comparable to Muriqi. However, his preference for passing over shooting is evident in his notably lower shot quality.

Then there's Messi, the World Cup champion. His shot quality and conversion rate were both slightly below the average, which might be attributed to him attempting a lot of shots on target (he had the third highest in the top 5 leagues last season). To determine if a player is genuinely fortuitous or unfortunate in front of the goal, one method is to subtract xG (from open play shots on target only) from xGOT (open). This gives insight into the threat level each shot on target posed. By comparing this to the number of goals scored, we can discern a player's luck factor. Unfortunately, given that xG encompasses all shots, including those off target, and without having the precise data, it's not entirely accurate, however it’s close.

An alternative is to use xGOT from open play minus xG (excluding penalties). Although it's a deviation, this is akin to a pre-existing metric known as Shooting Goals Added (SGA). To adjust for the potential bias introduced by penalties, we will subtract the product of the numbers of penalties taken and 0.76 (the constant xG value associated with penalty shots). This adjustment is essential in order to nullify the xG of shots taken from penalty spots since they spike the data in favour of the usual pen takers.

In [36]:
```
xgot_unfiltered_df['xGop'] = xgot_unfiltered_df['xG'] - xgot_unfiltered_df['PK'] * 0.76
xgot_unfiltered_df['adjSGA'] = xgot_unfiltered_df['xGOTop'] - xgot_unfiltered_df['xGop']


xgot_df = xgot_unfiltered_df[xgot_unfiltered_df['Gls'] >= 15]
print(xgot_df.columns)
xgot_df
```

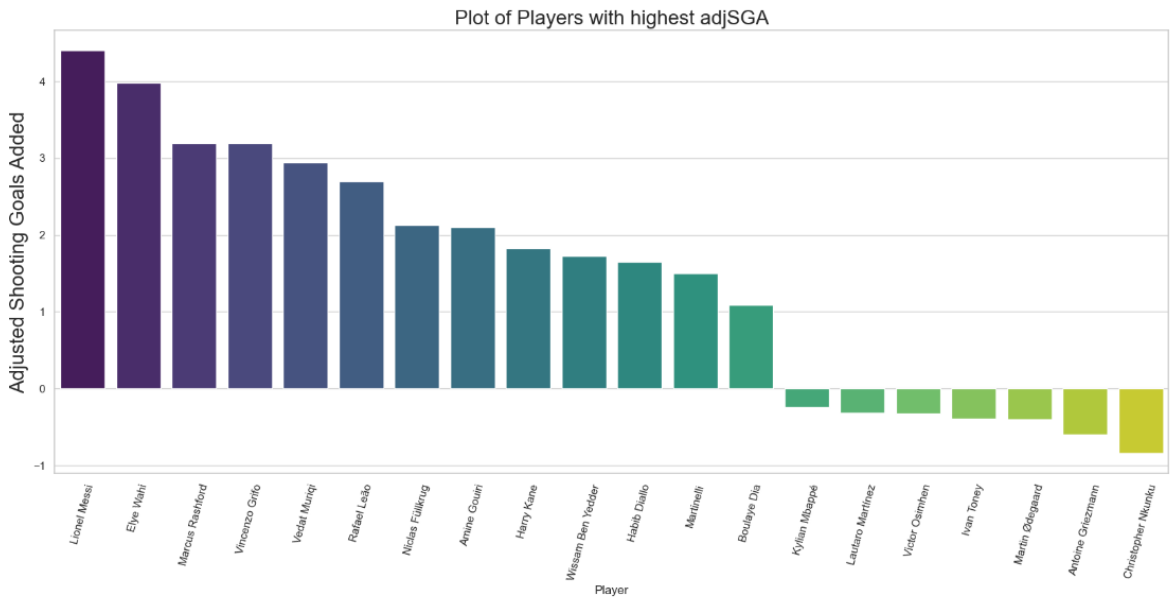
```
Index(['Rk', 'Player', 'Nation', 'Pos', 'Squad', 'Comp', 'Age', 'Born', '90s',
       'Gls', 'Sh', 'SoT', 'SoT%', 'Sh/90', 'SoT/90', 'G/Sh', 'G/SoT', 'Dist',
       'FK', 'PK', 'PKatt', 'xG', 'npxG', 'npxG/Sh', 'G-xG', 'np:G-xG', 'xGOT',
       'xGOTop', 'xGOT_difference', 'Shot_Quality', 'xGOTop/Gls', 'xGop',
       'adjSGA'],
      dtype='object')
```

Out[36]:

on	Pos	Squad	Comp	Age	Born	90s	Gls	...	npxG/Sh	G-xG	np:G-xG	xGOT	xGOTop	xGOT_difference	Shot_Quality	xGOTop/Gls	xGop	adjSGA
no DR	FW	Manchester City	eng Premier League	22	2000	30.8	36	...	0.20	7.6	5.9	29.0	22.175	6.825	0.418396	0.615972	23.08	-0.905
us SA	FW	Reims	fr Ligue 1	21	2001	33.3	21	...	0.18	-5.6	-6.0	23.6	17.750	5.850	0.355000	0.845238	22.04	-4.290
RA	FW	Paris S-G	fr Ligue 1	23	1998	31.3	29	...	0.15	2.7	3.8	26.7	23.775	2.925	0.325685	0.819828	24.02	-0.245
OL	FW	Barcelona	es La Liga	33	1988	31.6	23	...	0.18	-1.3	-0.5	22.8	22.800	0.000	0.407143	0.991304	24.30	-1.500
RA	FW	Lyon	fr Ligue 1	31	1991	32.5	27	...	0.17	2.9	3.2	23.1	17.250	5.850	0.367021	0.638889	19.54	-2.290
ca AN	FW	Lille	fr Ligue 1	22	2000	35.2	24	...	0.16	0.4	-1.1	24.7	14.950	9.750	0.339773	0.622917	16.00	-1.050
eg GY	FW	Liverpool	eng Premier League	30	1992	36.6	19	...	0.15	-2.7	-1.5	18.5	16.550	1.950	0.367778	0.871053	20.18	-3.630
ng VG	FW	Tottenham	eng Premier League	29	1993	37.8	30	...	0.13	8.5	8.3	24.4	19.525	4.875	0.361574	0.650833	17.70	1.825
RA	FW	Real Madrid	es La Liga	34	1987	22.6	19	...	0.15	-2.5	-2.9	18.4	11.575	6.825	0.321528	0.609211	16.18	-4.605

We've now introduced two additional columns: xG from open play (excluding penalties) and, based on that, the adjusted SGA (Shooting Goals Added) and now let’s delve deeper.

In [37]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Get the top 20 players with the highest 'adjSGA' values
top_adjSGA_players = xgot_df.nlargest(20, 'adjSGA')

# Sort the top_adjSGA_players DataFrame by 'adjSGA' column in descending order
top_adjSGA_players_sorted = top_adjSGA_players.sort_values('adjSGA', ascending=False)

# Set the theme and style for the plot using seaborn
sns.set_theme(style="whitegrid", color_codes=True)

# Create a bar plot using seaborn
# x-axis: Player names, y-axis: Adjusted Shooting Goals Added (adjSGA)
# Use 'viridis' palette for color variation
ax = sns.barplot(x='Player', y='adjSGA', data=top_adjSGA_players_sorted, palette='viridis')

# Set y-axis label
ax.set_ylabel("Adjusted Shooting Goals Added", fontsize=20)

# Rotate x-axis labels for better readability
plt.xticks(rotation=75)

# Set the figure size
plt.rcParams["figure.figsize"] = (20, 8)

# Set the title for the plot
plt.title('Plot of Players with highest adjSGA', fontsize=20)

plt.show()
```



Finally, we see Messi leading the pack. It's intriguing to observe that some players display a negative adjusted SGA. This could indicate that they garnered a lower xGOT compared to their xG. But there's a caveat: this analysis might not be entirely precise given that xG factors in all shots, whereas xGOT only considers those on target. Nonetheless, even with access to xG data exclusively for shots on target, certain players would inevitably register a negative value, perhaps due to subpar shooting skills or facing formidable defenses that applied pressure, restricting the quality of their shots.

Surprisingly, even after achieving the highest adjusted SGA, Messi didn't make it to the top 20 goal-scorers last season. This lends credence to our earlier assumption that he might have been up against superior goalkeepers or had a bout of bad luck in front of the net. In contrast, his teammate Mbappe, despite being the third-highest goal-scorer last season, recorded a negative adjusted SGA. This could hint at him being somewhat fortunate in his scoring endeavors. However, it's essential to note that this analysis doesn't paint the full picture. Mbappe's explosive style and knack for bypassing defenses often put him in prime positions close to the goal, naturally elevating his xG.

Moving forward, I plan to enrich this analysis by incorporating multiple datasets, allowing for a more comprehensive understanding of metrics like Shot Quality. Stay tuned for further projects where I'll be sharing my findings and exploring deeper into the intricate layers of football statistics.