I would like to explore how a fighter like Charles "Do Bronx" Oliveira, comes out on top in the UFC Lightweight Division, one of the most competative divisions in the promotion. This analysis comes directly after his dominant win over Dustin Poirier for the Lightweight title. 12/24/21

In [1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
```

In [2]:
```
pd.set_option('display.max_columns', None)
```

In the 10 fights prior to his current 10 fight win streak that includes fights with budding ufc hopefuls and future UFC Champtions like Max Holloway and Paul Felder, as well as the beginning of his switch to lightweight in which over time has shown to be a decision that has been able to return great dividends to his career. During that period its clear that Charles still did not posses the concrete skills nor the intangibles to compete with the elite of either the featherweight or Lightweight division. Here I would like to explore the difference between the different versions of Charles from 2014-2017, the Past Oliveira and 2018-2021 the Champ Oliveira.

Analysis of two different Charles

2014 - 2017 Charles Olivera (6-4)

In [3]:
```
df1 = pd.read_csv('charles_before_champ_run.csv', error_bad_lines=False)

df1.dropna()

df1['seconds'] = df1.control_time.str.split(':',expand=True)[1]
df1['minutes'] = df1.control_time.str.split(':',expand=True)[0]

df1['seconds'] = df1['seconds'].astype(int)
df1['minutes'] = df1['minutes'].astype(int)

df1['control_time'] = (df1['minutes'] * 60) + df1['seconds']
df1 = df1.drop(columns=['seconds','minutes'])


df1['seconds'] = df1.ending_time.str.split(':',expand=True)[1]
df1['minutes'] = df1.ending_time.str.split(':',expand=True)[0]

df1['seconds'] = df1['seconds'].astype(int)
df1['minutes'] = df1['minutes'].astype(int)

df1['ending_time'] = (df1['minutes'] * 60) + df1['seconds']
df1 = df1.drop(columns=['seconds','minutes'])

df1['standing_time'] = df1.ending_time - df1.control_time

df1.loc[df1.round_number != df1.ending_round,'ending_time'] = 300
print(df1.columns)
```

```
Index(['fight_id', 'fight_title', 'winning_fighter_name',
       'losing_fighter_name', 'weight_class', 'method_of_victory',
       'specific_victory_details', 'ending_round', 'ending_time', 'fight_url',
       'unique_round_id', 'unique_fight_id', 'unique_event_id', 'round_number',
       'fighter_name', 'knockdowns', 'total_strikes_landed',
       'total_strikes_attempted', 'takedowns', 'takedowns_attempted',
       'submission_attempts', 'guard_passes', 'reversals', 'control_time',
       'unique_round_id.1', 'unique_fight_id.1', 'unique_event_id.1',
       'round_number.1', 'fighter_name.1', 'significant_strikes_head_landed',
       'significant_strikes_head_attempted', 'significant_strikes_body_landed',
       'significant_strikes_body_attempted', 'significant_strikes_leg_landed',
       'significant_strikes_leg_attempted',
       'significant_strikes_standing_landed',
       'significant_strikes_standing_attempted',
       'significant_strikes_clinch_landed',
       'significant_strikes_clinch_attempted',
       'significant_strikes_ground_landed',
       'significant_strikes_ground_attempted', 'event_id', 'event_title',
       'event_date', 'event_location', 'event_attendance', 'event_url',
       'standing_time'],
      dtype='object')
```

In [4]:
```
df1.head(5)
#print(df1.tail(5))
```

Out[4]:

	fight_id	fight_title	winning_fighter_name	losing_fighter_name	weight_class	method_of_victory	specific_victory_details	ending_round	end
0	1864	UFC on FOX: Dos Anjos vs. Cowboy 2	Charles Oliveira	Myles Jury	Featherweight	SUB	Guillotine Choke	1	
1	2074	UFC Fight Night: Holloway vs Oliveira	Max Holloway	Charles Oliveira	Featherweight	KO/TKO	NaN	1	
2	2590	UFC Fight Night: Te Huna vs Marquardt	Charles Oliveira	Hatsu Hioki	Featherweight	SUB	Anaconda Choke	2	
3	2590	UFC Fight Night: Te Huna vs Marquardt	Charles Oliveira	Hatsu Hioki	Featherweight	SUB	Anaconda Choke	2	
4	2310	UFC 210: Cormier vs. Johnson 2	Charles Oliveira	Will Brooks	Lightweight	SUB	Rear Naked Choke	1	

In [5]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 48 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   fight_id                                21 non-null     int64
 1   fight_title                             21 non-null     object
 2   winning_fighter_name                    21 non-null     object
 3   losing_fighter_name                     21 non-null     object
 4   weight_class                            21 non-null     object
 5   method_of_victory                       21 non-null     object
 6   specific_victory_details                17 non-null     object
 7   ending_round                            21 non-null     int64
 8   ending_time                             21 non-null     int32
 9   fight_url                               21 non-null     object
 10  unique_round_id                         21 non-null     int64
 11  unique_fight_id                         21 non-null     int64
 12  unique_event_id                         21 non-null     int64
 13  round_number                            21 non-null     int64
 14  fighter_name                            21 non-null     object
 15  knockdowns                              21 non-null     int64
 16  total_strikes_landed                    21 non-null     int64
 17  total_strikes_attempted                 21 non-null     int64
 18  takedowns                               21 non-null     int64
 19  takedowns_attempted                     21 non-null     int64
 20  submission_attempts                     21 non-null     int64
 21  guard_passes                            21 non-null     int64
 22  reversals                               21 non-null     int64
 23  control_time                            21 non-null     int32
 24  unique_round_id.1                       21 non-null     int64
 25  unique_fight_id.1                       21 non-null     int64
 26  unique_event_id.1                       21 non-null     int64
 27  round_number.1                          21 non-null     int64
 28  fighter_name.1                          21 non-null     object
 29  significant_strikes_head_landed         21 non-null     int64
 30  significant_strikes_head_attempted      21 non-null     int64
 31  significant_strikes_body_landed         21 non-null     int64
 32  significant_strikes_body_attempted      21 non-null     int64
 33  significant_strikes_leg_landed          21 non-null     int64
 34  significant_strikes_leg_attempted       21 non-null     int64
 35  significant_strikes_standing_landed     21 non-null     int64
 36  significant_strikes_standing_attempted  21 non-null     int64
 37  significant_strikes_clinch_landed       21 non-null     int64
 38  significant_strikes_clinch_attempted    21 non-null     int64
 39  significant_strikes_ground_landed       21 non-null     int64
 40  significant_strikes_ground_attempted    21 non-null     int64
 41  event_id                                21 non-null     int64
 42  event_title                             21 non-null     object
 43  event_date                              21 non-null     object
 44  event_location                          21 non-null     object
 45  event_attendance                        21 non-null     object
 46  event_url                               21 non-null     object
 47  standing_time                           21 non-null     int32
dtypes: int32(3), int64(31), object(14)
memory usage: 7.8+ KB
```

In [6]:
```
df1.isnull().sum()
```

Out[6]:
```
fight_id                                  0
fight_title                               0
winning_fighter_name                      0
losing_fighter_name                       0
weight_class                              0
method_of_victory                         0
specific_victory_details                  4
ending_round                              0
ending_time                               0
fight_url                                 0
unique_round_id                           0
unique_fight_id                           0
unique_event_id                           0
round_number                              0
fighter_name                              0
knockdowns                                0
total_strikes_landed                      0
total_strikes_attempted                   0
takedowns                                 0
takedowns_attempted                       0
submission_attempts                       0
guard_passes                              0
reversals                                 0
control_time                              0
unique_round_id.1                         0
unique_fight_id.1                         0
unique_event_id.1                         0
round_number.1                            0
fighter_name.1                            0
significant_strikes_head_landed           0
significant_strikes_head_attempted        0
significant_strikes_body_landed           0
significant_strikes_body_attempted        0
significant_strikes_leg_landed            0
significant_strikes_leg_attempted         0
significant_strikes_standing_landed       0
significant_strikes_standing_attempted    0
significant_strikes_clinch_landed         0
significant_strikes_clinch_attempted      0
significant_strikes_ground_landed         0
significant_strikes_ground_attempted      0
event_id                                  0
event_title                               0
event_date                                0
event_location                            0
event_attendance                          0
event_url                                 0
standing_time                             0
dtype: int64
```

In [7]:
```
ko_percent = df1[['winning_fighter_name', 'losing_fighter_name', 'method_of_victory', 'specific_victory_details

#fill in Empty specific victory details
ko_percent.loc[1].at['specific_victory_details'] = 'At Distance Shoulder Injury'
ko_percent.loc[12].at['specific_victory_details'] = 'Decision'
ko_percent.loc[13].at['specific_victory_details'] = 'Decision'
ko_percent.loc[14].at['specific_victory_details'] = 'Decision'


ko_percent.head(21)
```

Out[7]:

	winning_fighter_name	losing_fighter_name	method_of_victory	specific_victory_details	event_date
0	Charles Oliveira	Myles Jury	SUB	Guillotine Choke	2015-12-19
1	Max Holloway	Charles Oliveira	KO/TKO	At Distance Shoulder Injury	2015-08-23
2	Charles Oliveira	Hatsu Hioki	SUB	Anaconda Choke	2014-06-28
3	Charles Oliveira	Hatsu Hioki	SUB	Anaconda Choke	2014-06-28
4	Charles Oliveira	Will Brooks	SUB	Rear Naked Choke	2017-04-08
5	Anthony Pettis	Charles Oliveira	SUB	Guillotine Choke	2016-08-27
6	Anthony Pettis	Charles Oliveira	SUB	Guillotine Choke	2016-08-27
7	Anthony Pettis	Charles Oliveira	SUB	Guillotine Choke	2016-08-27
8	Ricardo Lamas	Charles Oliveira	SUB	Guillotine Choke	2016-11-05
9	Ricardo Lamas	Charles Oliveira	SUB	Guillotine Choke	2016-11-05
10	Paul Felder	Charles Oliveira	KO/TKO	Elbows	2017-12-02
11	Paul Felder	Charles Oliveira	KO/TKO	Elbows	2017-12-02
12	Charles Oliveira	Jeremy Stephens	U-DEC	Decision	2014-12-12
13	Charles Oliveira	Jeremy Stephens	U-DEC	Decision	2014-12-12
14	Charles Oliveira	Jeremy Stephens	U-DEC	Decision	2014-12-12
15	Charles Oliveira	Nik Lentz	SUB	Guillotine Choke	2015-05-30
16	Charles Oliveira	Nik Lentz	SUB	Guillotine Choke	2015-05-30
17	Charles Oliveira	Nik Lentz	SUB	Guillotine Choke	2015-05-30
18	Charles Oliveira	Andy Ogle	SUB	Triangle Choke	2014-02-15
19	Charles Oliveira	Andy Ogle	SUB	Triangle Choke	2014-02-15
20	Charles Oliveira	Andy Ogle	SUB	Triangle Choke	2014-02-15

Dropping the duplicates and manually adding fight outcome ratios

In [8]:
```
ko_percent = ko_percent.drop([3, 6, 7, 9, 11, 13, 14, 16, 17, 19, 20])
ko_percent.sort_values(by=['event_date'], inplace=True)

ko_percent['finished_percentage'] = '0%'
ko_percent['finish_percentage'] = '100%'
ko_percent['finished_ratio'] = '0:0'
ko_percent['finish_ratio'] = '0:0'



ko_percent['finished_ratio'].iloc[[ko_percent.losing_fighter_name == 'Andy Ogle']] = '0:1'
ko_percent['finished_ratio'].iloc[[ko_percent.losing_fighter_name == 'Hatsu Hioki']] = '0:2'
ko_percent['finished_ratio'].iloc[[ko_percent.losing_fighter_name == 'Jeremy Stephens']] = '0:3'
ko_percent['finished_ratio'].iloc[[ko_percent.losing_fighter_name == 'Nik Lentz']] = '0:4'
ko_percent['finished_ratio'].iloc[[ko_percent.winning_fighter_name == 'Max Holloway']] = '1:5'
ko_percent['finished_ratio'].iloc[[ko_percent.losing_fighter_name == 'Myles Jury']] = '1:6'
ko_percent['finished_ratio'].iloc[[ko_percent.winning_fighter_name == 'Anthony Pettis']] = '2:7'
ko_percent['finished_ratio'].iloc[[ko_percent.winning_fighter_name == 'Ricardo Lamas']] = '3:8'
ko_percent['finished_ratio'].iloc[[ko_percent.losing_fighter_name == 'Will Brooks']] = '3:9'
ko_percent['finished_ratio'].iloc[[ko_percent.winning_fighter_name == 'Paul Felder']] = '4:10'


ko_percent['finish_ratio'].iloc[[ko_percent.losing_fighter_name == 'Andy Ogle']] = '1:1'
ko_percent['finish_ratio'].iloc[[ko_percent.losing_fighter_name == 'Hatsu Hioki']] = '2:2'
ko_percent['finish_ratio'].iloc[[ko_percent.losing_fighter_name == 'Jeremy Stephens']] = '2:3'
ko_percent['finish_ratio'].iloc[[ko_percent.losing_fighter_name == 'Nik Lentz']] = '3:4'
ko_percent['finish_ratio'].iloc[[ko_percent.winning_fighter_name == 'Max Holloway']] = '3:5'
ko_percent['finish_ratio'].iloc[[ko_percent.losing_fighter_name == 'Myles Jury']] = '4:6'
ko_percent['finish_ratio'].iloc[[ko_percent.winning_fighter_name == 'Anthony Pettis']] = '4:7'
ko_percent['finish_ratio'].iloc[[ko_percent.winning_fighter_name == 'Ricardo Lamas']] = '4:8'
ko_percent['finish_ratio'].iloc[[ko_percent.losing_fighter_name == 'Will Brooks']] = '5:9'
ko_percent['finish_ratio'].iloc[[ko_percent.winning_fighter_name == 'Paul Felder']] = '5:10'

#ko_percent.head(11)
percentages = ko_percent[['finished_percentage', 'finish_percentage', 'finished_ratio', 'finish_ratio', 'event_
percentages.head()
```

```
c:\users\antho\appdata\local\programs\python\python39\lib\site-packages\pandas\core\internals\blocks.py:1025: F
utureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` ins
tead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will
result either in an error or a different result.
  values[indexer] = value
```

Out[8]:

	finished_percentage	finish_percentage	finished_ratio	finish_ratio	event_date	winning_fighter_name	losing_fighter_name
18	0%	100%	0:1	1:1	2014-02-15	Charles Oliveira	Andy Ogle
2	0%	100%	0:2	2:2	2014-06-28	Charles Oliveira	Hatsu Hioki
12	0%	100%	0:3	2:3	2014-12-12	Charles Oliveira	Jeremy Stephens
15	0%	100%	0:4	3:4	2015-05-30	Charles Oliveira	Nik Lentz
1	0%	100%	1:5	3:5	2015-08-23	Max Holloway	Charles Oliveira

In [9]:
```
percentages = percentages.reset_index(drop=True)
#percentages = percentages.drop(columns='index')

finish_ratios = percentages.finish_ratio.array

finished_ratios = percentages.finished_ratio.array

finished = []
percent = []

for ratio in finished_ratios:
    x, y = ratio.split(':')
    p = (int(x) / int(y))
    per = '{:.0%}'.format(p)
    finished.append(str(per))

for ratio in finish_ratios:
    x, y = ratio.split(':')

    p = (int(x) / int(y))
    per = '{:.0%}'.format(p)
    percent.append(str(per))
    #print(f'x: {x}')
    #print(f'y: {y}')

print(percent)
print(finished)
#print(percentages.iloc[0, 'finish_ratio'])
percentages['finish_percentage'] = percent
percentages['finished_percentage'] = finished
percentages.head(101)
```

```
['100%', '100%', '67%', '75%', '60%', '67%', '57%', '50%', '56%', '50%']
['0%', '0%', '0%', '0%', '20%', '17%', '29%', '38%', '33%', '40%']
```

Out[9]:

	finished_percentage	finish_percentage	finished_ratio	finish_ratio	event_date	winning_fighter_name	losing_fighter_name
0	0%	100%	0:1	1:1	2014-02-15	Charles Oliveira	Andy Ogle
1	0%	100%	0:2	2:2	2014-06-28	Charles Oliveira	Hatsu Hioki
2	0%	67%	0:3	2:3	2014-12-12	Charles Oliveira	Jeremy Stephens
3	0%	75%	0:4	3:4	2015-05-30	Charles Oliveira	Nik Lentz
4	20%	60%	1:5	3:5	2015-08-23	Max Holloway	Charles Oliveira
5	17%	67%	1:6	4:6	2015-12-19	Charles Oliveira	Myles Jury
6	29%	57%	2:7	4:7	2016-08-27	Anthony Pettis	Charles Oliveira
7	38%	50%	3:8	4:8	2016-11-05	Ricardo Lamas	Charles Oliveira
8	33%	56%	3:9	5:9	2017-04-08	Charles Oliveira	Will Brooks
9	40%	50%	4:10	5:10	2017-12-02	Paul Felder	Charles Oliveira

Its clear to see that during this run Charles does not find consistency in his performances. He goes from finishing most of his fights to being finished multiple times and even trying a heavier division where he is also finished

In [10]:
```
# creating the group frame

group_frame = df1[['fighter_name','significant_strikes_standing_landed','significant_strikes_head_attempted','s
group_frame.head()
```

Out[10]:

	fighter_name	significant_strikes_standing_landed	significant_strikes_head_attempted	significant_strikes_body_attempted	significant_strikes_leg_a
0	Charles Oliveira	90	250	104	

In [11]:
```
group_frame.fighter_name = 'Past Oliveira'
```

In [12]:
```
group_frame['head_ratio'] = group_frame.significant_strikes_head_attempted / (group_frame.significant_strikes_h
group_frame['body_ratio'] = group_frame.significant_strikes_body_attempted / (group_frame.significant_strikes_h
group_frame['leg_ratio'] = group_frame.significant_strikes_leg_attempted / (group_frame.significant_strikes_hea

group_frame.head()
```

Out[12]:

	fighter_name	significant_strikes_standing_landed	significant_strikes_head_attempted	significant_strikes_body_attempted	significant_strikes_leg_a
0	Past Oliveira	90	250	104	

Lets look at the ratios of Head Strikes, Body Strikes and Leg strikes for Oliveira in his past

In [13]:
```
ratio_frame = group_frame[['fighter_name', 'head_ratio', 'body_ratio', 'leg_ratio' ]]

ratio_frame.head()
```

Out[13]:

	fighter_name	head_ratio	body_ratio	leg_ratio
0	Past Oliveira	0.652742	0.27154	0.075718

With these numbers it can be seen that in this period of Charles' Career that he avoids head hunting. Basically this past Oliveira strike attempts to the head 65%, to the body 27% and to the leg 8% while standing. These numbers also show that this past Charles Strikes to the body significantly more than other Lightweights he currently competes with. This result could likely be evidence of the different fighting dynamics that appear across the weightclasses amongst elite level competition. To compare this example I would note that Charles Started his career in the UFC as a Featherweight, 145lb weight class where volume, speed and power are high advantages for successful eye catching performances. Charles' persistance to the body is evident of his plan to tire out his faster and stronger opponents. Charles in this weight class was cutting a significant amount of weight despite his young age with a frame of 5'10" height and reach of 74". Charles would very quickly outgrow the featherweight division as time went on as well. Now he is currently shining as champion of the lightweight division

In [14]:
```
#STRIKE LOCATION
location_ratio = group_frame[['fighter_name', 'significant_strikes_standing_attempted', 'significant_strikes_gr
```

In [15]:
```
location_ratio['standing_ratio'] = location_ratio.significant_strikes_standing_attempted / (location_ratio.sign
location_ratio['ground_ratio'] = location_ratio.significant_strikes_ground_attempted / (location_ratio.signific
location_ratio['clinch_ratio'] = location_ratio.significant_strikes_clinch_attempted / (location_ratio.signific

print('Charles Oliveira 10 fights before championship run')
print('Strike Location')
print(location_ratio[['fighter_name', 'standing_ratio', 'ground_ratio', 'clinch_ratio']])
```

```
Charles Oliveira 10 fights before championship run
Strike Location
    fighter_name  standing_ratio  ground_ratio  clinch_ratio
0  Past Oliveira        0.527415      0.227154      0.245431
```

These numbers show that during this lackluster stretch of bouts, Oliveiras' Attack is dependent on his grappling and clinch transitions. This is evident in his wins where he finishes by either RNC, Guillotine choke or Anaconda Choke but this point is further solidified in his losses where he loses by submission, specifically guillotine choke.

In [16]:
```
# Turn standing time to minutes
group_frame['standing_time']=group_frame.standing_time/60
```

In [17]:
```
#SApSM
sapsm = group_frame[['fighter_name', 'significant_strikes_standing_attempted','standing_time', 'knockdowns']].g
```

In [18]:
```
#SApSM_Ratio
sapsm['SApSM'] = sapsm.significant_strikes_standing_attempted / sapsm.standing_time
# Sig Strike Attempts per Striking minute
sapsm.head()
```

Out[18]:

	fighter_name	significant_strikes_standing_attempted	standing_time	knockdowns	SApSM
0	Past Oliveira	202	31.7	1	6.37224

In [19]:
```
#SLpSM
slpsm = group_frame[['fighter_name', 'significant_strikes_standing_landed','standing_time', 'knockdowns']].grou
```

In [20]:
```
#SLpSM Ratio
slpsm['SLpSM'] = slpsm.significant_strikes_standing_landed / slpsm.standing_time
# Sig Strike Lands per Striking Minute
slpsm.head()
```

Out[20]:

	fighter_name	significant_strikes_standing_landed	standing_time	knockdowns	SLpSM
0	Past Oliveira	90	31.7	1	2.839117

In [21]:
```
accuracy_frame = group_frame[['significant_strikes_standing_landed', 'significant_strikes_standing_attempted',

accuracy_frame['standing_accuracy'] = accuracy_frame.significant_strikes_standing_landed/accuracy_frame.signifi
# Sig Strike Accuracy percentage
accuracy_frame.head()
```

Out[21]:

	fighter_name	significant_strikes_standing_landed	significant_strikes_standing_attempted	standing_accuracy
0	Past Oliveira	90	202	44.554455

In [22]:
```
# Minutes Standing per Knockdown
kd_frame = group_frame[['standing_time', 'knockdowns', 'fighter_name']].groupby('fighter_name').sum().reset_ind
kd_frame['SMpKD'] = kd_frame.standing_time / kd_frame.knockdowns
kd_frame.head()
```

Out[22]:

	fighter_name	standing_time	knockdowns	SMpKD
0	Past Oliveira	31.7	1	31.7

In [23]:
```
kdstrike_frame = group_frame[['significant_strikes_standing_landed', 'knockdowns', 'fighter_name']].groupby('fi

kdstrike_frame['SLpKD'] = kdstrike_frame.significant_strikes_standing_landed / kdstrike_frame.knockdowns
print(kdstrike_frame)
```

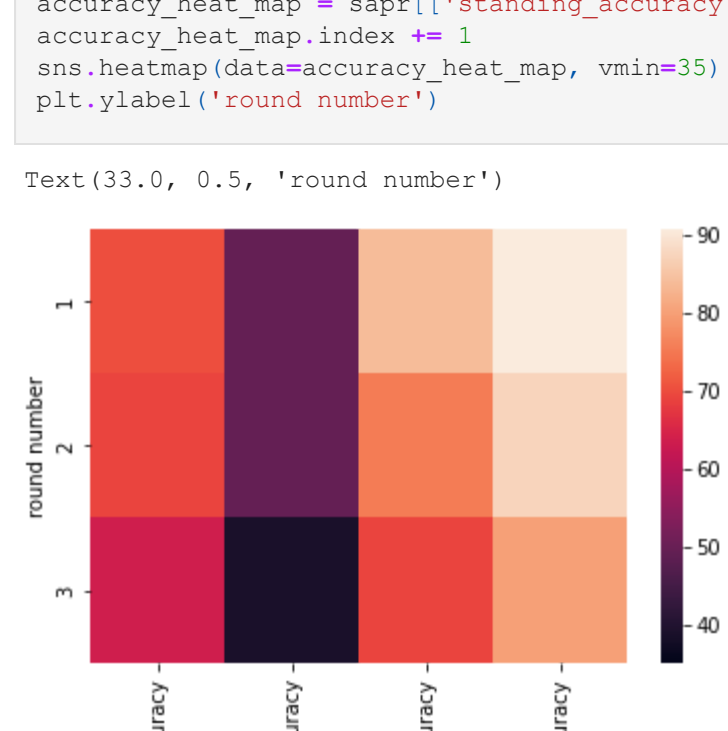
```
    fighter_name  significant_strikes_standing_landed  knockdowns  SLpKD
0  Past Oliveira                                   90           1   90.0
```

In [24]:
```
#striking accuray per round

sapr = df1[['total_strikes_landed', 'total_strikes_attempted', 'fighter_name', 'round_number', 'significant_str
sapr['standing_time']=sapr.standing_time/60
sapr['standing_accuracy'] = (sapr.total_strikes_landed / sapr.total_strikes_attempted) * 100
# graphs for head accuracy, body accuracy and leg accuracy by round

sapr.fighter_name = 'Past Oliveira'

sapr['head_accuracy']=(sapr.significant_strikes_head_landed/sapr.significant_strikes_head_attempted)*100
sapr['body_accuracy']=(sapr.significant_strikes_body_landed/sapr.significant_strikes_body_attempted)*100
sapr['leg_accuracy']=(sapr.significant_strikes_leg_landed/sapr.significant_strikes_leg_attempted)*100


print(sapr[['fighter_name', 'round_number', 'standing_accuracy', 'head_accuracy', 'body_accuracy', 'leg_accurac
```

```
    fighter_name  round_number  standing_accuracy  head_accuracy  \
0  Past Oliveira             1          70.238095      49.572650
1  Past Oliveira             2          69.158879      49.484536
2  Past Oliveira             3          63.636364      38.888889

   body_accuracy  leg_accuracy
0      84.000000     90.909091
1      75.609756     87.500000
2      69.230769     80.000000
```

In [25]:
```
accuracy_heat_map = sapr[['standing_accuracy', 'head_accuracy', 'body_accuracy', 'leg_accuracy']]
accuracy_heat_map.index += 1
sns.heatmap(data=accuracy_heat_map, vmin=35)
plt.ylabel('round number')
```

Out[25]:
```
Text(33.0, 0.5, 'round number')
```



Charles Oliveira 2018-2021 (10-0)

In [26]:
```
df2 = pd.read_csv('charles_champ_run.csv', error_bad_lines=False)

df2.dropna()

df2['seconds'] = df2.control_time.str.split(':',expand=True)[1]
df2['minutes'] = df2.control_time.str.split(':',expand=True)[0]

df2['seconds'] = df2['seconds'].astype(int)
df2['minutes'] = df2['minutes'].astype(int)

df2['control_time'] = (df2['minutes'] * 60) + df2['seconds']
df2 = df2.drop(columns=['seconds','minutes'])


df2['seconds'] = df2.ending_time.str.split(':',expand=True)[1]
df2['minutes'] = df2.ending_time.str.split(':',expand=True)[0]

df2['seconds'] = df2['seconds'].astype(int)
df2['minutes'] = df2['minutes'].astype(int)

df2['ending_time'] = (df2['minutes'] * 60) + df2['seconds']
df2 = df2.drop(columns=['seconds','minutes'])

df2['standing_time'] = df2.ending_time - df2.control_time

df2.loc[df2.round_number != df2.ending_round,'ending_time'] = 300
print(df2.columns)
```

```
Index(['fight_id', 'fight_title', 'winning_fighter_name',
       'losing_fighter_name', 'weight_class', 'method_of_victory',
       'specific_victory_details', 'ending_round', 'ending_time', 'fight_url',
       'unique_round_id', 'unique_fight_id', 'unique_event_id', 'round_number',
       'fighter_name', 'knockdowns', 'total_strikes_landed',
       'total_strikes_attempted', 'takedowns', 'takedowns_attempted',
       'submission_attempts', 'guard_passes', 'reversals', 'control_time',
       'unique_round_id.1', 'unique_fight_id.1', 'unique_event_id.1',
       'round_number.1', 'fighter_name.1', 'significant_strikes_head_landed',
       'significant_strikes_head_attempted', 'significant_strikes_body_landed',
       'significant_strikes_body_attempted', 'significant_strikes_leg_landed',
       'significant_strikes_leg_attempted',
       'significant_strikes_standing_landed',
       'significant_strikes_standing_attempted',
       'significant_strikes_clinch_landed',
       'significant_strikes_clinch_attempted',
       'significant_strikes_ground_landed',
       'significant_strikes_ground_attempted', 'event_id', 'event_title',
       'event_date', 'event_location', 'event_attendance', 'event_url',
       'standing_time'],
      dtype='object')
```

In [27]:
```
df2.head(5)
```

Out[27]:

	fight_id	fight_title	winning_fighter_name	losing_fighter_name	weight_class	method_of_victory	specific_victory_details	ending_round	endin
0	23	UFC Fight Night: Lee vs. Oliveira	Charles Oliveira	Kevin Lee	Lightweight	SUB	Guillotine Choke	3	
1	23	UFC Fight Night: Lee vs. Oliveira	Charles Oliveira	Kevin Lee	Lightweight	SUB	Guillotine Choke	3	
2	23	UFC Fight Night: Lee vs. Oliveira	Charles Oliveira	Kevin Lee	Lightweight	SUB	Guillotine Choke	3	
3	277	UFC Fight Night: Lee vs. Iaquinta	Charles Oliveira	Jim Miller	Lightweight	SUB	Rear Naked Choke	1	
4	941	UFC 225: Whittaker vs. Romero 2	Charles Oliveira	Clay Guida	Lightweight	SUB	Guillotine Choke	1	

In [28]:
```
df2.tail(5)
```

Out[28]:

	fight_id	fight_title	winning_fighter_name	losing_fighter_name	weight_class	method_of_victory	specific_victory_details	ending_round	endi
15	8130	UFC 262: Oliveira vs. Chandler	Charles Oliveira	Michael Chandler	Lightweight	KO/TKO	Punch	2	
16	8130	UFC 262: Oliveira vs. Chandler	Charles Oliveira	Michael Chandler	Lightweight	KO/TKO	Punch	2	
17	8651	UFC 269: Oliveira vs. Poirier	Charles Oliveira	Dustin Poirier	Lightweight	SUB	Rear Naked Choke	3	
18	8651	UFC 269: Oliveira vs. Poirier	Charles Oliveira	Dustin Poirier	Lightweight	SUB	Rear Naked Choke	3	
19	8651	UFC 269: Oliveira vs. Poirier	Charles Oliveira	Dustin Poirier	Lightweight	SUB	Rear Naked Choke	3	

In [29]:
```
# creating the group frame

group_frame_wins = df2[['fighter_name', 'significant_strikes_standing_attempted', 'significant_strikes_standing
group_frame_wins.head(3)
```

Out[29]:

	fighter_name	significant_strikes_standing_attempted	significant_strikes_standing_landed	significant_strikes_head_attempted	significant_strikes_b
0	Charles Oliveira	373	191	359	

In [30]:
```
group_frame_wins.fighter_name = 'Champ Oliveira'
print(group_frame_wins)
```

```
     fighter_name  significant_strikes_standing_attempted  \
0  Champ Oliveira                                     373

   significant_strikes_standing_landed  significant_strikes_head_attempted  \
0                                  191                                 359

   significant_strikes_body_attempted  significant_strikes_leg_attempted  \
0                                 103                                 52

   significant_strikes_ground_attempted  significant_strikes_clinch_attempted  \
0                                    83                                    58

   knockdowns  standing_time
0           4            888
```

In [31]:
```
group_frame_wins['head_ratio'] = group_frame_wins.significant_strikes_head_attempted / (group_frame_wins.signif
group_frame_wins['body_ratio'] = group_frame_wins.significant_strikes_body_attempted / (group_frame_wins.signif
group_frame_wins['leg_ratio'] = group_frame_wins.significant_strikes_leg_attempted / (group_frame_wins.signific
```

In [74]:
```
ratio_frame_wins = group_frame_wins[['fighter_name', 'head_ratio', 'body_ratio', 'leg_ratio' ]]
print('Mix Strikes')
ratio_frame_wins
```

```
Mix Strikes
```

Out[74]:

	fighter_name	head_ratio	body_ratio	leg_ratio
0	Champ Oliveira	0.698444	0.200389	0.101167

In [33]:
```
#STRIKE LOCATION
location_ratio_wins = group_frame_wins[['fighter_name', 'significant_strikes_standing_attempted', 'significant_
```

In [34]:
```
location_ratio_wins['standing_ratio'] = location_ratio_wins.significant_strikes_standing_attempted / (location_
location_ratio_wins['ground_ratio'] = location_ratio_wins.significant_strikes_ground_attempted / (location_rati
location_ratio_wins['clinch_ratio'] = location_ratio_wins.significant_strikes_clinch_attempted / (location_rati

print('Strike Location')
print(location_ratio_wins[['fighter_name', 'standing_ratio', 'ground_ratio', 'clinch_ratio']])
```

```
Strike Location
     fighter_name  standing_ratio  ground_ratio  clinch_ratio
0  Champ Oliveira        0.725681      0.161479       0.11284
```

In [35]:
```
# Turn standing time to minutes
group_frame_wins['standing_time']=group_frame_wins.standing_time/60
```

In [36]:
```
#SApSM
sapsm_wins = group_frame_wins[['fighter_name', 'significant_strikes_standing_attempted','standing_time', 'knock
```

In [75]:
```
#SApSM_Ratio
sapsm_wins['SApSM'] = sapsm_wins.significant_strikes_standing_attempted / sapsm_wins.standing_time
print('Standing Attempts per Standing minute')
sapsm_wins
```

```
Standing Attempts per Standing minute
```

Out[75]:

	fighter_name	significant_strikes_standing_attempted	standing_time	knockdowns	SApSM
0	Champ Oliveira	373	14.8	4	25.202703

In [38]:
```
#SLpSM
slpsm_wins = group_frame_wins[['fighter_name', 'significant_strikes_standing_landed','standing_time', 'knockdow
```

In [76]:
```
#SLpSM Ratio
slpsm_wins['SLpSM'] = slpsm_wins.significant_strikes_standing_landed / slpsm_wins.standing_time
print('Standing Lands per standing minute')
slpsm_wins
```

```
Standing Lands per standing minute
```

Out[76]:

	fighter_name	significant_strikes_standing_landed	standing_time	knockdowns	SLpSM
0	Champ Oliveira	191	14.8	4	12.905405

In [78]:
```
accuracy_frame_wins = group_frame_wins[['significant_strikes_standing_landed', 'significant_strikes_standing_at

accuracy_frame_wins['standing_accuracy'] = accuracy_frame_wins.significant_strikes_standing_landed/accuracy_fra
print('Overall Career Accuracy')
accuracy_frame_wins
```

```
Overall Career Accuracy
```

Out[78]:

	fighter_name	significant_strikes_standing_landed	significant_strikes_standing_attempted	standing_accuracy
0	Champ Oliveira	191	373	51.206434

In [79]:
```
# Minutes Standing per Knockdown
kd_frame_wins = group_frame_wins[['standing_time', 'knockdowns', 'fighter_name']].groupby('fighter_name').sum()
kd_frame_wins['SMpKD'] = kd_frame_wins.standing_time / kd_frame_wins.knockdowns
print('Standing Minutes per Knockdown')
kd_frame_wins
```

```
Standing Minutes per Knockdown
```

Out[79]:

	fighter_name	standing_time	knockdowns	SMpKD
0	Champ Oliveira	14.8	4	3.7

In [80]:
```
kdstrike_frame_wins = group_frame_wins[['significant_strikes_standing_landed', 'knockdowns', 'fighter_name']].g

kdstrike_frame_wins['SLpKD'] = kdstrike_frame_wins.significant_strikes_standing_landed / kdstrike_frame_wins.kn
print('Strike Lands per knockdown')
kdstrike_frame_wins
```

```
Strike Lands per knockdown
```

Out[80]:

	fighter_name	significant_strikes_standing_landed	knockdowns	SLpKD
0	Champ Oliveira	191	4	47.75

In [81]:
```
#striking accuray per round

sapr_wins = df2[['total_strikes_landed', 'total_strikes_attempted', 'fighter_name', 'round_number', 'significar
sapr_wins['standing_time']=sapr_wins.standing_time/60
sapr_wins['standing_accuracy'] = (sapr_wins.total_strikes_landed / sapr_wins.total_strikes_attempted) * 100

sapr_wins.fighter_name = 'Champ Oliveira'
# graphs for head accuracy, body accuracy and leg accuracy by round

sapr_wins['head_accuracy']=(sapr_wins.significant_strikes_head_landed/sapr_wins.significant_strikes_head_attemp
sapr_wins['body_accuracy']=(sapr_wins.significant_strikes_body_landed/sapr_wins.significant_strikes_body_attemp
sapr_wins['leg_accuracy']=(sapr_wins.significant_strikes_leg_landed/sapr_wins.significant_strikes_leg_attempted

print('Standing Accuracy by round')
sapr_wins[['fighter_name', 'round_number', 'standing_accuracy', 'head_accuracy', 'body_accuracy', 'leg_accuracy
```

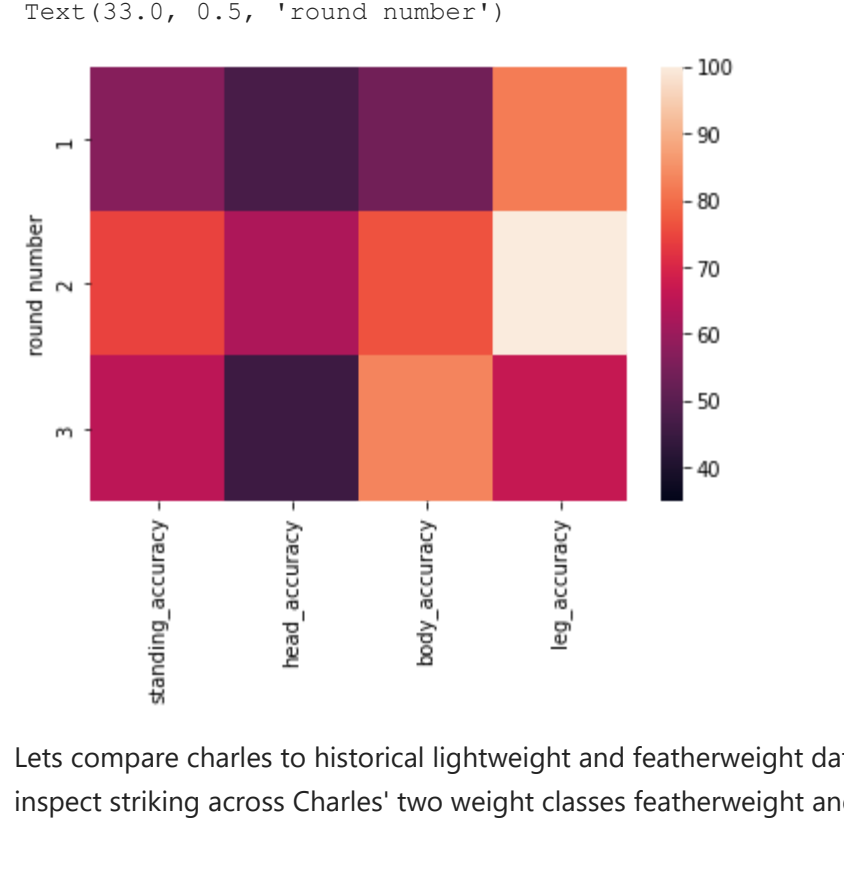
```
Standing Accuracy by round
```

Out[81]:

	fighter_name	round_number	standing_accuracy	head_accuracy	body_accuracy	leg_accuracy
0	Champ Oliveira	1	56.976744	47.290640	53.968254	82.051282
1	Champ Oliveira	2	74.561404	62.686567	76.470588	100.000000
2	Champ Oliveira	3	65.116279	45.454545	83.333333	66.666667

In [44]:
```
heat_map_ = sapr_wins[['standing_accuracy', 'head_accuracy', 'body_accuracy', 'leg_accuracy']]
heat_map_.index += 1
sns.heatmap(data=heat_map_, vmin=35)
plt.ylabel('round number')
```

Out[44]:
```
Text(33.0, 0.5, 'round number')
```



Lets compare charles to historical lightweight and featherweight data. This is used to compare striking by a weightclass basis. First lets inspect striking across Charles' two weight classes featherweight and lightweight

In [45]:

```
df_fw = pd.read_csv('fw_historical.csv', error_bad_lines=False)
df_fw.dropna()

df_fw = df_fw.loc[df_fw.control_time != '--']


df_fw['seconds'] = df_fw['control_time'].str.split(':',expand=True)[1]
df_fw['minutes'] = df_fw['control_time'].str.split(':',expand=True)[0]

df_fw['seconds'] = df_fw['seconds'].astype(dtype=int)
df_fw['minutes'] = df_fw['minutes'].astype(dtype=int)

df_fw['control_time'] = (df_fw['minutes'] * 60) + df_fw['seconds']
df_fw = df_fw.drop(columns=['seconds','minutes'])


df_fw['seconds'] = df_fw.ending_time.str.split(':',expand=True)[1]
df_fw['minutes'] = df_fw.ending_time.str.split(':',expand=True)[0]

df_fw['seconds'] = df_fw['seconds'].astype(int)
df_fw['minutes'] = df_fw['minutes'].astype(int)

df_fw['ending_time'] = (df_fw['minutes'] * 60) + df_fw['seconds']
df_fw = df_fw.drop(columns=['seconds','minutes'])

df_fw['standing_time'] = df_fw.ending_time - df_fw.control_time

df_fw.loc[df_fw.round_number != df_fw.ending_round,'ending_time'] = 300
print(df_fw.columns)
```

```
Index(['fight_id', 'fight_title', 'winning_fighter_name',
       'losing_fighter_name', 'weight_class', 'method_of_victory',
       'specific_victory_details', 'ending_round', 'ending_time', 'fight_url',
       'unique_round_id', 'unique_fight_id', 'unique_event_id', 'round_number',
       'fighter_name', 'knockdowns', 'total_strikes_landed',
       'total_strikes_attempted', 'takedowns', 'takedowns_attempted',
       'submission_attempts', 'guard_passes', 'reversals', 'control_time',
       'unique_round_id.1', 'unique_fight_id.1', 'unique_event_id.1',
       'round_number.1', 'fighter_name.1', 'significant_strikes_head_landed',
       'significant_strikes_head_attempted', 'significant_strikes_body_landed',
       'significant_strikes_body_attempted', 'significant_strikes_leg_landed',
       'significant_strikes_leg_attempted',
       'significant_strikes_standing_landed',
       'significant_strikes_standing_attempted',
       'significant_strikes_clinch_landed',
       'significant_strikes_clinch_attempted',
       'significant_strikes_ground_landed',
       'significant_strikes_ground_attempted', 'standing_time'],
      dtype='object')
```

In [46]:

```
df_fw.head()
```

Out[46]:

	fight_id	fight_title	winning_fighter_name	losing_fighter_name	weight_class	method_of_victory	specific_victory_details	ending_round
0	710	UFC Fight Night: Belfort vs Henderson	Thiago Tavares	Clay Guida	Featherweight	SUB	Guillotine Choke	1
1	710	UFC Fight Night: Belfort vs Henderson	Thiago Tavares	Clay Guida	Featherweight	SUB	Guillotine Choke	1
2	1913	UFC 199: Rockhold vs. Bisping 2	Brian Ortega	Clay Guida	Featherweight	KO/TKO	Knee	3
3	1913	UFC 199: Rockhold vs. Bisping 2	Brian Ortega	Clay Guida	Featherweight	KO/TKO	Knee	3
4	1913	UFC 199: Rockhold vs. Bisping 2	Brian Ortega	Clay Guida	Featherweight	KO/TKO	Knee	3

In [47]:

```
df_fw.tail()
```

Out[47]:

	fight_id	fight_title	winning_fighter_name	losing_fighter_name	weight_class	method_of_victory	specific_victory_details	ending_round
995	1276	UFC Fight Night: Lewis vs. Dos Santos	Grant Dawson	Julian Erosa	Featherweight	U-DEC	NaN	3
996	1276	UFC Fight Night: Lewis vs. Dos Santos	Grant Dawson	Julian Erosa	Featherweight	U-DEC	NaN	3
997	1276	UFC Fight Night: Lewis vs. Dos Santos	Grant Dawson	Julian Erosa	Featherweight	U-DEC	NaN	3
998	2242	UFC Fight Night: Thompson vs. Pettis	Luis Pena	Steven Peterson	Featherweight	U-DEC	NaN	3
999	2242	UFC Fight Night: Thompson vs. Pettis	Luis Pena	Steven Peterson	Featherweight	U-DEC	NaN	3

In [48]:

```
df_fw.shape
```

Out[48]:

```
(978, 42)
```

In [49]:

```
# creating the group frame

group_frame_fw = df_fw[['fighter_name', 'significant_strikes_standing_attempted', 'significant_strikes_standing
group_frame_fw.head(30)
```

Out[49]:

	fighter_name	significant_strikes_standing_attempted	significant_strikes_standing_landed	significant_strikes_head_attempted
0	Akira Corassani	32	11	24
1	Alex Caceres	580	250	509
2	Alex White	180	81	137
3	Alexander Volkanovski	513	283	393
4	Alexandre Franca Nogueira	23	4	21
5	Andre Fili	1055	385	984
6	Andy Ogle	141	48	117
7	Anistavio Medeiros	18	8	12
8	Arnold Allen	549	230	506
9	Artem Lobov	813	333	621
10	BJ Penn	84	16	88
11	Bobby Moffett	51	18	47
12	Brandon Davis	242	78	186
13	Brian Ortega	816	262	767
14	Calvin Kattar	382	154	368
15	Chad Mendes	76	27	122
16	Chan Sung Jung	453	148	483
17	Charles Jourdain	115	52	84
18	Charles Oliveira	40	12	42
19	Charles Rosa	344	115	275
20	Chas Skelly	301	97	347
21	Chris Fishgold	122	35	92
22	Chris Gruetzemacher	29	15	29
23	Clay Guida	339	83	304
24	Cole Miller	931	299	810
25	Conor McGregor	126	57	143
26	Cub Swanson	1377	675	1151
27	Dan Hooker	333	125	366
28	Dan Ige	70	30	97
29	Daniel Pineda	59	32	64

In [50]:

```
group_frame_fw['head_ratio'] = group_frame_fw.significant_strikes_head_attempted / (group_frame_fw.significant_
group_frame_fw['body_ratio'] = group_frame_fw.significant_strikes_body_attempted / (group_frame_fw.significant_
group_frame_fw['leg_ratio'] = group_frame_fw.significant_strikes_leg_attempted / (group_frame_fw.significant_st
```

In [51]:

```
ratio_frame_fw = group_frame_fw[['fighter_name', 'head_ratio', 'body_ratio', 'leg_ratio' ]]
print('Mix Strikes')
ratio_frame_fw.head(148)
```

Mix Strikes

Out[51]:

	fighter_name	head_ratio	body_ratio	leg_ratio
0	Akira Corassani	0.727273	0.121212	0.151515
1	Alex Caceres	0.795312	0.135937	0.068750
2	Alex White	0.678218	0.232673	0.089109
3	Alexander Volkanovski	0.670648	0.098976	0.230375
4	Alexandre Franca Nogueira	0.840000	0.000000	0.160000
...	...	...	...	...
143	Yair Rodriguez	0.747021	0.123740	0.129239
144	Yaotzin Meza	0.704918	0.131148	0.163934
145	Yui Chul Nam	0.700000	0.227778	0.072222
146	Zabit Magomedsharipov	0.753425	0.121005	0.125571
147	Zubaira Tukhugov	0.944444	0.055556	0.000000

148 rows × 4 columns

In [52]:

```
print('Average Strike Mix ratiosof Featherweight')
ratio_frame_fw.mean()
```

Average Strike Mix ratiosof Featherweight

Out[52]:

```
head_ratio    0.783072
body_ratio    0.125097
leg_ratio     0.091831
dtype: float64
```

In [53]:

```
#STRIKE LOCATION
location_ratio_fw = group_frame_fw[['fighter_name', 'significant_strikes_standing_attempted', 'significant_stri
```

In [54]:

```
location_ratio_fw['standing_ratio'] = location_ratio_fw.significant_strikes_standing_attempted / (location_rati
location_ratio_fw['ground_ratio'] = location_ratio_fw.significant_strikes_ground_attempted / (location_ratio_fw
location_ratio_fw['clinch_ratio'] = location_ratio_fw.significant_strikes_clinch_attempted / (location_ratio_fw

print('Strike Location')
location_ratio_fw[['fighter_name', 'standing_ratio', 'ground_ratio', 'clinch_ratio']]
```

Strike Location

Out[54]:

	fighter_name	standing_ratio	ground_ratio	clinch_ratio
0	Akira Corassani	0.969697	0.000000	0.030303
1	Alex Caceres	0.906250	0.031250	0.062500
2	Alex White	0.891089	0.039604	0.069307
3	Alexander Volkanovski	0.875427	0.046075	0.078498
4	Alexandre Franca Nogueira	0.920000	0.000000	0.080000
...	...	...	...	...
143	Yair Rodriguez	0.808433	0.143905	0.047663
144	Yaotzin Meza	0.918033	0.000000	0.081967
145	Yui Chul Nam	0.500000	0.183333	0.316667
146	Zabit Magomedsharipov	0.831050	0.127854	0.041096
147	Zubaira Tukhugov	0.611111	0.388889	0.000000

148 rows × 4 columns

In [55]:

```
print('Average FW Fight locations')
location_ratio_fw[['fighter_name', 'standing_ratio', 'ground_ratio', 'clinch_ratio']].mean()
```

Average FW Fight locations

Out[55]:

```
standing_ratio    0.808510
ground_ratio      0.096566
clinch_ratio      0.094923
dtype: float64
```

In [56]:

```
# Turn standing time to minutes
group_frame_fw['standing_time']=group_frame_fw.standing_time/60
```

In [57]:

```
#SApSM
sapsm_fw = group_frame_fw[['fighter_name', 'significant_strikes_standing_attempted','standing_time', 'knockdown
```

In [58]:

```
#SApSM_Ratio
sapsm_fw['SApSM'] = sapsm_fw.significant_strikes_standing_attempted / sapsm_fw.standing_time
print('Standing Attempts per Standing minute')
print(sapsm_fw)
```

```
Standing Attempts per Standing minute
                  fighter_name  significant_strikes_standing_attempted  \
0              Akira Corassani                                      32
1                 Alex Caceres                                     580
2                   Alex White                                     180
3        Alexander Volkanovski                                     513
4    Alexandre Franca Nogueira                                      23
..                         ...                                     ...
143             Yair Rodriguez                                     882
144               Yaotzin Meza                                      56
145                Yui Chul Nam                                      90
146      Zabit Magomedsharipov                                     364
147           Zubaira Tukhugov                                      22

     standing_time  knockdowns      SApSM
0         3.816667           0   8.384279
1        68.950000           2   8.411893
2        15.283333           1  11.777535
3        45.500000           3  11.274725
4         6.383333           0   3.603133
..             ...         ...        ...
143      86.750000           3  10.167147
144      12.966667           0   4.318766
145      18.316667           0   4.913558
146      25.966667           0  14.017972
147       3.200000           2   6.875000

[148 rows x 5 columns]
```

In [59]:

```
print(f'Average strike Attempts per standing minute of FW: {sapsm_fw.SApSM.mean()}')
```

```
Average strike Attempts per standing minute of FW: 18.646684769538453
```

In [60]:

```
#SLpSM
slpsm_fw = group_frame_fw[['fighter_name', 'significant_strikes_standing_landed','standing_time', 'knockdowns']
```

In [61]:

```
#SLpSM Ratio
slpsm_fw['SLpSM'] = slpsm_fw.significant_strikes_standing_landed / slpsm_fw.standing_time
print('Standing Lands per standing minute')
print(slpsm_fw)
```

```
Standing Lands per standing minute
                  fighter_name  significant_strikes_standing_landed  \
0              Akira Corassani                                   11
1                 Alex Caceres                                  250
2                   Alex White                                   81
3        Alexander Volkanovski                                  283
4    Alexandre Franca Nogueira                                    4
..                         ...                                  ...
143             Yair Rodriguez                                  366
144               Yaotzin Meza                                   21
145                Yui Chul Nam                                   26
146      Zabit Magomedsharipov                                  159
147           Zubaira Tukhugov                                   15

     standing_time  knockdowns     SLpSM
0         3.816667           0  2.882096
1        68.950000           2  3.625816
2        15.283333           1  5.299891
3        45.500000           3  6.219780
4         6.383333           0  0.626632
..             ...         ...       ...
143      86.750000           3  4.219020
144      12.966667           0  1.619537
145      18.316667           0  1.419472
146      25.966667           0  6.123235
147       3.200000           2  4.687500

[148 rows x 5 columns]
```

In [62]:

```
print(f'Average strike Lands per striking minute of FW: {slpsm_fw.SLpSM.mean()}')
```

```
Average strike Lands per striking minute of FW: 8.122281753160623
```

In [63]:

```
accuracy_frame_fw = group_frame_fw[['significant_strikes_standing_landed', 'significant_strikes_standing_attemp

accuracy_frame_fw['standing_accuracy'] = accuracy_frame_fw.significant_strikes_standing_landed/accuracy_frame_f
print('Overall Career Accuracy')
print(accuracy_frame_fw)
```

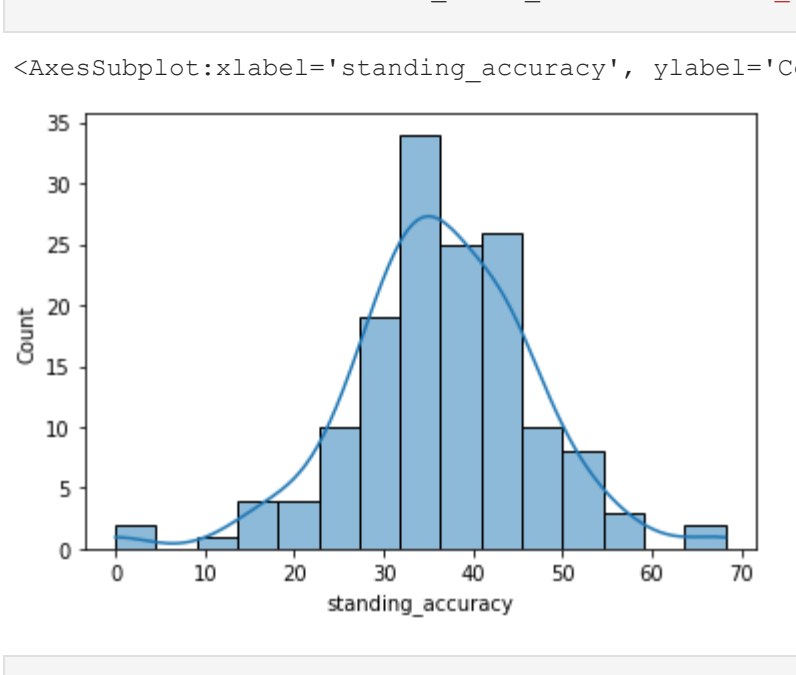
```
Overall Career Accuracy
                  fighter_name  significant_strikes_standing_landed  \
0              Akira Corassani                                   11
1                 Alex Caceres                                  250
2                   Alex White                                   81
3        Alexander Volkanovski                                  283
4    Alexandre Franca Nogueira                                    4
..                         ...                                  ...
143             Yair Rodriguez                                  366
144               Yaotzin Meza                                   21
145                Yui Chul Nam                                   26
146      Zabit Magomedsharipov                                  159
147           Zubaira Tukhugov                                   15

     significant_strikes_standing_attempted  standing_accuracy
0                                        32          34.375000
1                                       580          43.103448
2                                       180          45.000000
3                                       513          55.165692
4                                        23          17.391304
..                                      ...                ...
143                                     882          41.496599
144                                      56          37.500000
145                                      90          28.888889
146                                     364          43.681319
147                                      22          68.181818

[148 rows x 4 columns]
```

In [64]:

```
sns.histplot(data=accuracy_frame_fw, x="standing_accuracy", kde=True)
```

Out[64]:

```
<AxesSubplot:xlabel='standing_accuracy', ylabel='Count'>
```



In [65]:

```
#find the average accuracy for featherweight
perc = "{:.0f}".format(accuracy_frame_fw.standing_accuracy.mean())
perc = int(perc)
print(f'average accuracy for featherweight: {perc}%')
```

```
average accuracy for featherweight: 36%
```

In [66]:

```
olive_performances = df1[[ 'significant_strikes_standing_attempted', 'significant_strikes_standing_landed', 'un

perc = 0.36
#print(olive_performances.columns)
olive_performances['significant_strikes_standing_attempted'] = olive_performances['significant_strikes_standing
olive_performances['expected_strike_lands'] = olive_performances.significant_strikes_standing_attempted * perc

olive_performances.head(20)
```

```
<ipython-input-66-ff76b424edf5>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  olive_performances['significant_strikes_standing_attempted'] = olive_performances['significant_strikes_standi
ng_attempted'].astype(int)
<ipython-input-66-ff76b424edf5>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  olive_performances['expected_strike_lands'] = olive_performances.significant_strikes_standing_attempted * per
c
```

Out[66]:

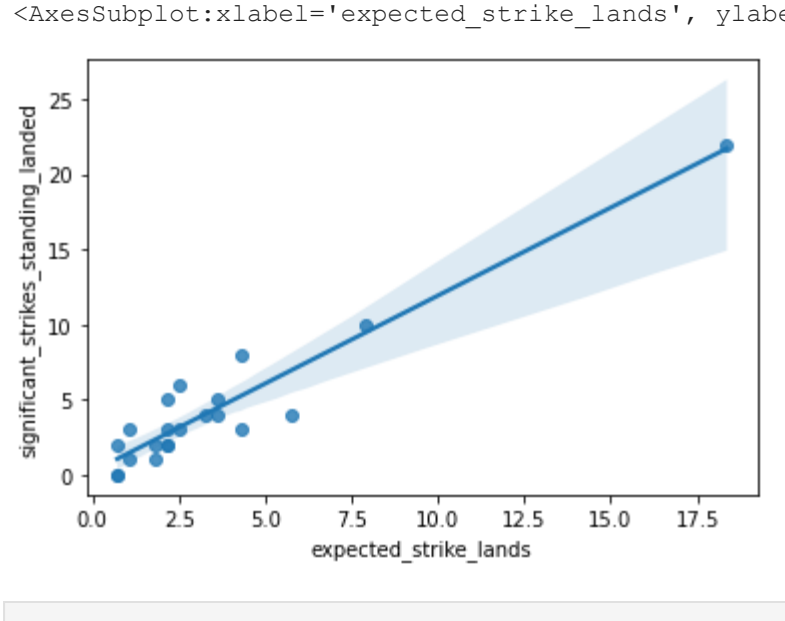
	significant_strikes_standing_attempted	significant_strikes_standing_landed	unique_round_id	event_date	winning_fighter_name	losing_fighter_
0	10	5	583	2015-12-19	Charles Oliveira	Myle
1	16	4	848	2015-08-23	Max Holloway	Charles O
2	2	0	2680	2014-06-28	Charles Oliveira	Hatsu
3	12	3	2682	2014-06-28	Charles Oliveira	Hatsu
4	6	2	3104	2017-04-08	Charles Oliveira	Will E
5	7	3	14806	2016-08-27	Anthony Pettis	Charles O
6	51	22	14808	2016-08-27	Anthony Pettis	Charles O
7	5	2	14810	2016-08-27	Anthony Pettis	Charles O
8	3	1	14814	2016-11-05	Ricardo Lamas	Charles O
9	6	5	14816	2016-11-05	Ricardo Lamas	Charles O
10	3	3	14817	2017-12-02	Paul Felder	Charles O
11	2	0	14819	2017-12-02	Paul Felder	Charles O
12	10	4	26814	2014-12-12	Charles Oliveira	Jeremy Ste
13	7	6	26816	2014-12-12	Charles Oliveira	Jeremy Ste
14	22	10	26818	2014-12-12	Charles Oliveira	Jeremy Ste
15	12	8	28006	2015-05-30	Charles Oliveira	Nik
16	2	2	28008	2015-05-30	Charles Oliveira	Nik
17	6	3	28010	2015-05-30	Charles Oliveira	Nik
18	5	1	28979	2014-02-15	Charles Oliveira	Andy
19	6	2	28981	2014-02-15	Charles Oliveira	Andy

In [67]:

```
sns.regplot(data=olive_performances, y='significant_strikes_standing_landed', x='expected_strike_lands')
```

Out[67]:

```
<AxesSubplot:xlabel='expected_strike_lands', ylabel='significant_strikes_standing_landed'>
```



In [68]:

```
fw_performance = df_fw[[ 'significant_strikes_standing_attempted', 'significant_strikes_standing_landed', 'uniq

perc = 0.36
#print(fw_performance.columns)
fw_performance['significant_strikes_standing_attempted'] = fw_performance['significant_strikes_standing_attempt
fw_performance['expected_strike__standing_lands'] = fw_performance.significant_strikes_standing_attempted * per

fw_performance.head(20)
```

```
<ipython-input-68-bc102304186d>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  fw_performance['significant_strikes_standing_attempted'] = fw_performance['significant_strikes_standing_attem
pted'].astype(int)
<ipython-input-68-bc102304186d>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  fw_performance['expected_strike__standing_lands'] = fw_performance.significant_strikes_standing_attempted * p
erc
```

Out[68]:

	significant_strikes_standing_attempted	significant_strikes_standing_landed	unique_round_id	winning_fighter_name	losing_fighter_name
0	11	3	451	Thiago Tavares	Clay Guida
1	7	2	452	Thiago Tavares	Clay Guida
2	29	5	453	Brian Ortega	Clay Guida
3	53	15	454	Brian Ortega	Clay Guida
4	74	24	455	Brian Ortega	Clay Guida
5	70	23	456	Brian Ortega	Clay Guida
6	82	31	457	Brian Ortega	Clay Guida
7	65	20	458	Brian Ortega	Clay Guida
8	61	34	465	Yair Rodriguez	Jeremy Stephens
9	21	9	466	Yair Rodriguez	Jeremy Stephens
10	13	5	467	Yair Rodriguez	Jeremy Stephens
11	8	1	468	Yair Rodriguez	Jeremy Stephens
12	26	15	469	Yair Rodriguez	Jeremy Stephens
13	22	9	470	Yair Rodriguez	Jeremy Stephens
14	2	2	471	Yair Rodriguez	Jeremy Stephens
15	1	0	472	Yair Rodriguez	Jeremy Stephens
16	41	16	473	Jose Aldo	Jeremy Stephens
17	52	16	474	Jose Aldo	Jeremy Stephens
18	41	10	475	Renato Moicano	Jeremy Stephens
19	34	20	476	Renato Moicano	Jeremy Stephens

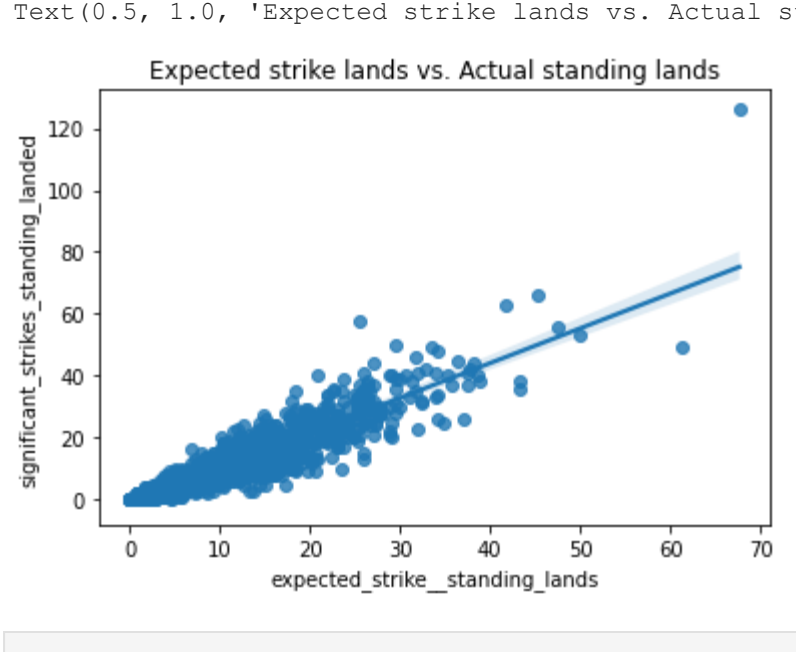
Expected Strikes is Strike Attempts for every round multiplied by the Average standing accuracy for Featherweight which is 36%

In [69]:

```
sns.regplot(data=fw_performance, y='significant_strikes_standing_landed', x='expected_strike__standing_lands')
plt.title('Expected strike lands vs. Actual standing lands')
```

Out[69]:

```
Text(0.5, 1.0, 'Expected strike lands vs. Actual standing lands')
```

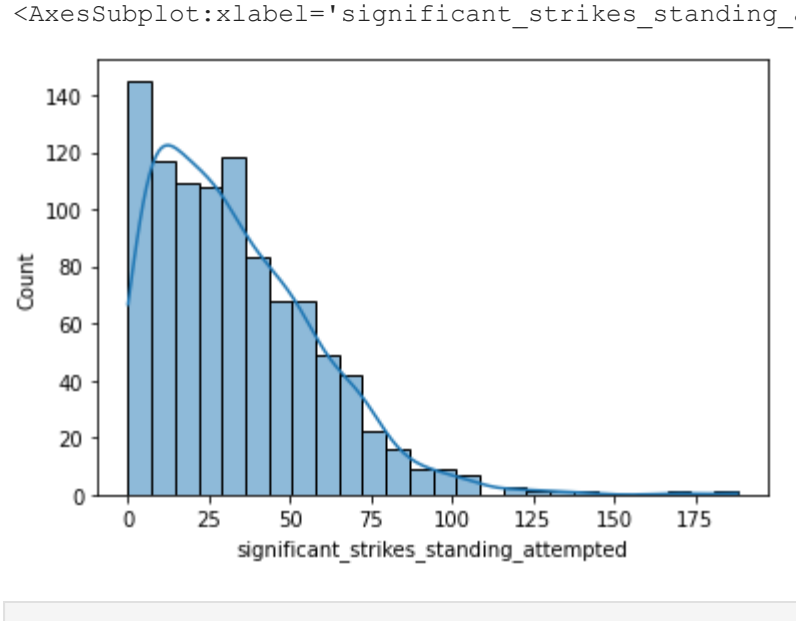


In [70]:

```
sns.histplot(data=fw_performance, x="significant_strikes_standing_attempted", kde=True, )
#Numbers are lower because these are rounds performances
```

Out[70]:

```
<AxesSubplot:xlabel='significant_strikes_standing_attempted', ylabel='Count'>
```



In [71]:

```
fw_performance.loc[ fw_performance.significant_strikes_standing_attempted == fw_performance.significant_strikes
```

Out[71]:

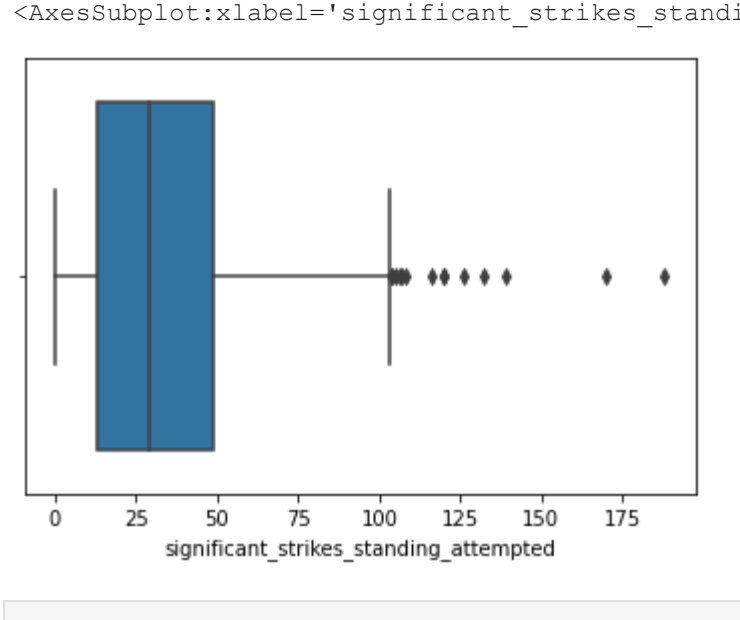
	significant_strikes_standing_attempted	significant_strikes_standing_landed	unique_round_id	winning_fighter_name	losing_fighter_name
228	188	126	2653	Max Holloway	Brian Ortega

In [72]:

```
sns.boxplot(data=fw_performance, x="significant_strikes_standing_attempted")
```

Out[72]:

```
<AxesSubplot:xlabel='significant_strikes_standing_attempted'>
```



In [82]:

```
expected_by_target = df_fw[[ 'significant_strikes_head_landed', 'significant_strikes_head_attempted', 'signific

expected_by_target['head_accuracy_fw'] = expected_by_target.significant_strikes_head_landed / expected_by_targe
expected_by_target['body_accuracy_fw'] = expected_by_target.significant_strikes_body_landed / expected_by_targe
expected_by_target['leg_accuracy_fw'] = expected_by_target.significant_strikes_leg_landed / expected_by_target.


expected_by_target = expected_by_target.dropna()
expected_by_target = expected_by_target.reset_index(drop=True)

head_accuracy_avg = expected_by_target['head_accuracy_fw'].mean()
body_accuracy_avg = expected_by_target['body_accuracy_fw'].mean()
leg_accuracy_avg = expected_by_target['leg_accuracy_fw'].mean()

expected_by_target.head(20)
```

```
<ipython-input-82-0428451930d6>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  expected_by_target['head_accuracy_fw'] = expected_by_target.significant_strikes_head_landed / expected_by_tar
get.significant_strikes_head_attempted * 100
<ipython-input-82-0428451930d6>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  expected_by_target['body_accuracy_fw'] = expected_by_target.significant_strikes_body_landed / expected_by_tar
get.significant_strikes_body_attempted * 100
<ipython-input-82-0428451930d6>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  expected_by_target['leg_accuracy_fw'] = expected_by_target.significant_strikes_leg_landed / expected_by_targe
t.significant_strikes_leg_attempted * 100
```

Out[82]:

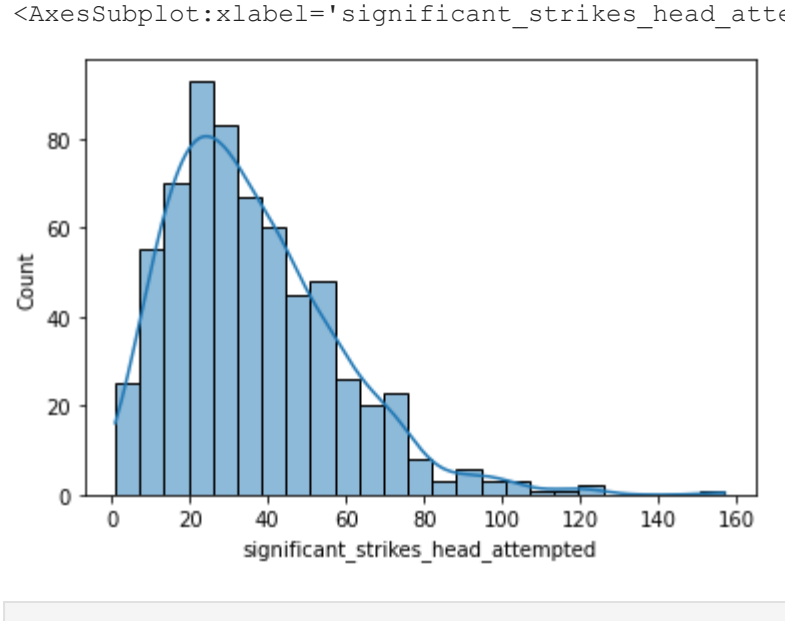
	significant_strikes_head_landed	significant_strikes_head_attempted	significant_strikes_body_landed	significant_strikes_body_attempted
0	12	47	1	4
1	16	62	6	7
2	28	78	3	4
3	18	43	9	13
4	3	14	4	4
5	37	67	2	4
6	9	19	2	2
7	10	17	3	6
8	23	43	4	4
9	21	55	3	3
10	16	55	2	3
11	3	31	1	1
12	4	11	1	3
13	9	62	4	6
14	17	38	1	1
15	16	75	2	3
16	18	29	1	1
17	7	25	2	6
18	4	25	0	1
19	3	24	2	2

In [83]:

```
sns.histplot(data=expected_by_target, x="significant_strikes_head_attempted",kde=True)
```

Out[83]:

```
<AxesSubplot:xlabel='significant_strikes_head_attempted', ylabel='Count'>
```

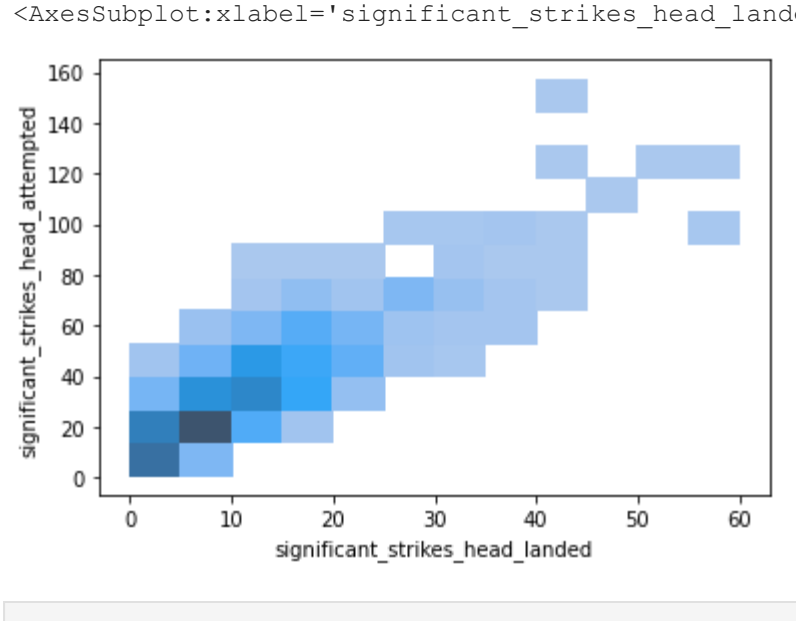


In [84]:

```
sns.histplot(data=expected_by_target, x="significant_strikes_head_landed", y="significant_strikes_head_attempte
```

Out[84]:

```
<AxesSubplot:xlabel='significant_strikes_head_landed', ylabel='significant_strikes_head_attempted'>
```

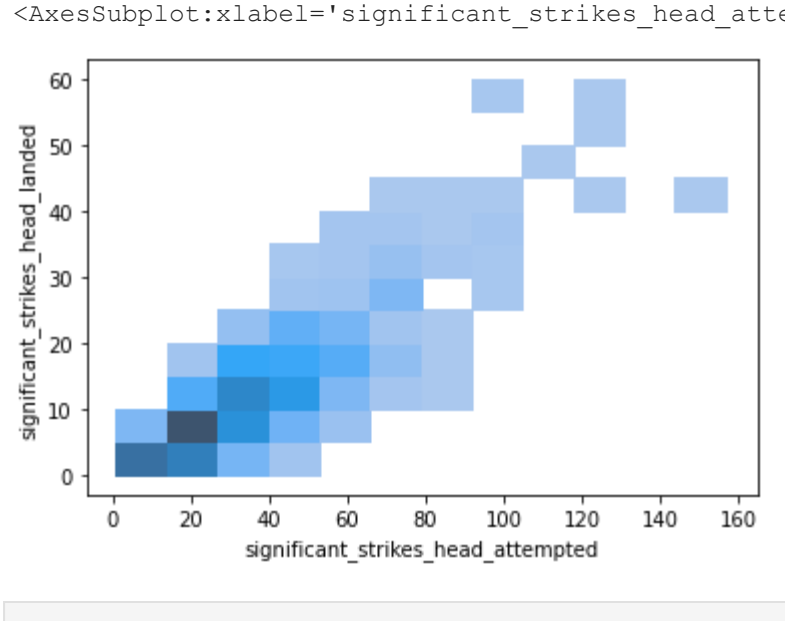


In [85]:

```
sns.histplot(data=expected_by_target, x="significant_strikes_head_attempted", y="significant_strikes_head_lande
```

Out[85]:

```
<AxesSubplot:xlabel='significant_strikes_head_attempted', ylabel='significant_strikes_head_landed'>
```

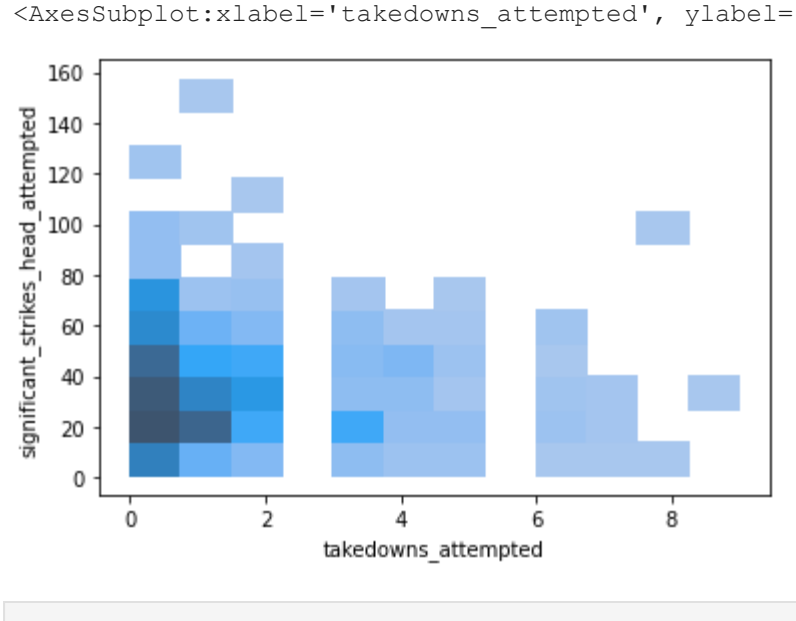


In [86]:

```
sns.histplot(data=expected_by_target, x="takedowns_attempted", y="significant_strikes_head_attempted" , bins=12
```

Out[86]:

```
<AxesSubplot:xlabel='takedowns_attempted', ylabel='significant_strikes_head_attempted'>
```

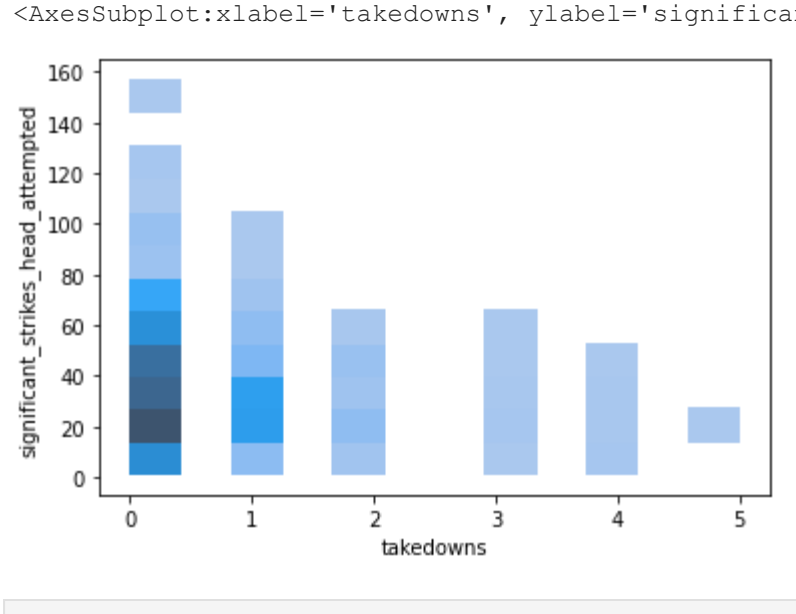


In [87]:

```
sns.histplot(data=expected_by_target, x="takedowns", y="significant_strikes_head_attempted" , bins=12)
```

Out[87]:

```
<AxesSubplot:xlabel='takedowns', ylabel='significant_strikes_head_attempted'>
```

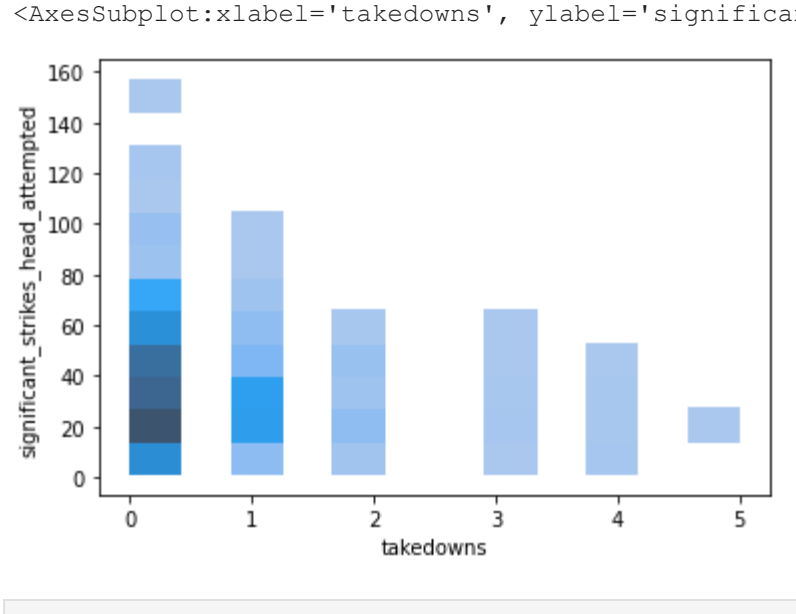


In [88]:

```
sns.histplot(data=expected_by_target, x="takedowns", y="significant_strikes_head_attempted" , bins=12)
```

Out[88]:

```
<AxesSubplot:xlabel='takedowns', ylabel='significant_strikes_head_attempted'>
```

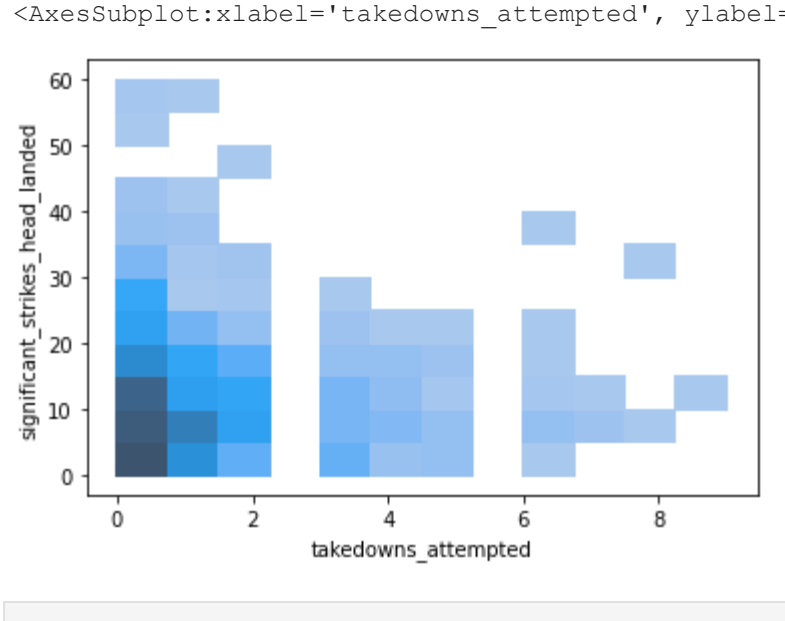


In [89]:

```
sns.histplot(data=expected_by_target, x="takedowns_attempted", y="significant_strikes_head_landed" , bins=12)
```

Out[89]:

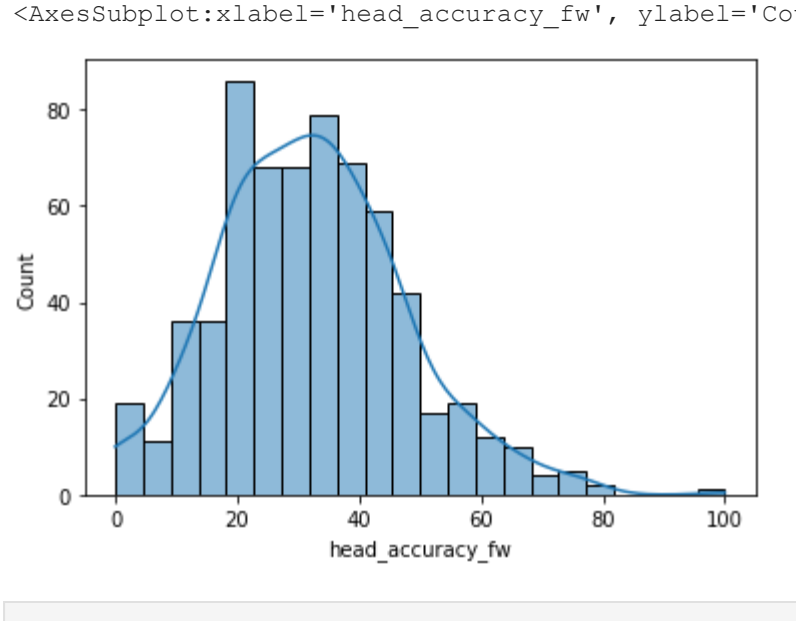
```
<AxesSubplot:xlabel='takedowns_attempted', ylabel='significant_strikes_head_landed'>
```



In [90]:

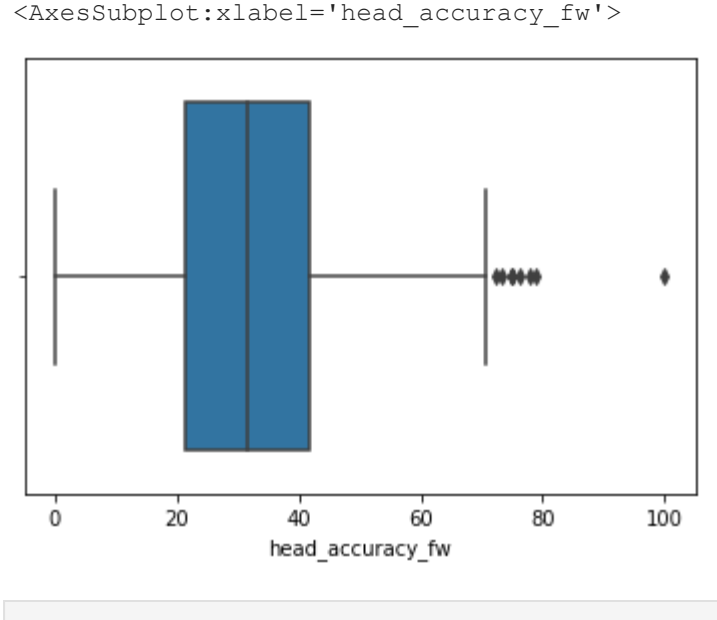
```
sns.histplot(data=expected_by_target, x="head_accuracy_fw", kde=True)
```

Out[90]: <AxesSubplot:xlabel='head_accuracy_fw', ylabel='Count'>



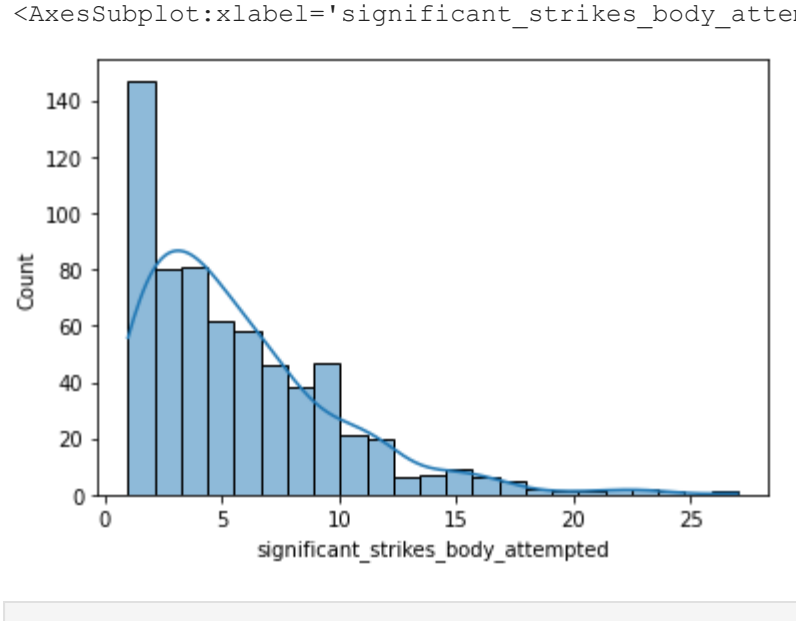
In [91]:
```
sns.boxplot(data=expected_by_target, x="head_accuracy_fw")
```

Out[91]: <AxesSubplot:xlabel='head_accuracy_fw'>



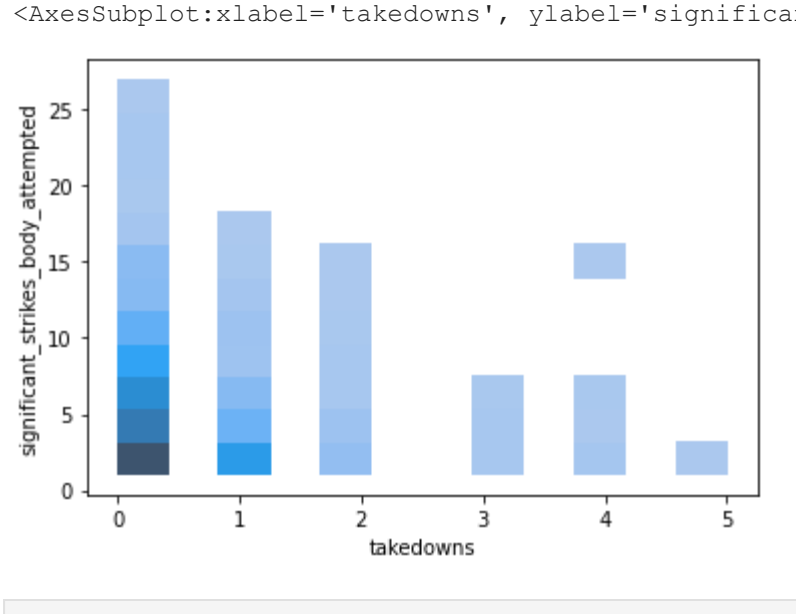
In [92]:
```
sns.histplot(data=expected_by_target, x="significant_strikes_body_attempted", kde=True)
```

Out[92]: <AxesSubplot:xlabel='significant_strikes_body_attempted', ylabel='Count'>



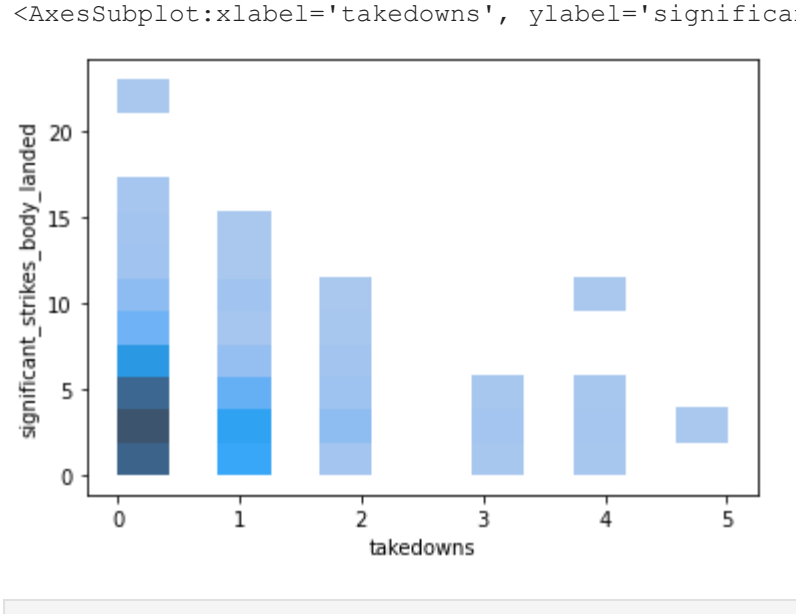
In [93]:
```
sns.histplot(data=expected_by_target, x="takedowns", y="significant_strikes_body_attempted" , bins=12)
```

Out[93]: <AxesSubplot:xlabel='takedowns', ylabel='significant_strikes_body_attempted'>



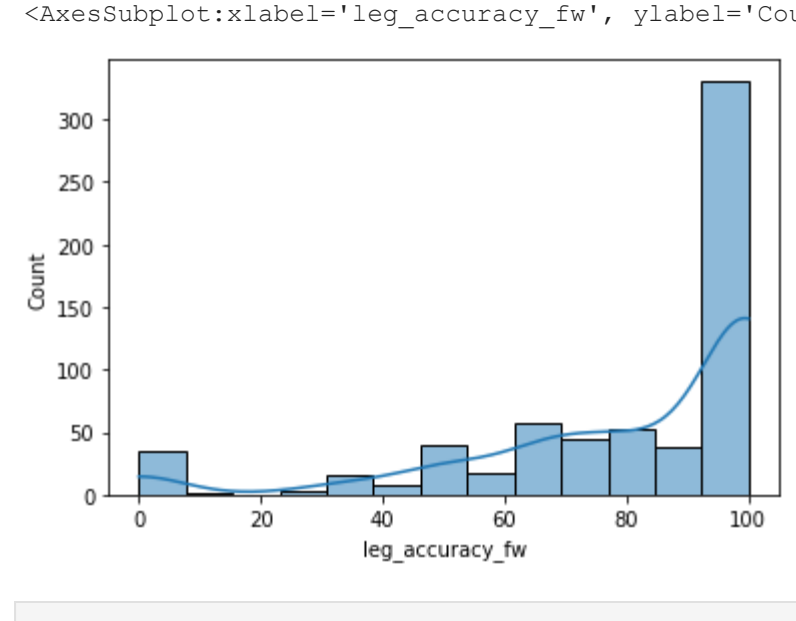
In [94]:
```
sns.histplot(data=expected_by_target, x="takedowns", y="significant_strikes_body_landed" , bins=12)
```

Out[94]: <AxesSubplot:xlabel='takedowns', ylabel='significant_strikes_body_landed'>



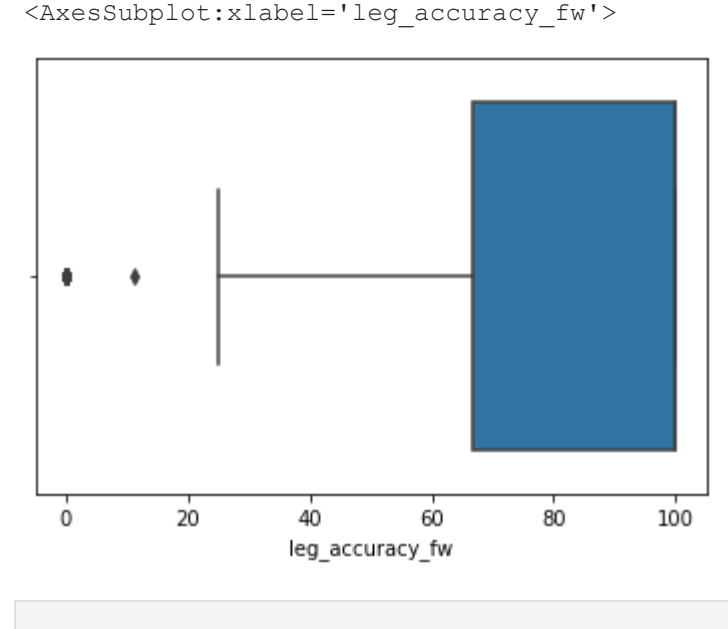
In [95]:
```
sns.histplot(data=expected_by_target, x="body_accuracy_fw", kde=True)
```

Out[95]: <AxesSubplot:xlabel='body_accuracy_fw', ylabel='Count'>



In [96]:
```
sns.boxplot(data=expected_by_target, x="body_accuracy_fw")
```

Out[96]: <AxesSubplot:xlabel='body_accuracy_fw'>

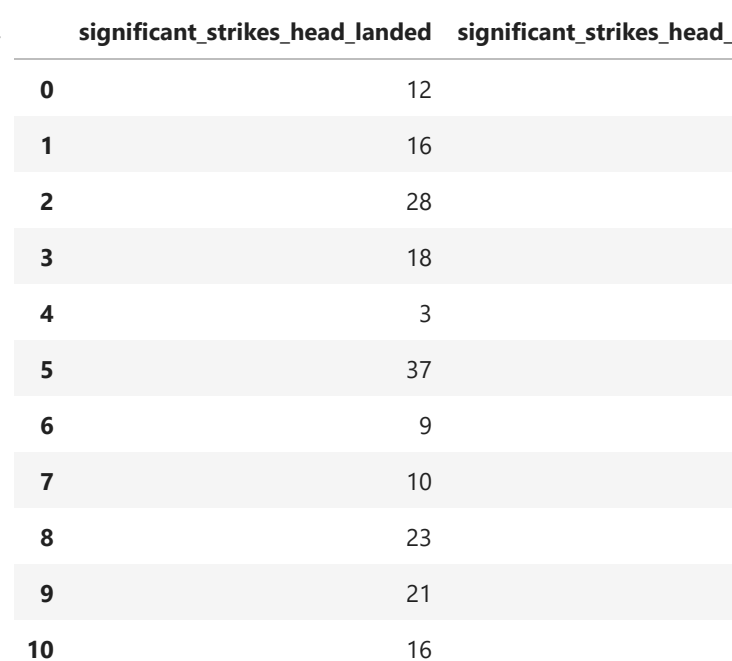


In [97]:
```
sns.histplot(data=expected_by_target, x="leg_accuracy_fw", kde=True)
```

Out[97]: <AxesSubplot:xlabel='leg_accuracy_fw', ylabel='Count'>

In [98]:
```
sns.boxplot(data=expected_by_target, x="leg_accuracy_fw")
```

Out[98]: <AxesSubplot:xlabel='leg_accuracy_fw'>



In [99]:
```
expected_by_target.body_accuracy_fw.median()
```

Out[99]: 66.66666666666666

In [100…
```
print(head_accuracy_avg)
print(body_accuracy_avg)
print(leg_accuracy_avg)

#print(fw_performance.columns)
expected_by_target['significant_strikes_head_attempted'] = expected_by_target['significant_strikes_head_attempt
expected_by_target['expected_head_lands'] = expected_by_target.significant_strikes_head_attempted * head_accura


expected_by_target['significant_strikes_body_attempted'] = expected_by_target['significant_strikes_body_attempt
expected_by_target['expected_body_lands'] = expected_by_target.significant_strikes_body_attempted * body_accura


expected_by_target['significant_strikes_leg_attempted'] = expected_by_target['significant_strikes_leg_attempted
expected_by_target['expected_leg_lands'] = expected_by_target.significant_strikes_leg_attempted * leg_accuracy_

expected_by_target.head(20)
```

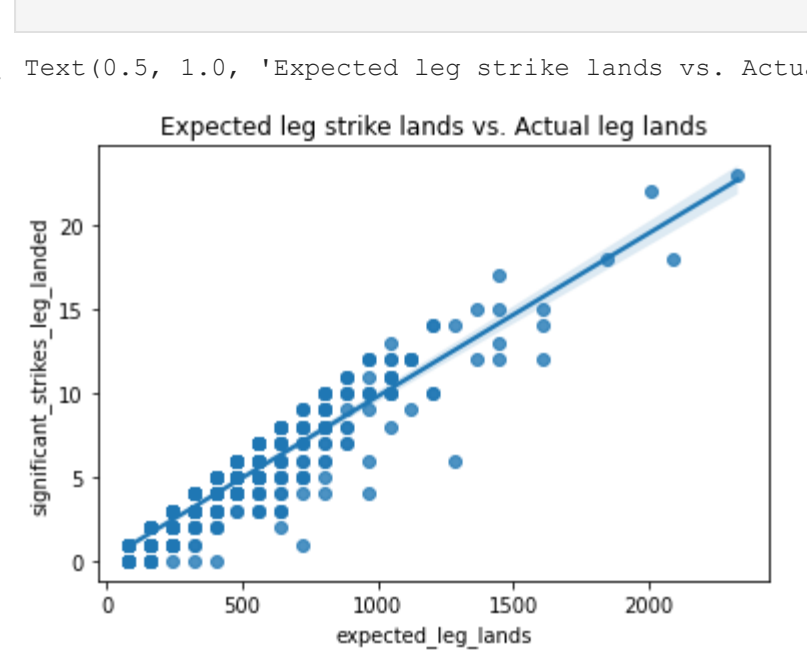
```
32.26032561486461
65.63404771975117
80.32315830824356
```

Out[100…

	significant_strikes_head_landed	significant_strikes_head_attempted	significant_strikes_body_landed	significant_strikes_body_attempted	signific
0	12	47	1	4	
1	16	62	6	7	
2	28	78	3	4	
3	18	43	9	13	
4	3	14	4	4	
5	37	67	2	4	
6	9	19	2	2	
7	10	17	3	6	
8	23	43	4	4	
9	21	55	3	3	
10	16	55	2	3	
11	3	31	1	1	
12	4	11	1	3	
13	9	62	4	6	
14	17	38	1	1	
15	16	75	2	3	
16	18	29	1	1	
17	7	25	2	6	
18	4	25	0	1	
19	3	24	2	2	

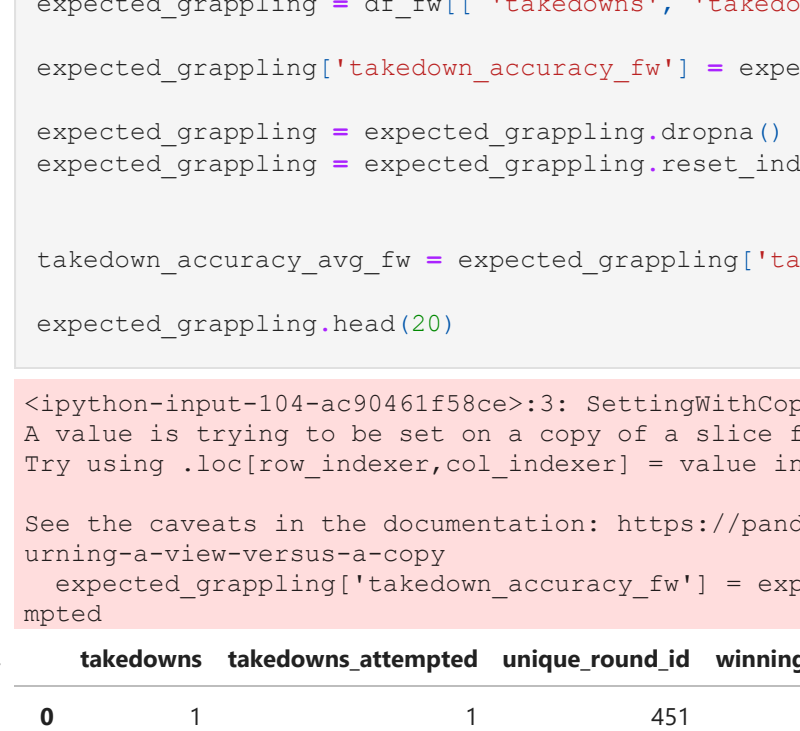
In [101…
```
sns.regplot(data=expected_by_target, y='significant_strikes_head_landed', x='expected_head_lands')
plt.title('Expected Head strike lands vs. Actual Head lands')
```

Out[101… Text(0.5, 1.0, 'Expected Head strike lands vs. Actual Head lands')



In [102…
```
sns.regplot(data=expected_by_target, y='significant_strikes_body_landed', x='expected_body_lands')
plt.title('Expected Body strike lands vs. Actual Body lands')
```

Out[102… Text(0.5, 1.0, 'Expected Body strike lands vs. Actual Body lands')



In [103…
```
sns.regplot(data=expected_by_target, y='significant_strikes_leg_landed', x='expected_leg_lands')
plt.title('Expected leg strike lands vs. Actual leg lands')
```

Out[103… Text(0.5, 1.0, 'Expected leg strike lands vs. Actual leg lands')

In [104…
```
expected_grappling = df_fw[[ 'takedowns', 'takedowns_attempted','unique_round_id', 'winning_fighter_name','losi

expected_grappling['takedown_accuracy_fw'] = expected_grappling.takedowns / expected_grappling.takedowns_attemp

expected_grappling = expected_grappling.dropna()
expected_grappling = expected_grappling.reset_index(drop=True)


takedown_accuracy_avg_fw = expected_grappling['takedown_accuracy_fw'].mean()

expected_grappling.head(20)
```

```
&lt;ipython-input-104-ac90461f58ce&gt;:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  expected_grappling['takedown_accuracy_fw'] = expected_grappling.takedowns / expected_grappling.takedowns_atte
mpted
```

Out[104…

	takedowns	takedowns_attempted	unique_round_id	winning_fighter_name	losing_fighter_name	takedown_accuracy_fw
0	1	1	451	Thiago Tavares	Clay Guida	1.000000
1	0	3	454	Brian Ortega	Clay Guida	0.000000
2	0	3	456	Brian Ortega	Clay Guida	0.000000
3	0	3	465	Yair Rodriguez	Jeremy Stephens	0.000000
4	1	1	468	Yair Rodriguez	Jeremy Stephens	1.000000
5	2	3	470	Yair Rodriguez	Jeremy Stephens	0.666667
6	1	1	476	Renato Moicano	Jeremy Stephens	1.000000
7	0	1	477	Renato Moicano	Jeremy Stephens	0.000000
8	0	1	478	Renato Moicano	Jeremy Stephens	0.000000
9	1	3	480	Renato Moicano	Jeremy Stephens	0.333333
10	1	1	482	Cub Swanson	Jeremy Stephens	1.000000
11	1	1	484	Cub Swanson	Jeremy Stephens	1.000000
12	0	3	486	Cub Swanson	Jeremy Stephens	0.000000
13	0	3	488	Cub Swanson	Jeremy Stephens	0.000000
14	0	2	530	Max Holloway	Ricardo Lamas	0.000000
15	0	3	532	Max Holloway	Ricardo Lamas	0.000000
16	0	1	534	Max Holloway	Ricardo Lamas	0.000000
17	0	3	537	Mirsad Bektic	Ricardo Lamas	0.000000
18	1	1	538	Mirsad Bektic	Ricardo Lamas	1.000000
19	0	1	539	Mirsad Bektic	Ricardo Lamas	0.000000

In [105…
```
print(takedown_accuracy_avg_fw)
```

0.3954921830628046

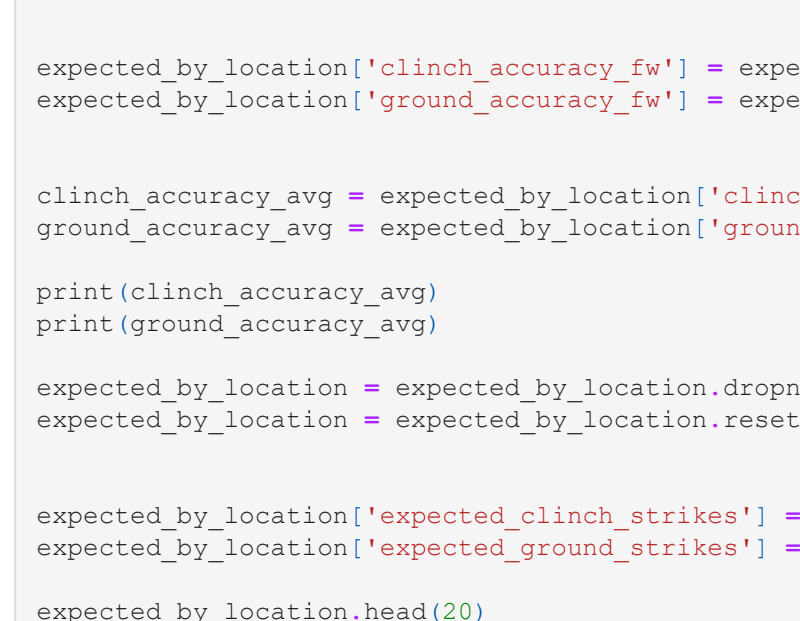
In [106…
```
expected_grappling['takedowns_attempted'] = expected_grappling['takedowns_attempted'].astype(int)
expected_grappling['expected_takedowns'] = expected_grappling.takedowns_attempted * takedown_accuracy_avg_fw

expected_grappling.head(20)
```

Out[106…

	takedowns	takedowns_attempted	unique_round_id	winning_fighter_name	losing_fighter_name	takedown_accuracy_fw	expected_takedowns
0	1	1	451	Thiago Tavares	Clay Guida	1.000000	0.395492
1	0	3	454	Brian Ortega	Clay Guida	0.000000	1.186477
2	0	3	456	Brian Ortega	Clay Guida	0.000000	1.186477
3	0	3	465	Yair Rodriguez	Jeremy Stephens	0.000000	1.186477
4	1	1	468	Yair Rodriguez	Jeremy Stephens	1.000000	0.395492
5	2	3	470	Yair Rodriguez	Jeremy Stephens	0.666667	1.186477
6	1	1	476	Renato Moicano	Jeremy Stephens	1.000000	0.395492
7	0	1	477	Renato Moicano	Jeremy Stephens	0.000000	0.395492
8	0	1	478	Renato Moicano	Jeremy Stephens	0.000000	0.395492
9	1	3	480	Renato Moicano	Jeremy Stephens	0.333333	1.186477
10	1	1	482	Cub Swanson	Jeremy Stephens	1.000000	0.395492
11	1	1	484	Cub Swanson	Jeremy Stephens	1.000000	0.395492
12	0	3	486	Cub Swanson	Jeremy Stephens	0.000000	1.186477
13	0	3	488	Cub Swanson	Jeremy Stephens	0.000000	1.186477
14	0	2	530	Max Holloway	Ricardo Lamas	0.000000	0.790984
15	0	3	532	Max Holloway	Ricardo Lamas	0.000000	1.186477
16	0	1	534	Max Holloway	Ricardo Lamas	0.000000	0.395492
17	0	3	537	Mirsad Bektic	Ricardo Lamas	0.000000	1.186477
18	1	1	538	Mirsad Bektic	Ricardo Lamas	1.000000	0.395492
19	0	1	539	Mirsad Bektic	Ricardo Lamas	0.000000	0.395492

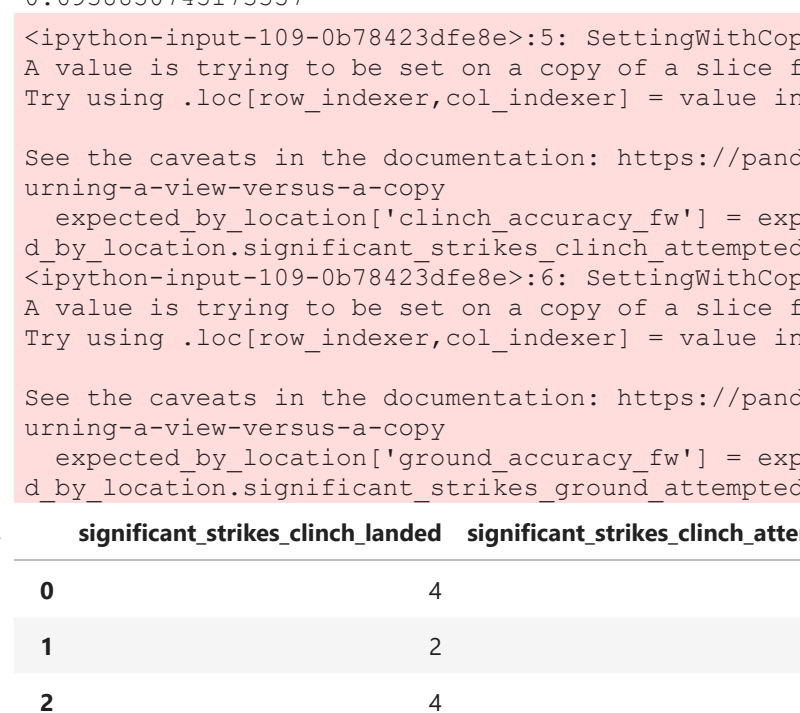
In [107…
```
sns.histplot(data=expected_grappling, x="takedowns_attempted", kde=True, element="poly")
```

Out[107… <AxesSubplot:xlabel='takedowns_attempted', ylabel='Count'>



In [108…
```
sns.regplot(data=expected_grappling, y='takedowns', x='expected_takedowns')
plt.title('Expected leg strike lands vs. Actual leg lands')
```

Out[108… Text(0.5, 1.0, 'Expected leg strike lands vs. Actual leg lands')



In [109…
```
expected_by_location = df_fw[[ 'significant_strikes_clinch_landed', 'significant_strikes_clinch_attempted', 'si



expected_by_location['clinch_accuracy_fw'] = expected_by_location.significant_strikes_clinch_landed / expected_
expected_by_location['ground_accuracy_fw'] = expected_by_location.significant_strikes_ground_landed / expected_


clinch_accuracy_avg = expected_by_location['clinch_accuracy_fw'].mean()
ground_accuracy_avg = expected_by_location['ground_accuracy_fw'].mean()

print(clinch_accuracy_avg)
print(ground_accuracy_avg)

expected_by_location = expected_by_location.dropna()
expected_by_location = expected_by_location.reset_index(drop=True)


expected_by_location['expected_clinch_strikes'] = expected_by_location.significant_strikes_clinch_attempted * c
expected_by_location['expected_ground_strikes'] = expected_by_location.significant_strikes_ground_attempted * g

expected_by_location.head(20)
```

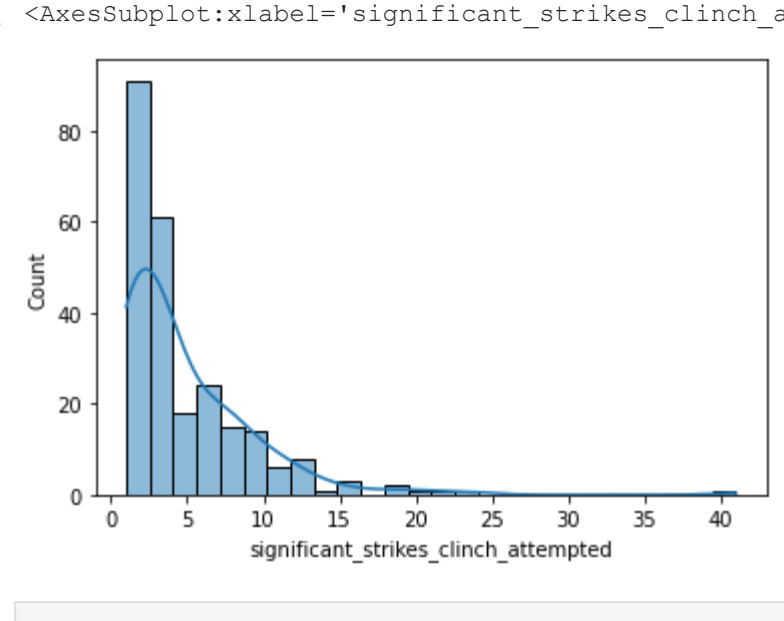
```
0.6447571081990285
0.6938850745173357
&lt;ipython-input-109-0b78423dfe8e&gt;:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  expected_by_location['clinch_accuracy_fw'] = expected_by_location.significant_strikes_clinch_landed / expecte
d_by_location.significant_strikes_clinch_attempted
&lt;ipython-input-109-0b78423dfe8e&gt;:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  expected_by_location['ground_accuracy_fw'] = expected_by_location.significant_strikes_ground_landed / expecte
d_by_location.significant_strikes_ground_attempted
```

Out[109…

	significant_strikes_clinch_landed	significant_strikes_clinch_attempted	significant_strikes_ground_landed	significant_strikes_ground_attempted	u
0	4	8	1	1	
1	2	2	34	58	
2	4	7	6	7	
3	0	1	19	26	
4	1	1	12	21	
5	2	3	10	11	
6	5	7	17	18	
7	8	11	1	3	
8	4	6	2	3	
9	1	1	0	1	
10	1	2	8	8	
11	0	1	25	37	
12	3	4	0	2	
13	2	4	8	12	
14	2	5	1	4	
15	3	5	1	1	
16	1	4	8	10	
17	1	4	3	7	
18	0	1	2	2	
19	0	1	1	1	

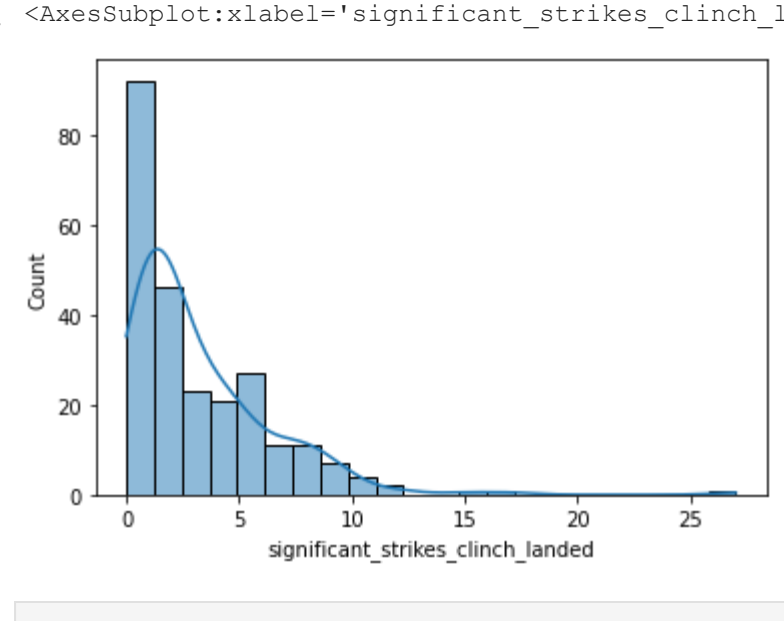
In [110…
```
sns.histplot(data=expected_by_location, x="clinch_accuracy_fw", kde=True)
```

Out[110… <AxesSubplot:xlabel='clinch_accuracy_fw', ylabel='Count'>



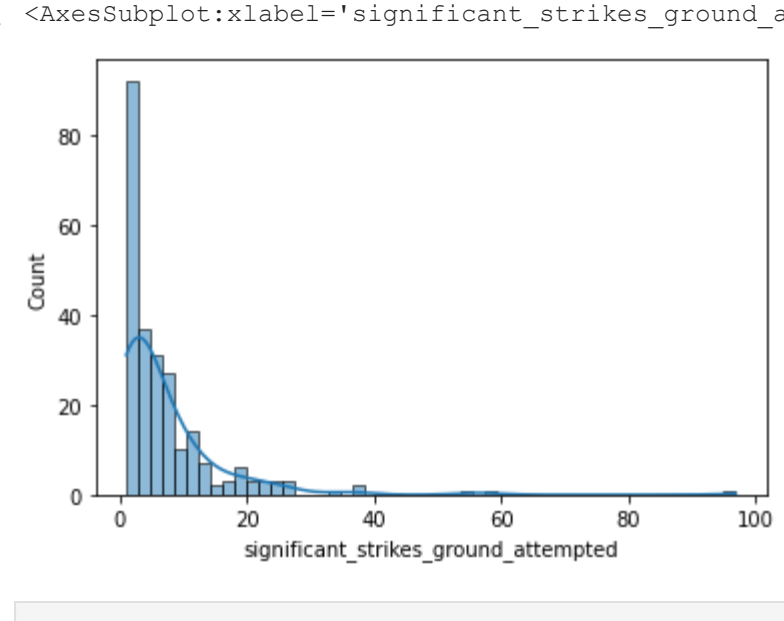
In [111…
```
sns.histplot(data=expected_by_location, x="ground_accuracy_fw", kde=True)
```

Out[111… <AxesSubplot:xlabel='ground_accuracy_fw', ylabel='Count'>



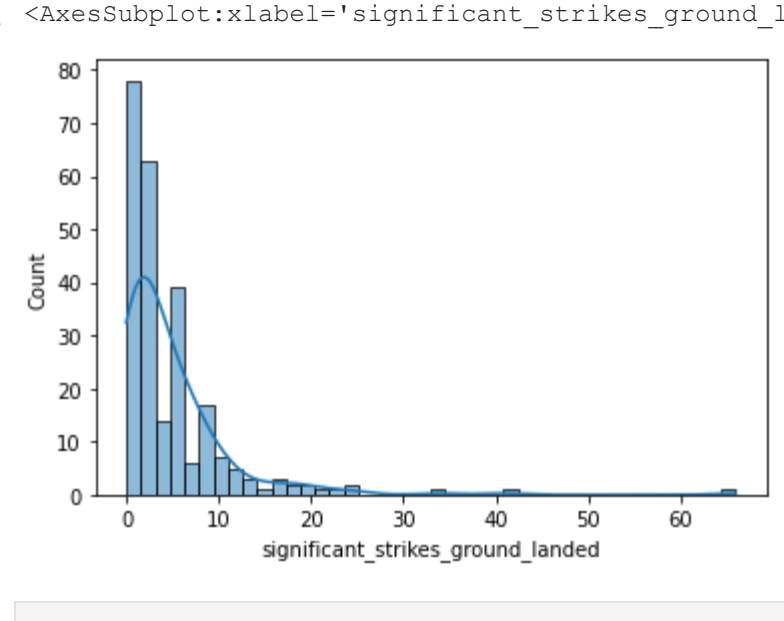
In [112…
```
sns.histplot(data=expected_by_location, x="significant_strikes_clinch_attempted", kde=True)
```

Out[112… <AxesSubplot:xlabel='significant_strikes_clinch_attempted', ylabel='Count'>



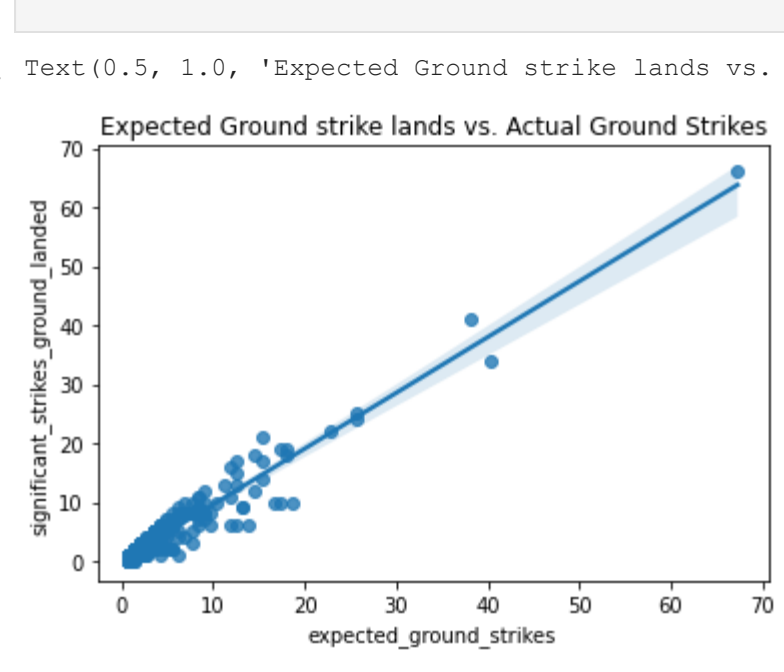
In [113…
```
sns.histplot(data=expected_by_location, x="significant_strikes_clinch_landed", kde=True)
```

Out[113… <AxesSubplot:xlabel='significant_strikes_clinch_landed', ylabel='Count'>



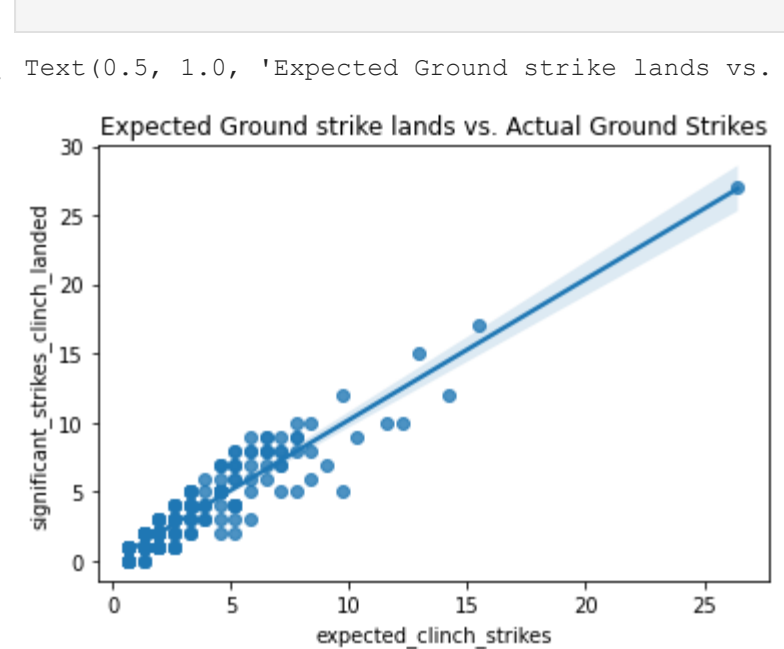
In [114…
```
sns.histplot(data=expected_by_location, x="significant_strikes_ground_attempted", kde=True)
```

Out[114… <AxesSubplot:xlabel='significant_strikes_ground_attempted', ylabel='Count'>



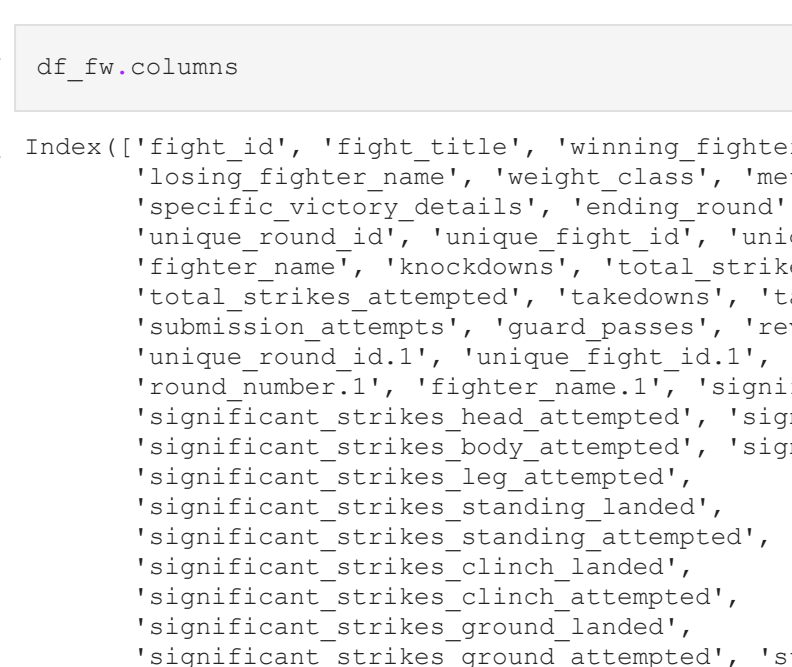
In [115…
```
sns.histplot(data=expected_by_location, x="significant_strikes_ground_landed", kde=True)
```

Out[115… <AxesSubplot:xlabel='significant_strikes_ground_landed', ylabel='Count'>



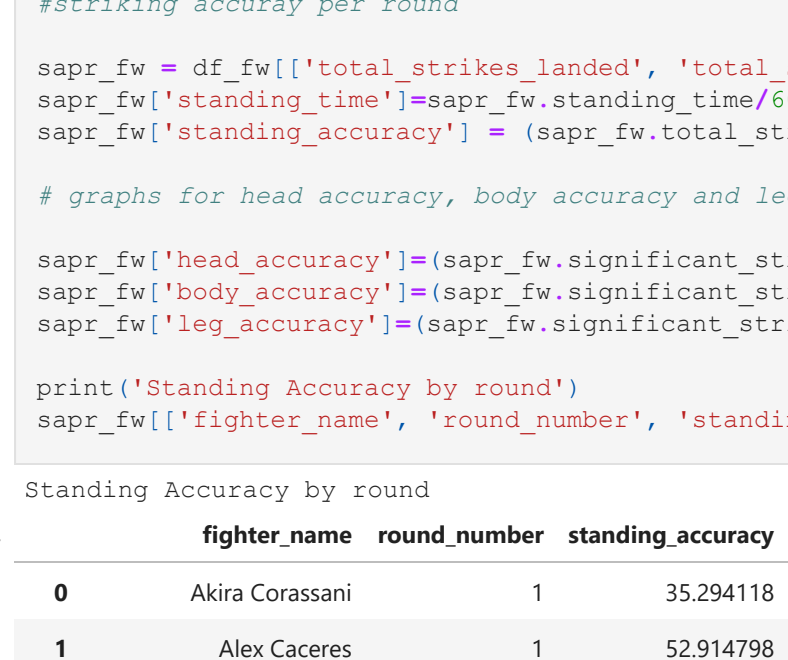
In [116…
```
sns.regplot(data=expected_by_location, y='significant_strikes_ground_landed', x='expected_ground_strikes')
plt.title('Expected Ground strike lands vs. Actual Ground Strikes')
```

Out[116… Text(0.5, 1.0, 'Expected Ground strike lands vs. Actual Ground Strikes')



In [117…
```
sns.regplot(data=expected_by_location, y='significant_strikes_clinch_landed', x='expected_clinch_strikes')
plt.title('Expected Ground strike lands vs. Actual Ground Strikes')
```

Out[117… Text(0.5, 1.0, 'Expected Ground strike lands vs. Actual Ground Strikes')



Examine by location of strikes i.e Standing (all done), Ground, and Clinch

In [118…
```
df_fw.columns
```

Out[118…
```
Index(['fight_id', 'fight_title', 'winning_fighter_name',
       'losing_fighter_name', 'weight_class', 'method_of_victory',
       'specific_victory_details', 'ending_round', 'ending_time', 'fight_url',
       'unique_round_id', 'unique_fight_id', 'unique_event_id', 'round_number',
       'fighter_name', 'knockdowns', 'total_strikes_landed',
       'total_strikes_attempted', 'takedowns', 'takedowns_attempted',
       'submission_attempts', 'guard_passes', 'reversals', 'control_time',
       'unique_round_id.1', 'unique_fight_id.1', 'unique_event_id.1',
       'round_number.1', 'fighter_name.1', 'significant_strikes_head_landed',
       'significant_strikes_head_attempted', 'significant_strikes_body_landed',
       'significant_strikes_body_attempted', 'significant_strikes_leg_landed',
       'significant_strikes_leg_attempted',
       'significant_strikes_standing_landed',
       'significant_strikes_standing_attempted',
       'significant_strikes_clinch_landed',
       'significant_strikes_clinch_attempted',
       'significant_strikes_ground_landed',
       'significant_strikes_ground_attempted', 'standing_time'],
      dtype='object')
```

In [119…
```
#striking accuray per round

sapr_fw = df_fw[['total_strikes_landed', 'total_strikes_attempted', 'fighter_name', 'round_number', 'significar
sapr_fw['standing_time']=sapr_fw.standing_time/60
sapr_fw['standing_accuracy'] = (sapr_fw.total_strikes_landed / sapr_fw.total_strikes_attempted) * 100

# graphs for head accuracy, body accuracy and leg accuracy by round

sapr_fw['head_accuracy']=(sapr_fw.significant_strikes_head_landed/sapr_fw.significant_strikes_head_attempted)*1
sapr_fw['body_accuracy']=(sapr_fw.significant_strikes_body_landed/sapr_fw.significant_strikes_body_attempted)*1
sapr_fw['leg_accuracy']=(sapr_fw.significant_strikes_leg_landed/sapr_fw.significant_strikes_leg_attempted)*100

print('Standing Accuracy by round')
sapr_fw[['fighter_name', 'round_number', 'standing_accuracy', 'head_accuracy', 'body_accuracy', 'leg_accuracy']
```

```
Standing Accuracy by round
```

Out[119…

	fighter_name	round_number	standing_accuracy	head_accuracy	body_accuracy	leg_accuracy
0	Akira Corassani	1	35.294118	25.000000	25.000000	80.000000
1	Alex Caceres	1	52.914798	39.490446	81.250000	86.363636
2	Alex Caceres	2	45.408163	36.486486	70.967742	75.000000
3	Alex Caceres	3	47.445255	40.740741	80.000000	75.000000
4	Alex Caceres	4	26.415094	24.489796	66.666667	0.000000
...	...	...	...	...	...	...
411	Yui Chul Nam	3	50.617284	21.951220	50.000000	80.000000
412	Zabit Magomedsharipov	1	53.669725	40.140845	60.000000	90.909091
413	Zabit Magomedsharipov	2	56.730769	43.511450	43.750000	100.000000
414	Zabit Magomedsharipov	3	47.191011	35.087719	28.571429	92.857143
415	Zubaira Tukhugov	1	75.675676	76.470588	100.000000	NaN

416 rows × 6 columns

In [120…
```
df_lw = pd.read_csv('lw_historical.csv', error_bad_lines=False)
df_lw.dropna()

df_lw = df_lw.loc[df_lw.control_time != '--']


df_lw['seconds'] = df_lw['control_time'].str.split(':',expand=True)[1]
df_lw['minutes'] = df_lw['control_time'].str.split(':',expand=True)[0]

df_lw['seconds'] = df_lw['seconds'].astype(dtype=int)
df_lw['minutes'] = df_lw['minutes'].astype(dtype=int)

df_lw['control_time'] = (df_lw['minutes'] * 60) + df_lw['seconds']
df_lw = df_lw.drop(columns=['seconds','minutes'])


df_lw['seconds'] = df_lw.ending_time.str.split(':',expand=True)[1]
df_lw['minutes'] = df_lw.ending_time.str.split(':',expand=True)[0]

df_lw['seconds'] = df_lw['seconds'].astype(int)
df_lw['minutes'] = df_lw['minutes'].astype(int)

df_lw['ending_time'] = (df_lw['minutes'] * 60) + df_lw['seconds']
df_lw = df_lw.drop(columns=['seconds','minutes'])

df_lw['standing_time'] = df_lw.ending_time - df_lw.control_time

df_lw.loc[df_lw.round_number != df_lw.ending_round,'ending_time'] = 300
print(df_lw.columns)
```

```
Index(['fight_id', 'fight_title', 'winning_fighter_name',
       'losing_fighter_name', 'weight_class', 'method_of_victory',
       'specific_victory_details', 'ending_round', 'ending_time', 'fight_url',
       'unique_round_id', 'unique_fight_id', 'unique_event_id', 'round_number',
       'fighter_name', 'knockdowns', 'total_strikes_landed',
       'total_strikes_attempted', 'takedowns', 'takedowns_attempted',
       'submission_attempts', 'guard_passes', 'reversals', 'control_time',
       'unique_round_id.1', 'unique_fight_id.1', 'unique_event_id.1',
       'round_number.1', 'fighter_name.1', 'significant_strikes_head_landed',
       'significant_strikes_head_attempted', 'significant_strikes_body_landed',
       'significant_strikes_body_attempted', 'significant_strikes_leg_landed',
       'significant_strikes_leg_attempted',
       'significant_strikes_standing_landed',
       'significant_strikes_standing_attempted',
       'significant_strikes_clinch_landed',
       'significant_strikes_clinch_attempted',
       'significant_strikes_ground_landed',
       'significant_strikes_ground_attempted', 'standing_time'],
      dtype='object')
```

In [121…
```
df_lw.head()
```

Out[121…

	fight_id	fight_title	winning_fighter_name	losing_fighter_name	weight_class	method_of_victory	specific_victory_details	ending_round	endin
0	2089	UFC 226: Miocic vs. Cormier	Anthony Pettis	Michael Chiesa	Lightweight	SUB	Triangle Choke	2	
1	2089	UFC 226: Miocic vs. Cormier	Anthony Pettis	Michael Chiesa	Lightweight	SUB	Triangle Choke	2	
2	2089	UFC 226: Miocic vs. Cormier	Anthony Pettis	Michael Chiesa	Lightweight	SUB	Triangle Choke	2	
3	2089	UFC 226: Miocic vs. Cormier	Anthony Pettis	Michael Chiesa	Lightweight	SUB	Triangle Choke	2	
4	2334	UFC Fight Night: Souza vs Mousasi	Joe Lauzon	Michael Chiesa	Lightweight	KO/TKO	NaN	2	

In [122…
```
df_lw.shape
```

Out[122… (960, 42)

In [123…

```
# creating the group frame

group_frame_lw = df_lw[['fighter_name', 'significant_strikes_standing_attempted', 'significant_strikes_standing
group_frame_lw.head(30)
```

Out[123…

	fighter_name	significant_strikes_standing_attempted	significant_strikes_standing_landed	significant_strikes_head_attempted	significant_strike
0	Aaron Riley	49	11	35	
1	Abel Trujillo	62	12	75	
2	Adriano Martins	14	10	17	
3	Akbarh Arreola	87	37	84	
4	Al Iaquinta	1309	530	1102	
5	Alan Patrick	121	32	103	
6	Alex Oliveira	138	50	162	
7	Alex White	214	74	233	
8	Alexander Hernandez	183	63	122	
9	Alexander Yakovlev	87	27	93	
10	Andrew Holbrook	31	12	36	
11	Anthony Njokuani	159	58	112	
12	Anthony Pettis	464	198	397	
13	Anthony Rocco Martin	118	39	123	
14	BJ Penn	38	22	66	
15	Beneil Dariush	593	256	615	
16	Benson Henderson	160	88	79	
17	Billy Evangelista	205	51	200	
18	Bob Cook	34	15	30	
19	Bobby Green	721	311	549	
20	Brad Riddell	45	20	49	
21	Bryan Barberena	173	72	122	
22	Chad Laprise	180	86	144	
23	Charles Oliveira	85	49	67	
24	Charlie Brenneman	8	3	4	
25	Chris Gruetzemacher	89	39	69	
26	Clay Guida	60	21	74	
27	Colton Smith	1	0	1	
28	Curt Warburton	53	24	36	
29	Damien Brown	124	31	103	

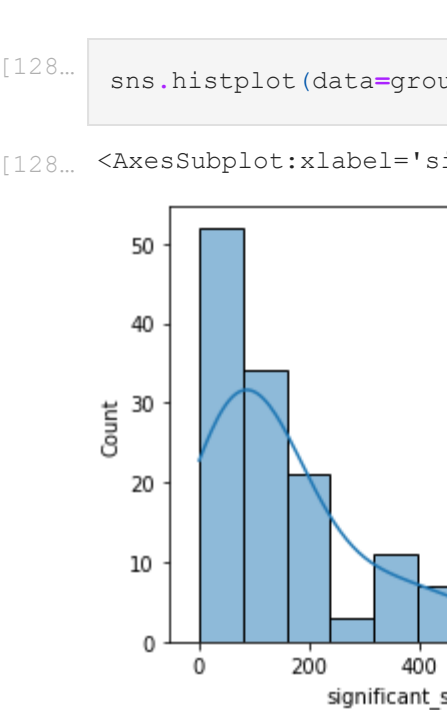
In [124…

```
group_frame_lw['head_ratio'] = group_frame_lw.significant_strikes_head_attempted / (group_frame_lw.significant_
group_frame_lw['body_ratio'] = group_frame_lw.significant_strikes_body_attempted / (group_frame_lw.significant_
group_frame_lw['leg_ratio'] = group_frame_lw.significant_strikes_leg_attempted / (group_frame_lw.significant_st
```

In [125…

```
sns.histplot(data=group_frame_lw, x="significant_strikes_standing_attempted", kde=True)
#overall career numbers
```

Out[125… <AxesSubplot:xlabel='significant_strikes_standing_attempted', ylabel='Count'>



In [126…

```
group_frame_lw.loc[ group_frame_lw.significant_strikes_standing_attempted == group_frame_lw.significant_strikes
```

Out[126…

	fighter_name	significant_strikes_standing_attempted	significant_strikes_standing_landed	significant_strikes_head_attempted	significant_strikes_b
4	Al Iaquinta	1309	530	1102	

In [127…

```
group_frame_lw.loc[ group_frame_lw.significant_strikes_standing_attempted == group_frame_lw.significant_strikes
```

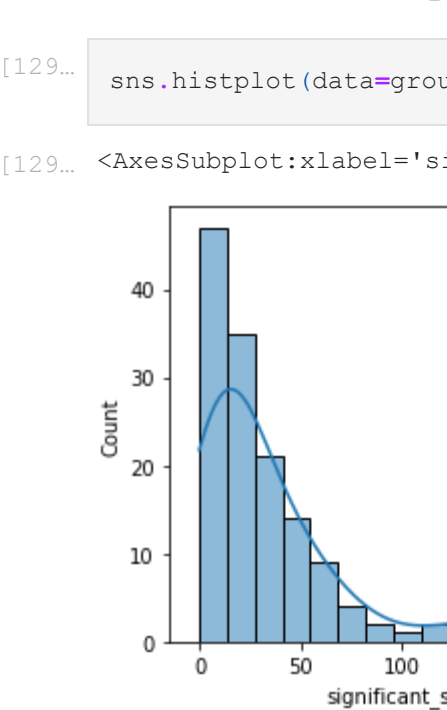
Out[127…

	fighter_name	significant_strikes_standing_attempted	significant_strikes_standing_landed	significant_strikes_head_attempted	significant_strikes_
27	Colton Smith	1	0	1	
66	Jens Pulver	1	0	1	

In [128…

```
sns.histplot(data=group_frame_lw, x="significant_strikes_head_attempted", kde=True)
```

Out[128… <AxesSubplot:xlabel='significant_strikes_head_attempted', ylabel='Count'>



In [129…

```
sns.histplot(data=group_frame_lw, x="significant_strikes_body_attempted", kde=True)
```

Out[129… <AxesSubplot:xlabel='significant_strikes_body_attempted', ylabel='Count'>

In [130…

```
group_frame_lw.loc[ group_frame_lw.significant_strikes_body_attempted == group_frame_lw.significant_strikes_boc
```

Out[130…

	fighter_name	significant_strikes_standing_attempted	significant_strikes_standing_landed	significant_strikes_head_attempted	significant_strike
102	Michael Johnson	1191	450	832	

In [131…

```
ratio_frame_lw = group_frame_lw[['fighter_name', 'head_ratio', 'body_ratio', 'leg_ratio' ]]
print('Mix Strikes')
ratio_frame_lw.head(148)
```

```
Mix Strikes
```

Out[131…

	fighter_name	head_ratio	body_ratio	leg_ratio
0	Aaron Riley	0.714286	0.142857	0.142857
1	Abel Trujillo	0.949367	0.050633	0.000000
2	Adriano Martins	0.772727	0.136364	0.090909
3	Akbarh Arreola	0.730435	0.130435	0.139130
4	Al Iaquinta	0.803793	0.125456	0.070751
...	...	...	...	...
139	Tyson Griffin	0.781609	0.183908	0.034483
140	Valmir Lazaro	0.830709	0.118110	0.051181
141	Vinc Pichel	0.601770	0.159292	0.238938
142	Will Brooks	0.660000	0.253333	0.086667
143	Yancy Medeiros	0.848780	0.087805	0.063415

144 rows × 4 columns

In [132…

```
print('Average Strike Mix ratios of LW')
ratio_frame_lw.mean()
```

Out[132…

```
Average Strike Mix ratios of LW
head_ratio    0.761955
body_ratio    0.138499
leg_ratio     0.099546
dtype: float64
```

In [133…

```
#STRIKE LOCATION
location_ratio_lw = group_frame_lw[['fighter_name', 'significant_strikes_standing_attempted', 'significant_stri
```

In [134…

```
location_ratio_lw['standing_ratio'] = location_ratio_lw.significant_strikes_standing_attempted / (location_rati
location_ratio_lw['ground_ratio'] = location_ratio_lw.significant_strikes_ground_attempted / (location_ratio_lw
location_ratio_lw['clinch_ratio'] = location_ratio_lw.significant_strikes_clinch_attempted / (location_ratio_lw

print('Strike Location')
location_ratio_lw[['fighter_name', 'standing_ratio', 'ground_ratio', 'clinch_ratio']]
```

```
Strike Location
```

Out[134…

	fighter_name	standing_ratio	ground_ratio	clinch_ratio
0	Aaron Riley	1.000000	0.000000	0.000000
1	Abel Trujillo	0.784810	0.000000	0.215190
2	Adriano Martins	0.636364	0.363636	0.000000
3	Akbarh Arreola	0.756522	0.226087	0.017391
4	Al Iaquinta	0.954778	0.020423	0.024799
...	...	...	...	...
139	Tyson Griffin	0.666667	0.218391	0.114943
140	Valmir Lazaro	0.960630	0.000000	0.039370
141	Vinc Pichel	0.504425	0.221239	0.274336
142	Will Brooks	0.706667	0.040000	0.253333
143	Yancy Medeiros	0.985366	0.004878	0.009756

144 rows × 4 columns

In [135…

```
print('Average Lw Fight locations')
#advise should probably drop some fighters as they have extreme values
location_ratio_lw[['fighter_name', 'standing_ratio', 'ground_ratio', 'clinch_ratio']].mean()
```

Out[135…

```
Average Lw Fight locations
standing_ratio    0.819711
ground_ratio      0.087456
clinch_ratio      0.092832
dtype: float64
```

In [136…

```
expected_by_target_lw = df_lw[[ 'significant_strikes_head_landed', 'significant_strikes_head_attempted', 'signi

expected_by_target_lw['head_accuracy_lw'] = expected_by_target_lw.significant_strikes_head_landed / expected_by
expected_by_target_lw['body_accuracy_lw'] = expected_by_target_lw.significant_strikes_body_landed / expected_by
expected_by_target_lw['leg_accuracy_lw'] = expected_by_target_lw.significant_strikes_leg_landed / expected_by_t




expected_by_target_lw = expected_by_target_lw.dropna()
expected_by_target_lw = expected_by_target_lw.reset_index(drop=True)

head_accuracy_avg_lw = expected_by_target_lw['head_accuracy_lw'].mean()
body_accuracy_avg_lw = expected_by_target_lw['body_accuracy_lw'].mean()
leg_accuracy_avg_lw = expected_by_target_lw['leg_accuracy_lw'].mean()

print(head_accuracy_avg_lw)
print(body_accuracy_avg_lw)
print(leg_accuracy_avg_lw)


expected_by_target_lw.head(20)
```

```
31.685994043427122
64.50435254752306
81.62854584049855
```

```
<ipython-input-136-78229188b346>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  expected_by_target_lw['head_accuracy_lw'] = expected_by_target_lw.significant_strikes_head_landed / expected_
by_target_lw.significant_strikes_head_attempted * 100
<ipython-input-136-78229188b346>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  expected_by_target_lw['body_accuracy_lw'] = expected_by_target_lw.significant_strikes_body_landed / expected_
by_target_lw.significant_strikes_body_attempted * 100
<ipython-input-136-78229188b346>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  expected_by_target_lw['leg_accuracy_lw'] = expected_by_target_lw.significant_strikes_leg_landed / expected_by
_target_lw.significant_strikes_leg_attempted * 100
```

Out[136…

	significant_strikes_head_landed	significant_strikes_head_attempted	significant_strikes_body_landed	significant_strikes_body_attempted	signific
0	4	17	6	6	
1	2	10	3	4	
2	8	23	6	7	
3	6	17	4	4	
4	9	38	2	2	
5	34	67	8	8	
6	1	12	2	2	
7	7	18	3	4	
8	9	15	1	2	
9	15	54	3	5	
10	9	46	14	19	
11	16	25	1	2	
12	7	30	0	1	
13	8	20	6	7	
14	5	18	3	3	
15	4	9	3	4	
16	7	21	2	3	
17	7	13	3	5	
18	11	30	8	10	
19	14	36	4	6	

In [137…

```
sns.histplot(data=expected_by_target_lw, x="significant_strikes_head_attempted", kde=True)
```

Out[137… <AxesSubplot:xlabel='significant_strikes_head_attempted', ylabel='Count'>

In [ ]: