UFC Data: EDA (EN)

Introduction

The Ultimate Fighting Championship (UFC) is an American mixed martial arts (MMA) promotion company based in Las Vegas, Nevada.

UFC has revolutionized the fight business and today stands as a premium global sports brand, media content company and the largest Pay-Per-View (PPV) event provider in the world. It is the largest MMA promotion in the world as of 2023.
In this project we'll be taking a look at the athlete's stats and determine whether there is any effect of one's skills on their professional record.

Data Source

The dataset is available on Kaggle at this link

Data Description

name: the name of the UFC athlete

nickname: the nickname of the UFC athlete

wins: the number of wins the athlete has in their career

losses: the number of losses the athlete has in their career

draws: the numaber of draws the athlete has in their career

height_cm: the height of the athlete in centimeters

weight_in_kg: the weight of the athlete in kilograms

reach_in_cm: the reach of the athlete in centimeters

stance: the fighting stance of the athlete

date_of_birth: the date of birth of the athlete

significant_strikes_landed_per_minute: the number of significant strikes the athlete lands per minute

significant_striking_accuracy: the significant striking accuracy coefficient

significant_strikes_absorbed_per_minute: the number of significant strikes the athlete absorbs per minute

significant_strike_defence: the coefficient showing how well the athlete defends significant striking

average_takedowns_landed_per_15_minutes: the number of takedowns the athlete manages to get per 15 minutes

takedown_accuracy: the coefficient showing how many takedowns of the athlete are successful

takedown_defense: the coefficient showing how well the athlete defends takedowns

average_submissions_attempted_per_15_minutes: the number of submissions the athlete attempts per 15 minutes

---

Preliminary Steps

First of all, we’ll modify the names of columns that are too long for the sake of convenience. We’ll also drop columns we won’t need.

▼ Code

```
df.rename(columns={subset_col[0]: 'sig_strikes_minute', subset_col[1]: 'sig_strike_accuracy', subset_col[2]: 'sig_strikes_absorb_min', subset_col[3]: 'sig_strike_defence', subset_col[4]: 'avg_takedowns_per_15min', subset_col[5]: 'avg_sub_per_15min'}, inplace=True)
```

▼ Code

```
df.drop(columns=['nickname', 'date_of_birth'], axis=1, inplace=True)
```

We’re going to be analysing athletes’ physical features so we have to drop rows with NaNs in columns **height_cm, weight_in_kg, reach_in_cm**.
As for the
stance column we'll replace NaNs with 'free'

▼ Code

```
subset_cols = ['stance']
[df[col].fillna('Free', inplace=True) for col in subset_cols]
```

▼ Code

```
df.dropna(subset=['height_cm','weight_in_kg','reach_in_cm'], inplace=True)
df.isna().sum()
```

Feature Engineering

Since we don't have weight division data in our dataset, we'll assign it manually.

This will obviuosly be an approximation and not full-proof but enough for us to perform analysis.

▼ Code

```
#creating a dictionary which we'll use to assign a weight category
weight_class = pd.DataFrame({'from': [0, 58, 62, 67, 71, 85, 94],
'to': [57.9, 61.9, 66.9, 70.9, 84.9, 93.9, 120.9],
'value': ['Flyweight', 'Bantamweight', 'Featherweight', 'Lightweight', 'Middleweight', 'Light Heavyweight', 'Heavyweight']})

#assigning the values
def assign_weight_class(x):
    return weight_class.loc[(x >= weight_class['from']) & (x <= weight_class['to']), 'value'].squeeze()

df['weight_class'] = df['weight_in_kg'].apply(assign_weight_class)
```

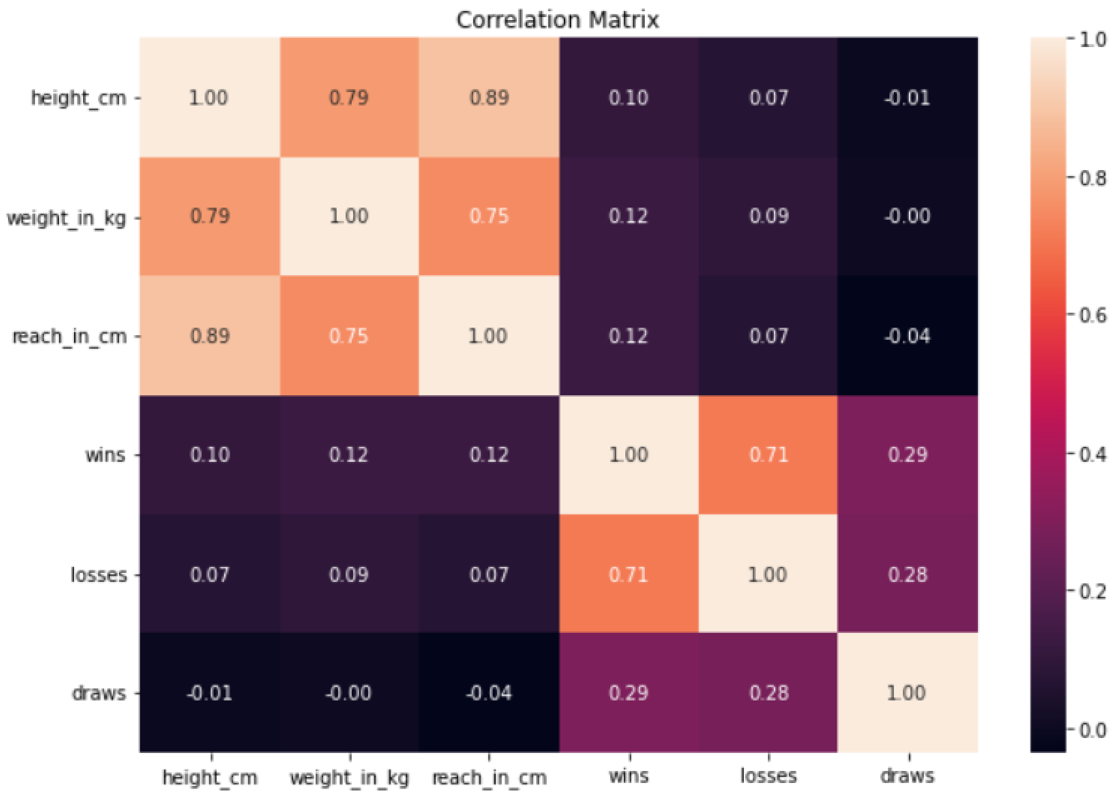
Correlation Analysis

We'll create 2 groups of characteristics and analyze the possible correlation between them:

1. Physical features: height, weight, reach.
2. Record: wins, losses, draws.

▼ Code

```
physical_features = ['height_cm', 'weight_in_kg', 'reach_in_cm']
record = ['wins', 'losses', 'draws']

corr_matrix = df[physical_features + record].corr()

plt.figure(figsize=(10,7))
sns.heatmap(data = corr_matrix,
           annot=True,
           cmap="rocket",
           fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```



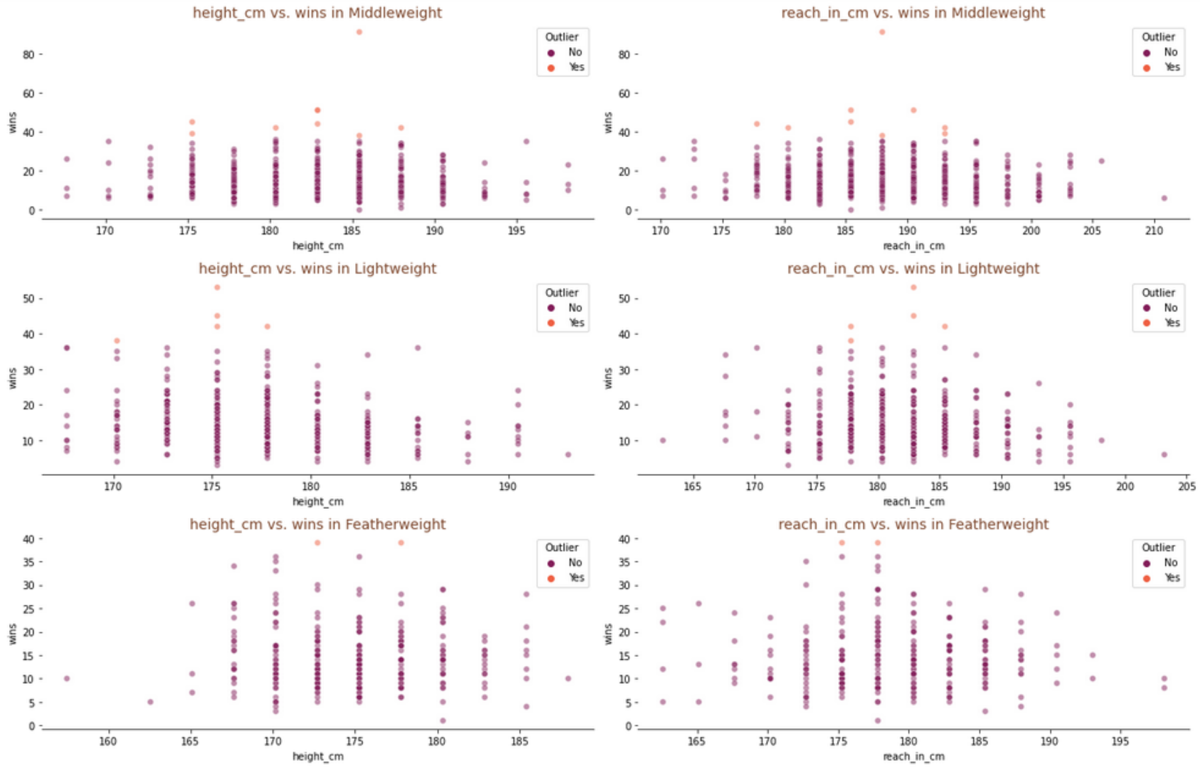
Correlation between physical features and record appears to be quite low judging by the correlation matrix.

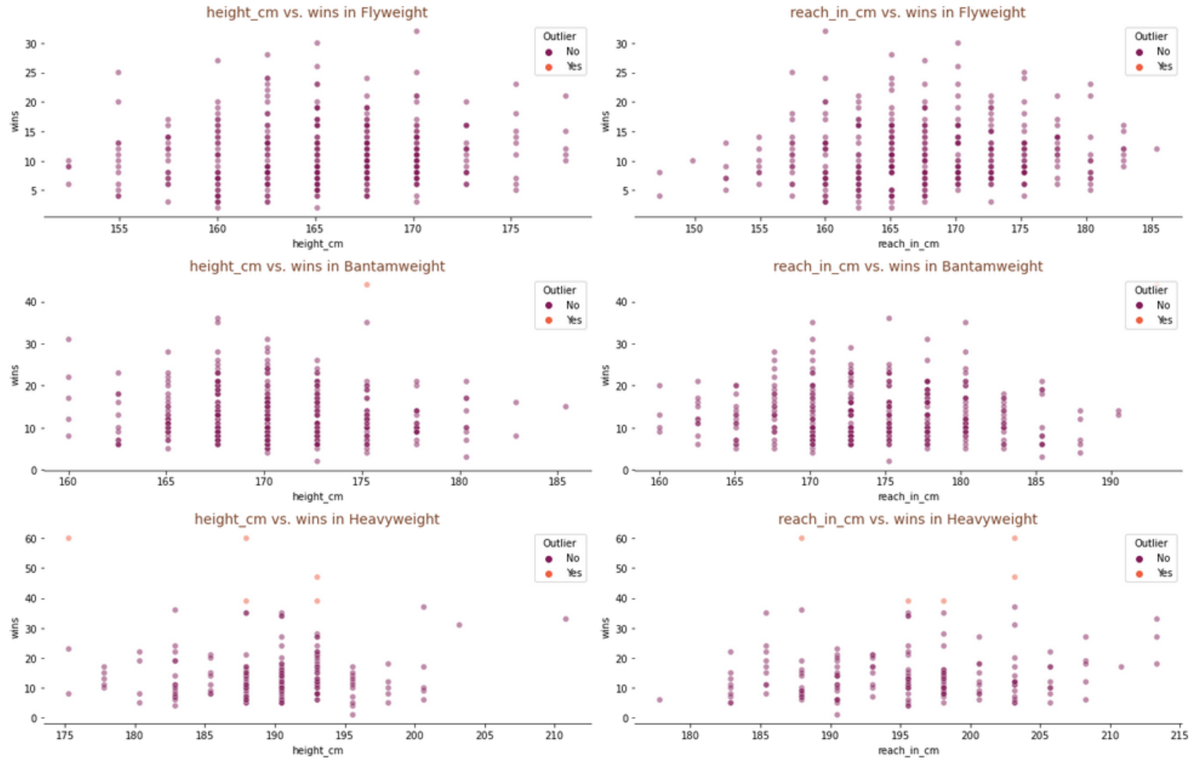
Interestingly enough, there is a strong positive correlation between losses and wins. In fact, there are only 70 UFC athletes who are undefeated

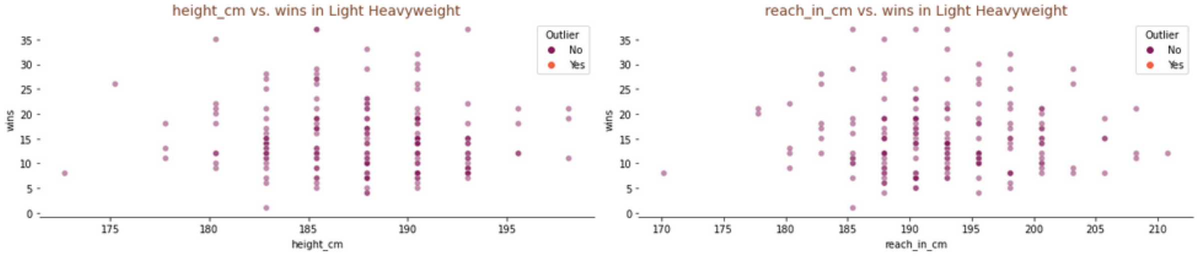
Analysing physical features vs wins by weight class

▼ Code

```
fig = plt.figure(figsize=(25, 25))
gs = GridSpec(ncols=3, nrows=7, figure=fig)

x=0
c=0
r=0
features = ['height_cm', 'reach_in_cm']    #not including weight since we wil
l use the weight_class criteria
wc = list(df['weight_class'].unique())

for weight in wc:
    for i, feature in enumerate(features):
        plt.subplot(gs[c,r])
        is_outlier = (df['wins'] >= df['wins'].mean() + df['wins'].std()*3)
        dt = df[df['weight_class'] == wc[x]]
        ax = sns.scatterplot(data = dt, x = df[df['weight_class'] == wc[x]][f
eature], y = df[df['weight_class'] == wc[x]]['wins'], hue = is_outlier, palet
te = 'rocket', alpha=0.5)
        handles, labels  =  ax.get_legend_handles_labels()
        ax.legend(handles, ['No', 'Yes'], loc='upper right', title = 'Outlier')
        plt.title(f'{feature} vs. wins in {weight}', color='#7A442A', fontsize=14)
        r+=1
    r=0
    x+=1
    c+=1

sns.despine(bottom=False, left=True)
plt.tight_layout()
plt.show()
```







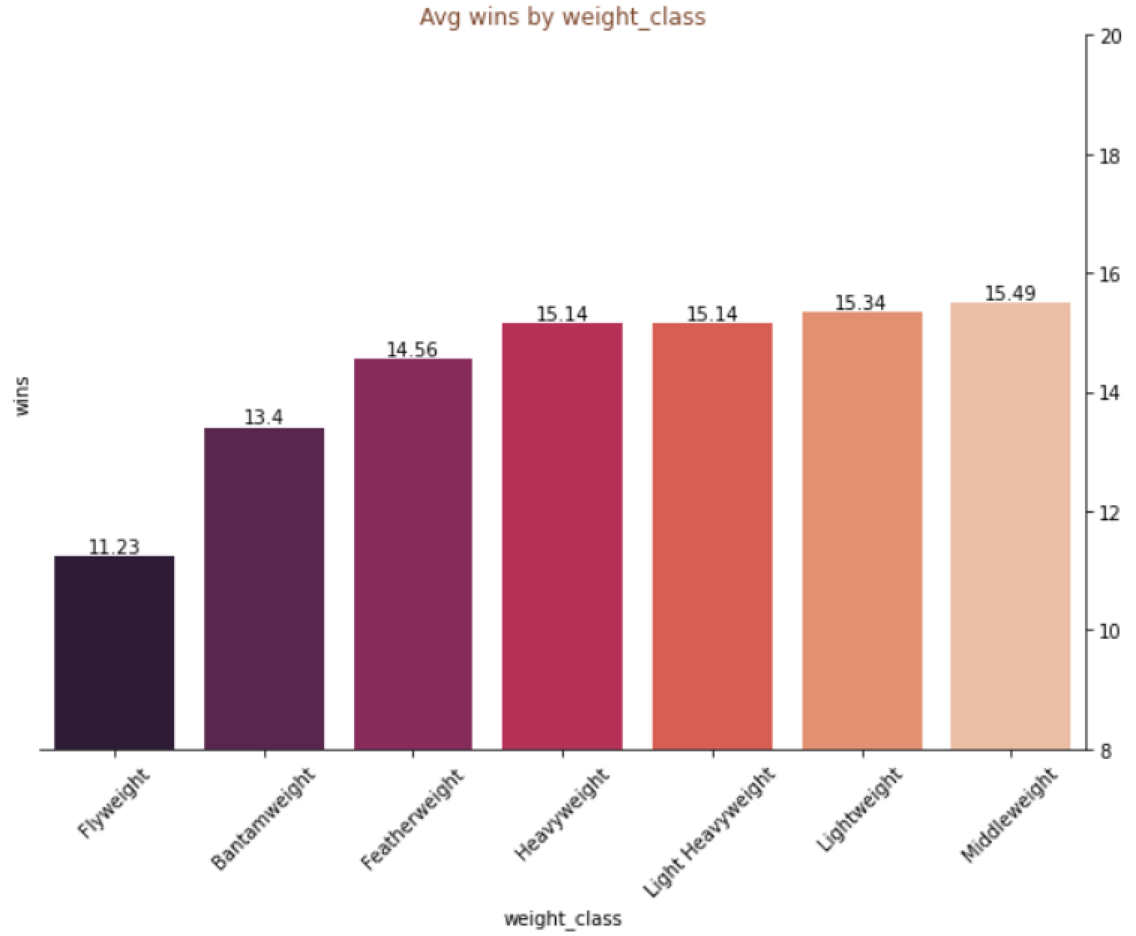
Fun Fact: the most significant outlier in Middleweight is **Jeremy Horn** with the astounding 91-22-5 record:

Weight category and number of wins

▼ Code

```
a = df.groupby('weight_class')['wins'].mean().round(2).reset_index()

fig = plt.figure(figsize=(10,7))

ax = sns.barplot(data = a, x = 'weight_class', y = 'wins', palette='rocket',
order=a.sort_values('wins').weight_class)
```

```
plt.xticks(rotation=45)

sns.despine(left=True, right=False)

ax.set(ylim=(8, 20))

for i in ax.containers:
    ax.bar_label(i,)

plt.title('Avg wins by weight_class', color = '#7A442A')
```



The Avg wins by weight_class above demonstrates that the average amount of wins is the highest in Middleweights and the lowest in Flyweights.

Let's dig deeper and see if there is actually any statistical significance to the difference we observe here.

We'll be using Welch’s t-Test to determine whether weight class influences the outcome of the fight because the variance of our two samples as well as sample sizes are not equal.

Our null hypothesis is H0: middleweight mean of wins = flyweight mean of wins

Our alternative hypothesis is H1: middleweight mean of wins > flyweight mean of wins

▼ Code

```
mw = df.query('weight_class == "Middleweight"')['wins']
fw = df.query('weight_class == "Flyweight"')['wins']

alpha = 0.05
t_crit, p_value = ttest_ind(mw, fw, equal_var=False)

if alpha <  p_value:
    print('Middleweight mean of wins is bigger \nT-statictic: {:.2f}\np-value: {:.2f}'.format(t_crit, p_value))
else:
    print('No significant statictic difference detected, the difference in samples is accidental\nT-statictic: {:.2f}, p-value: {:.2f}'.format(t_crit, p_value))
```

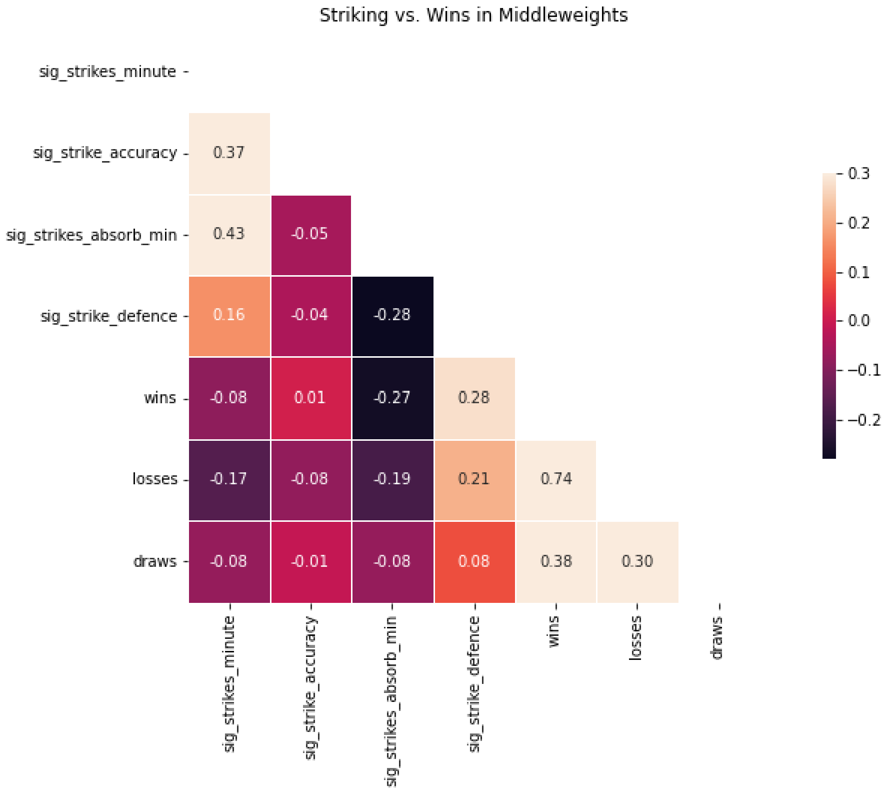
No significant statictic difference detected, the difference in samples is accidental.

Weight Category + Striking/Grappling vs Wins

Strong positive correlation (coef>0.3) detected between significant strikes defence and wins in Middleweights

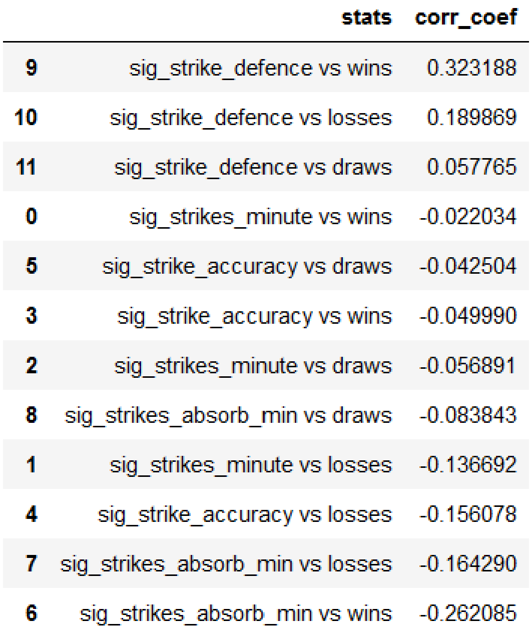
▼ Code

```
mw = df.query('weight_class=="Middleweight"')[striking_stats + record]

corr_matrix = mw[striking_stats + record].corr()

plt.figure(figsize = (15,7))

mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
sns.heatmap(corr_matrix, annot=True, fmt='.2f', mask=mask, cmap="rocket", vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.title('Striking vs. Wins in Middleweights')

plt.show()
```



▼ Code

```
striking = list(mw[striking_stats].columns)
rec = list(mw[record].columns)
striking_len = len(striking)
rec_len = len(rec)
l = []
coefs = []
striking_record_corr = {}
x=0

for i in range(striking_len):
    for j in range(rec_len):
        striking_rec = striking[i]+' vs '+rec[j]
        l.append(striking_rec)
        spearman_coef = spearmanr(mw[striking[i]], mw[record[j]]).correlation
        coefs.append(spearman_coef)

for i in l:
    striking_record_corr[l[x]] = coefs[x]
    x+=1

t = pd.DataFrame(striking_record_corr.items(), columns=['stats', 'corr_coe
f'])
pd.set_option('display.max_colwidth', None)
t.sort_values(by='corr_coef', ascending=False)
```

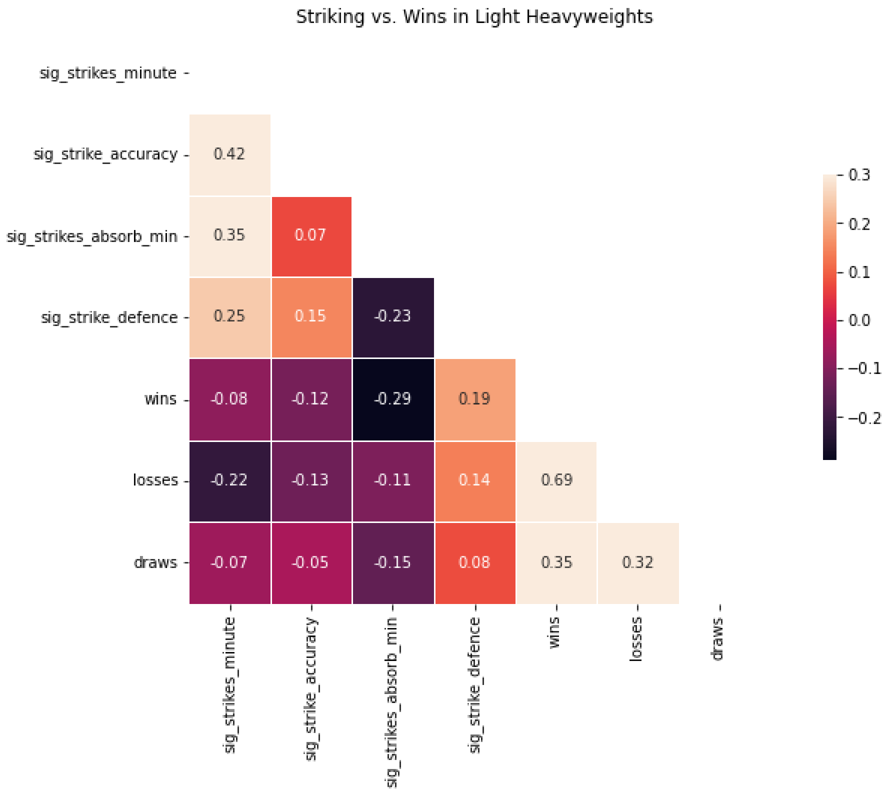


	stats	corr_coef
9	sig_strike_defence vs wins	0.323188
10	sig_strike_defence vs losses	0.189869
11	sig_strike_defence vs draws	0.057765
0	sig_strikes_minute vs wins	-0.022034
5	sig_strike_accuracy vs draws	-0.042504
3	sig_strike_accuracy vs wins	-0.049990
2	sig_strikes_minute vs draws	-0.056891
8	sig_strikes_absorb_min vs draws	-0.083843
1	sig_strikes_minute vs losses	-0.136692
4	sig_strike_accuracy vs losses	-0.156078
7	sig_strikes_absorb_min vs losses	-0.164290
6	sig_strikes_absorb_min vs wins	-0.262085

Strong negative correlation (coef>|0.3|) detected between significant strikes absorbed per minute and wins in Light Heavyweights.

▼ Code

```
lhw = df.query('weight_class=="Light Heavyweight"')[striking_stats + record]

corr_matrix = lhw[striking_stats + record].corr()

plt.figure(figsize = (15,7))

mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
sns.heatmap(corr_matrix, annot=True, fmt='.2f', mask=mask, cmap="rocket", vma
x=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.title('Striking vs. Wins in Light Heavyweights')

plt.show()
```



▼ Code

```
striking = list(lhw[striking_stats].columns)
rec = list(lhw[record].columns)
striking_len = len(striking)
rec_len = len(rec)
l = []
coefs = []
striking_record_corr = {}
x=0

for i in range(striking_len):
    for j in range(rec_len):
        striking_rec = striking[i]+' vs '+rec[j]
        l.append(striking_rec)
        spearman_coef = spearmanr(lhw[striking[i]], lhw[record[j]]).correlati
on
        coefs.append(spearman_coef)

for i in l:
    striking_record_corr[l[x]] = coefs[x]
    x+=1

t = pd.DataFrame(striking_record_corr.items(), columns=['stats', 'corr_coe
f'])
pd.set_option('display.max_colwidth', None)
t.sort_values(by='corr_coef', ascending=False)
```

	stats	corr_coef
9	sig_strike_defence vs wins	0.214402
10	sig_strike_defence vs losses	0.179109
11	sig_strike_defence vs draws	0.132196
2	sig_strikes_minute vs draws	0.001499
5	sig_strike_accuracy vs draws	-0.022034
0	sig_strikes_minute vs wins	-0.054499
7	sig_strikes_absorb_min vs losses	-0.130587
3	sig_strike_accuracy vs wins	-0.139284
8	sig_strikes_absorb_min vs draws	-0.166686
4	sig_strike_accuracy vs losses	-0.176550
1	sig_strikes_minute vs losses	-0.192814
6	sig_strikes_absorb_min vs wins	-0.315041

On average, middleweights exhibit a better level of grappling with the only exception being takedown defence.

▼ Code

```
df.groupby('weight_class')[grappling_stats].mean().round(2)
```

	avg_takedowns_per_15min	takedown_accuracy	takedown_defense	avg_sub_per_15min
weight_class				
Bantamweight	1.51	32.13	54.68	0.53
Featherweight	1.57	32.51	53.52	0.62
Flyweight	1.53	35.31	51.11	0.60
Heavyweight	1.49	33.54	54.01	0.49
Light Heavyweight	1.50	31.64	55.02	0.48
Lightweight	1.50	32.61	52.94	0.64
Middleweight	1.64	37.51	53.77	0.70

Conclusions

Having analyzed the dataset we can summarize everything we've identified:

- No particular stance type seems to be correlated with a bigger number of wins.
- Striking skills have a strong positive correlation with the number of wins in Middleweights.
- Strong negative correlation between significant strikes absorbed per minute and the number of wins exists in Light Heavyweights.
- On average, Middleweights exhibit a better level of grappling with the only exception being takedown defence - the light heavyweights take the lead here since.
- Featherweights exhibit positive correlation between an average submission attempts per 15 minutes and number of wins.
- Strong positive correlation identified in light heavyweights between an average of submission attempts per 15 minutes and the number of losses.

Links

Link to the full Jupyter Notebook

Contacts

lianazaripovar@gmail.com

tg: @zaripova_liana