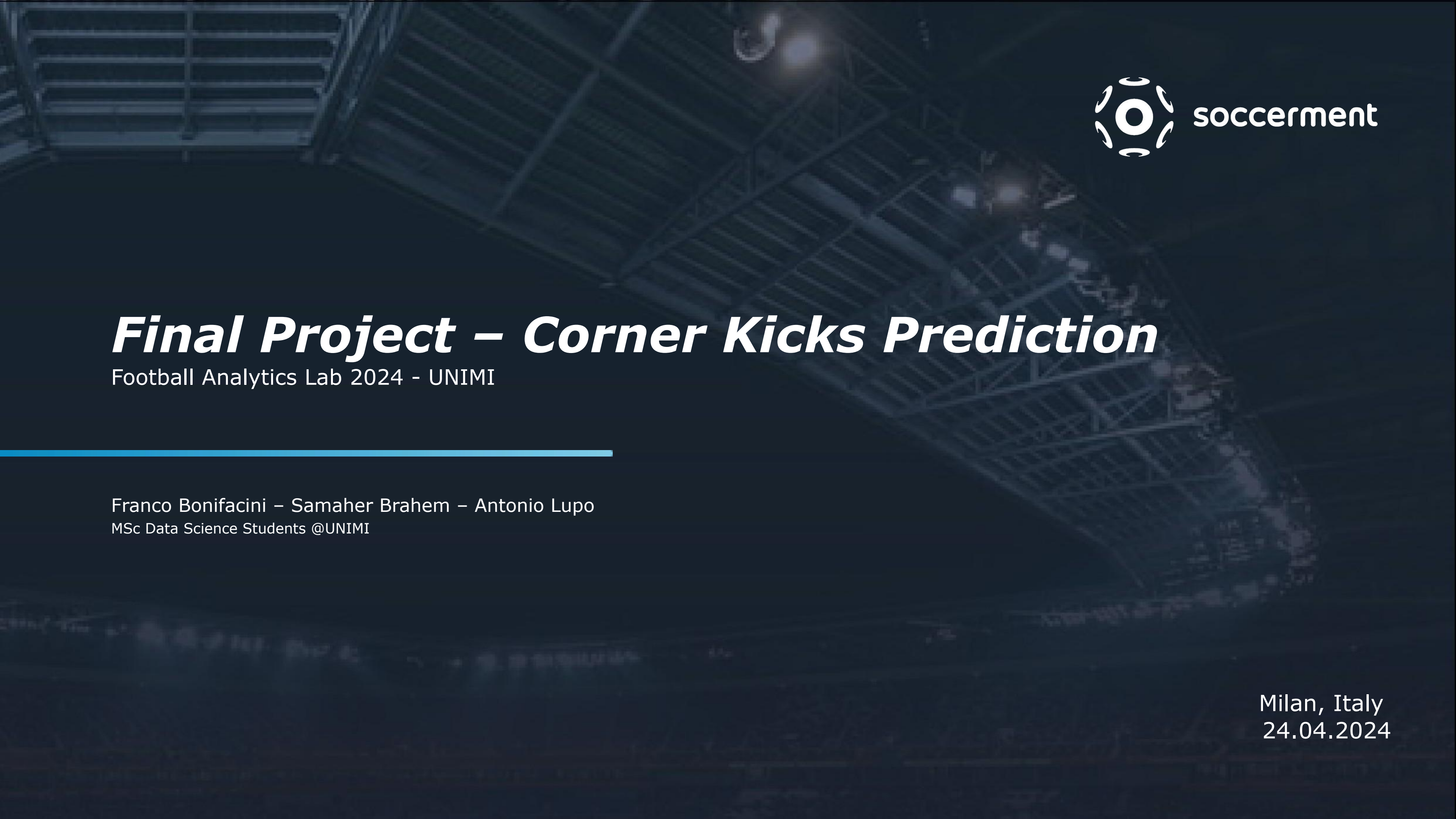


soccerment

Final Project – Corner Kicks Prediction

Football Analytics Lab 2024 - UNIMI

Franco Bonifacini – Samaher Brahem – Antonio Lupo
MSc Data Science Students @UNIMI

Milan, Italy
24.04.2024

Project Description

Goal:

Predict the total number of corner kicks in a match using team statistics and historical performance data.

Dataset:

Team performance statistics and style metrics up to the current match.
Includes historical data to provide context and trends.

Challenges:

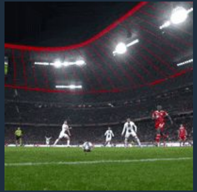
- Different play styles and strategies may significantly affect the number of corners, requiring adaptive modeling techniques.
- Low-frequency events like corner kicks in defensively strong or low-scoring games could skew predictions.
- Ensuring the model accounts for variations in tactical approaches between teams and across different leagues or seasons.

Data Understanding

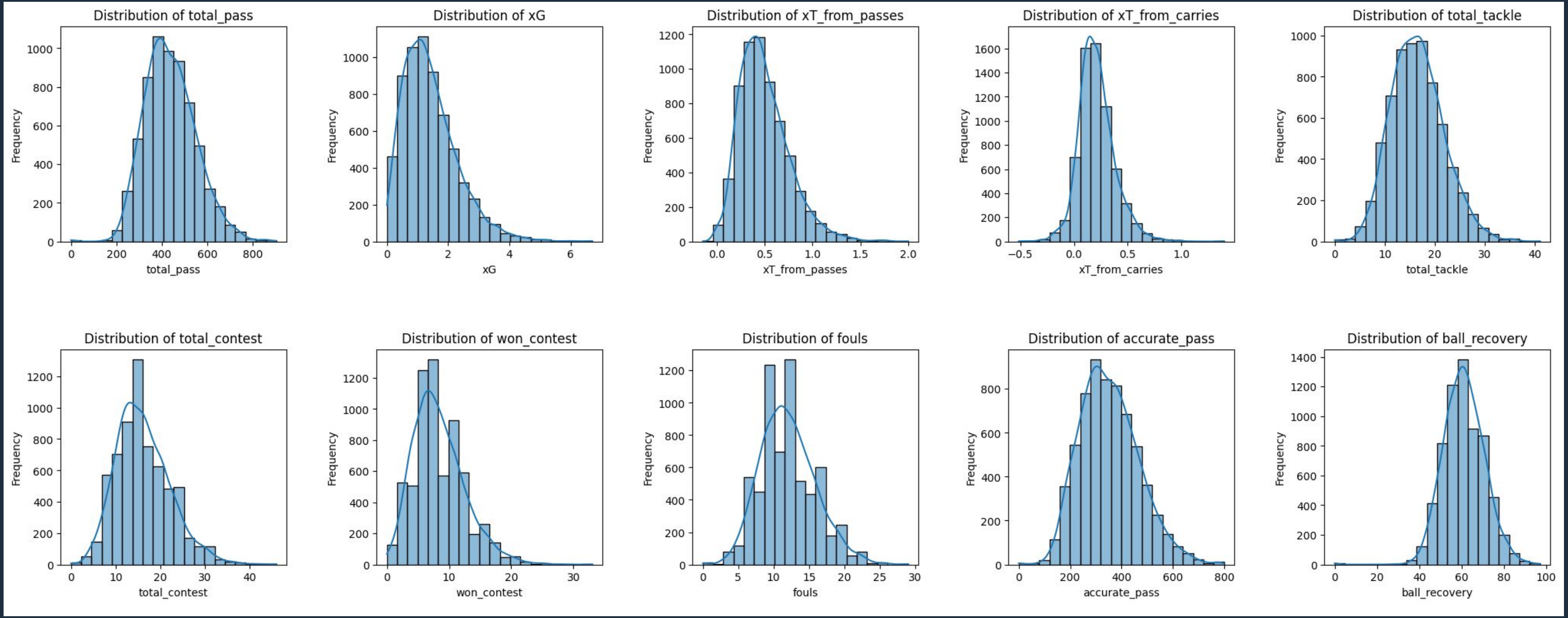
25 Variables 👉 4 Cat + 21 Num || No ✨ Null Values ✨ || s 2022 || diff leagues

team_id	xG	won_contest	aerial_won	lost_corners
game_id	xT_from_passes	accurate_pass	aerial_lost	goals
team	xT_from_carries	total_clearance	total_long_balls	match_day
season	total_tackle	goal_kicks	total_cross_nocorner	fouls
total_pass	total_contest	total_scoring_att	corner_taken ⭐	ball_recovery

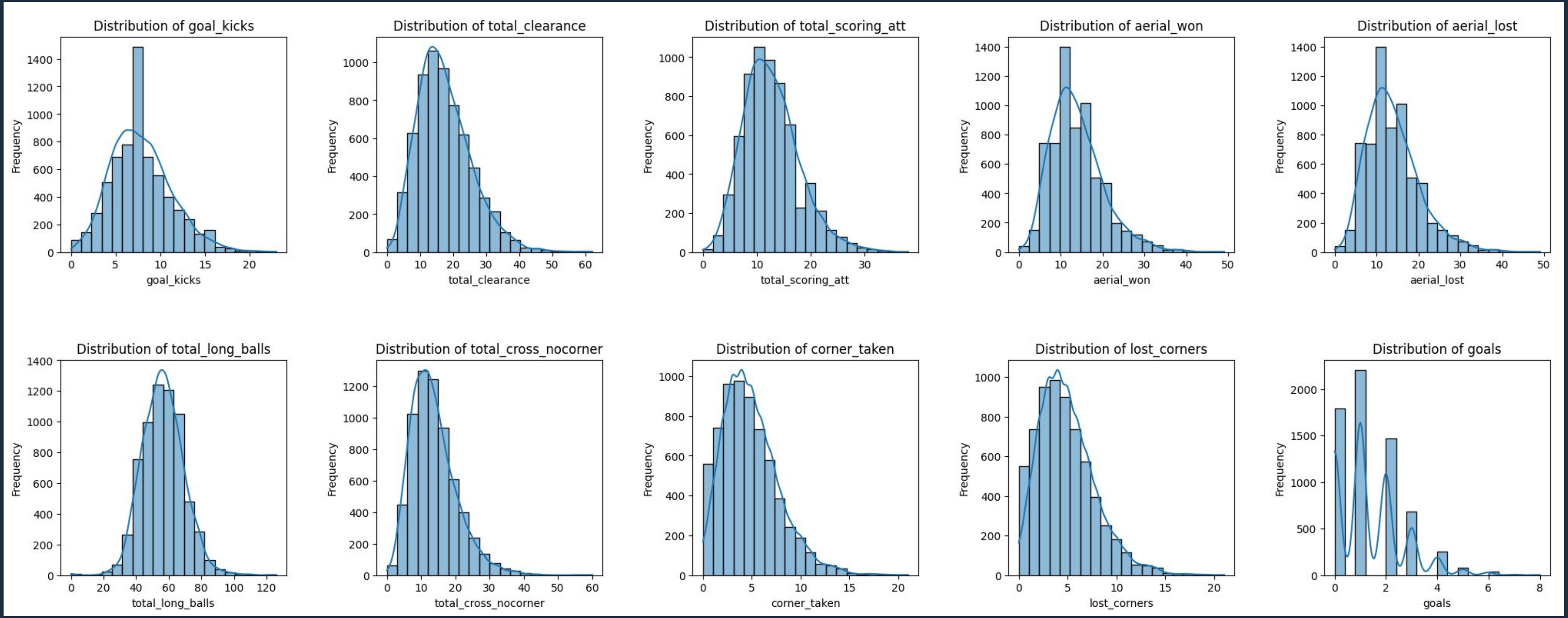
Data Understanding

Metric	Description	
total_contest	An attempted dribble past a player (contests always involve 2 players) - doesn’t include ‘overrun’ situations where the attacking player takes on an opponent but the ball runs away from them out of play or to an opponent	
total_clearance	A successful defensive clearance - where a player under pressure kicks the ball clear of the defensive zone or/and out of play	
total_cross_nocorner	Total number of crosses that are not from corners. A cross is a pass made from a wide position near the opponent's penalty area, aiming to deliver the ball into the penalty area.	

Data Understanding

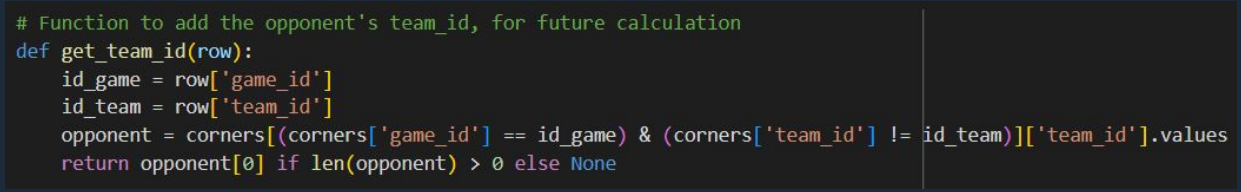


Data Understanding

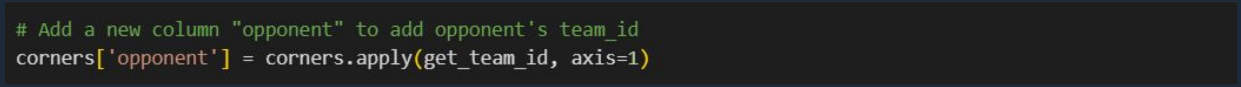


Data Preprocessing

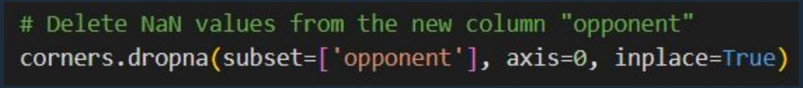
- Drop games with match_day equal to 0 — From 6.537 rows to 6.530
- Add opponent's team_id for future analysis — From 6.530 rows to 6.524



```python
# Function to add the opponent's team_id, for future calculation
def get_team_id(row):
    id_game = row['game_id']
    id_team = row['team_id']
    opponent = corners[(corners['game_id'] == id_game) & (corners['team_id'] != id_team)]['team_id'].values
    return opponent[0] if len(opponent) > 0 else None
```

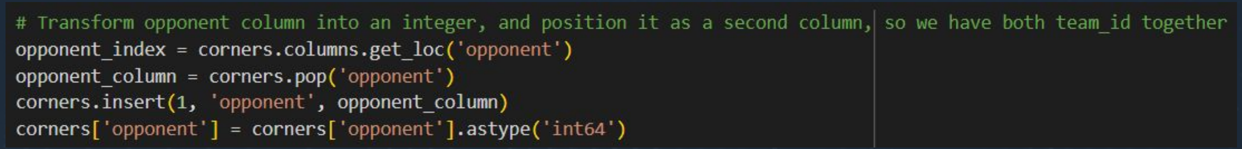


```python
# Add a new column "opponent" to add opponent's team_id
corners['opponent'] = corners.apply(get_team_id, axis=1)
```



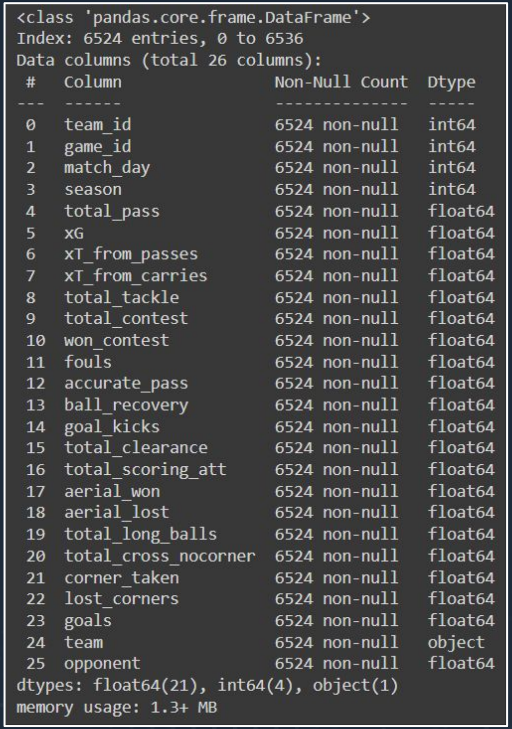
```python
# Delete NaN values from the new column "opponent"
corners.dropna(subset=['opponent'], axis=0, inplace=True)
```

6 matches with no opponent



```python
# Transform opponent column into an integer, and position it as a second column, so we have both team_id together
opponent_index = corners.columns.get_loc('opponent')
opponent_column = corners.pop('opponent')
corners.insert(1, 'opponent', opponent_column)
corners['opponent'] = corners['opponent'].astype('int64')
```

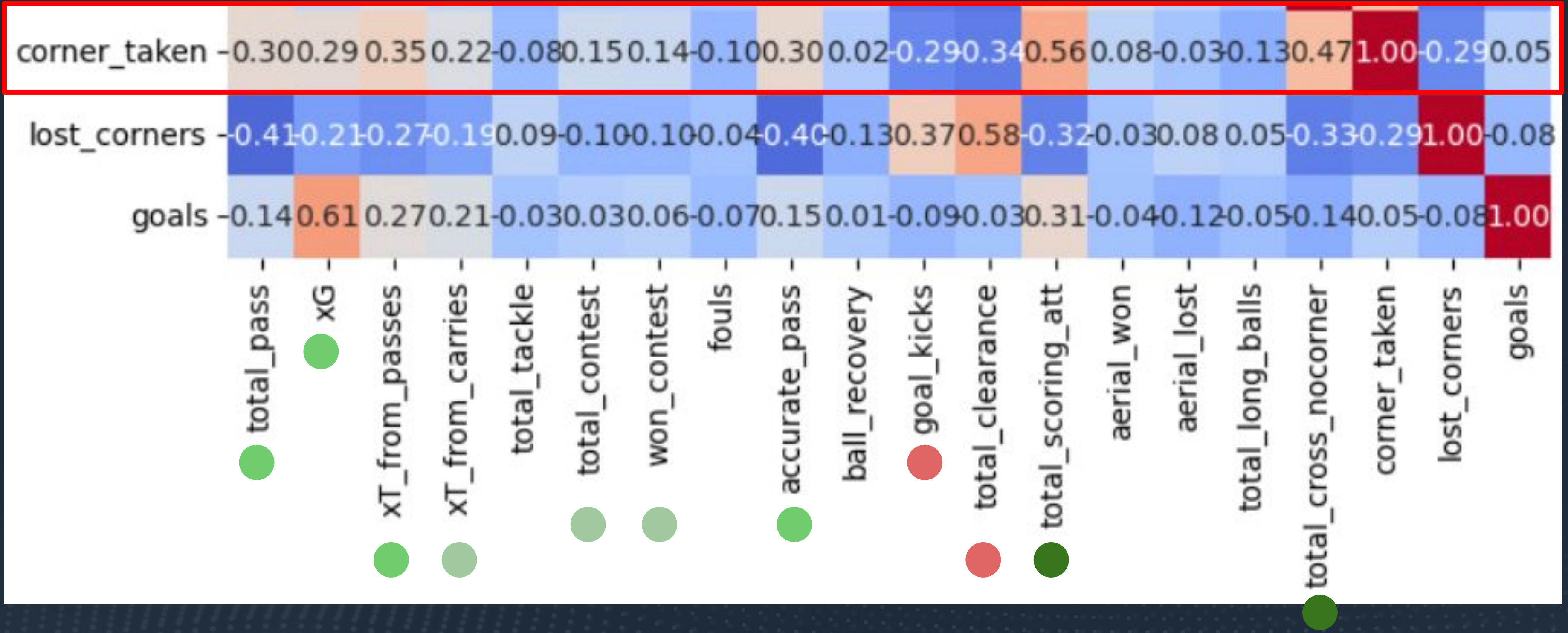
Final dataset



```
<class 'pandas.core.frame.DataFrame'>
Index: 6524 entries, 0 to 6536
Data columns (total 26 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   team_id               6524 non-null   int64
 1   game_id               6524 non-null   int64
 2   match_day             6524 non-null   int64
 3   season                6524 non-null   int64
 4   total_pass            6524 non-null   float64
 5   xG                    6524 non-null   float64
 6   xT_from_passes        6524 non-null   float64
 7   xT_from_carries       6524 non-null   float64
 8   total_tackle          6524 non-null   float64
 9   total_contest         6524 non-null   float64
 10  won_contest           6524 non-null   float64
 11  fouls                 6524 non-null   float64
 12  accurate_pass         6524 non-null   float64
 13  ball_recovery         6524 non-null   float64
 14  goal_kicks            6524 non-null   float64
 15  total_clearance       6524 non-null   float64
 16  total_scoring_att     6524 non-null   float64
 17  aerial_won            6524 non-null   float64
 18  aerial_lost           6524 non-null   float64
 19  total_long_balls      6524 non-null   float64
 20  total_cross_nocorner  6524 non-null   float64
 21  corner_taken          6524 non-null   float64
 22  lost_corners          6524 non-null   float64
 23  goals                 6524 non-null   float64
 24  team                  6524 non-null   object
 25  opponent              6524 non-null   float64
dtypes: float64(21), int64(4), object(1)
memory usage: 1.3+ MB
```

Data Preprocessing

Analysis of correlation between variables and corner_taken



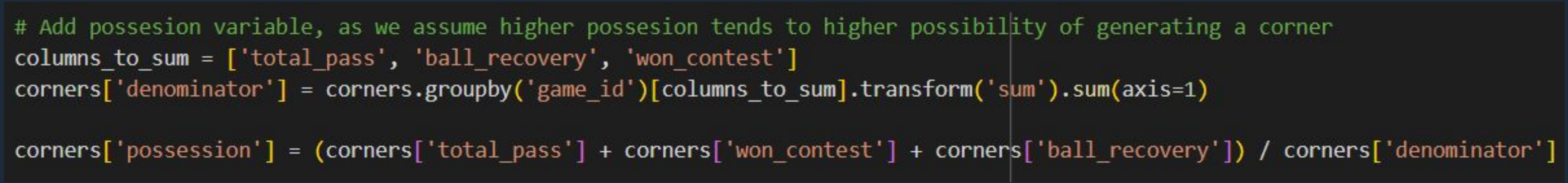
	total_pass	xG	xT_from_passes	xT_from_carries	total_tackle	total_contest	won_contest	fouls	accurate_pass	ball_recovery	goal_kicks	total_clearance	total_scoring_att	aerial_won	aerial_lost	total_long_balls	total_cross_nocorner	corner_taken	lost_corners	goals
corner_taken	0.30	0.29	0.35	0.22	-0.08	0.15	0.14	-0.10	0.30	0.02	-0.29	-0.34	0.56	0.08	-0.03	-0.13	0.47	1.00	-0.29	0.05
lost_corners	-0.41	-0.21	-0.27	-0.19	0.09	-0.10	-0.10	-0.04	-0.40	-0.13	0.37	0.58	-0.32	-0.03	0.08	0.05	-0.33	-0.29	1.00	-0.08
goals	0.14	0.61	0.27	0.21	-0.03	0.03	0.06	-0.07	0.15	0.01	-0.09	-0.03	0.31	-0.04	-0.12	-0.05	0.14	0.05	-0.08	1.00

Data Preprocessing

Creation of new variables:

Total match

★ **possession** = (total_pass+ball_recovery+won_contest) / (total_pass+ball_recovery+won_contest)



```
# Add possesion variable, as we assume higher possesion tends to higher possibility of generating a corner
columns_to_sum = ['total_pass', 'ball_recovery', 'won_contest']
corners['denominator'] = corners.groupby('game_id')[columns_to_sum].transform('sum').sum(axis=1)

corners['possession'] = (corners['total_pass'] + corners['won_contest'] + corners['ball_recovery']) / corners['denominator']
```

★ **pass_accuracy** = accurate_pass / total_pass

★ **won_contest_accuracy** = won_contest / total_contest

★ **outperformance** = goals / xG ✖ ***Not statistically significant***

And variable corner_taken?...

The Model

Intercept	4.795760
Goals	-0.093029
xG	-0.181465
xT from passes	0.320596
Total Contest	0.022270
Fouls	-0.027978
Ball Recovery	-0.032609
Total Clearance	-0.009877
Goal Kicks	-0.026598
Pass Accuracy	-6.022043
Dribbling Accuracy	-0.420872
Corners Conceded	-0.025124
Aerial Won	-0.008168
Total Long Balls	-0.005173
Total Shots	0.177025
Possession	2.900296
Total Crosses (No Corner)	0.080647
Average Corners (3 games)	0.698725

Feature Estimation

Simple Average

prev 7 games

Weighted Average

exponential moving average

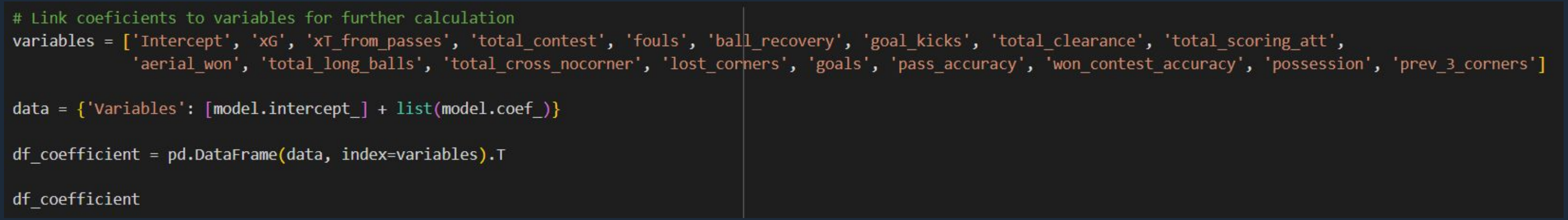
Adjusted Average

adjust for the opponent

Model implementation

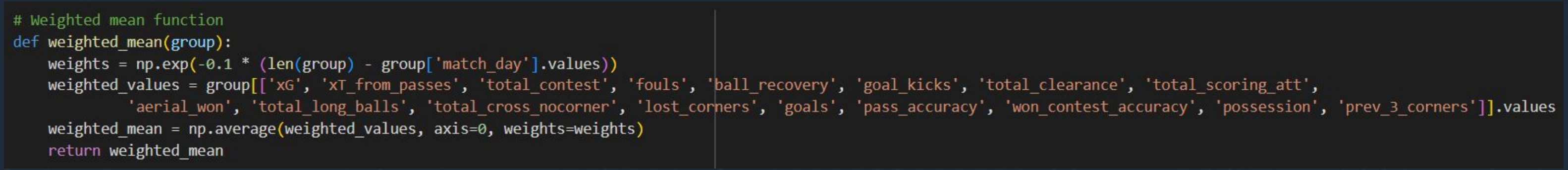
For creating this estimator, we did the following steps:

1. Linked the coefficients to each variable.



```python
# Link coeficients to variables for further calculation
variables = ['Intercept', 'xG', 'xT_from_passes', 'total_contest', 'fouls', 'ball_recovery', 'goal_kicks', 'total_clearance', 'total_scoring_att',
             'aerial_won', 'total_long_balls', 'total_cross_nocorner', 'lost_corners', 'goals', 'pass_accuracy', 'won_contest_accuracy', 'possession', 'prev_3_corners']

data = {'Variables': [model.intercept_] + list(model.coef_)}

df_coefficient = pd.DataFrame(data, index=variables).T

df_coefficient
```

2. Created a function to calculate a weighted average for each variable, giving lower “importance” to older matches.



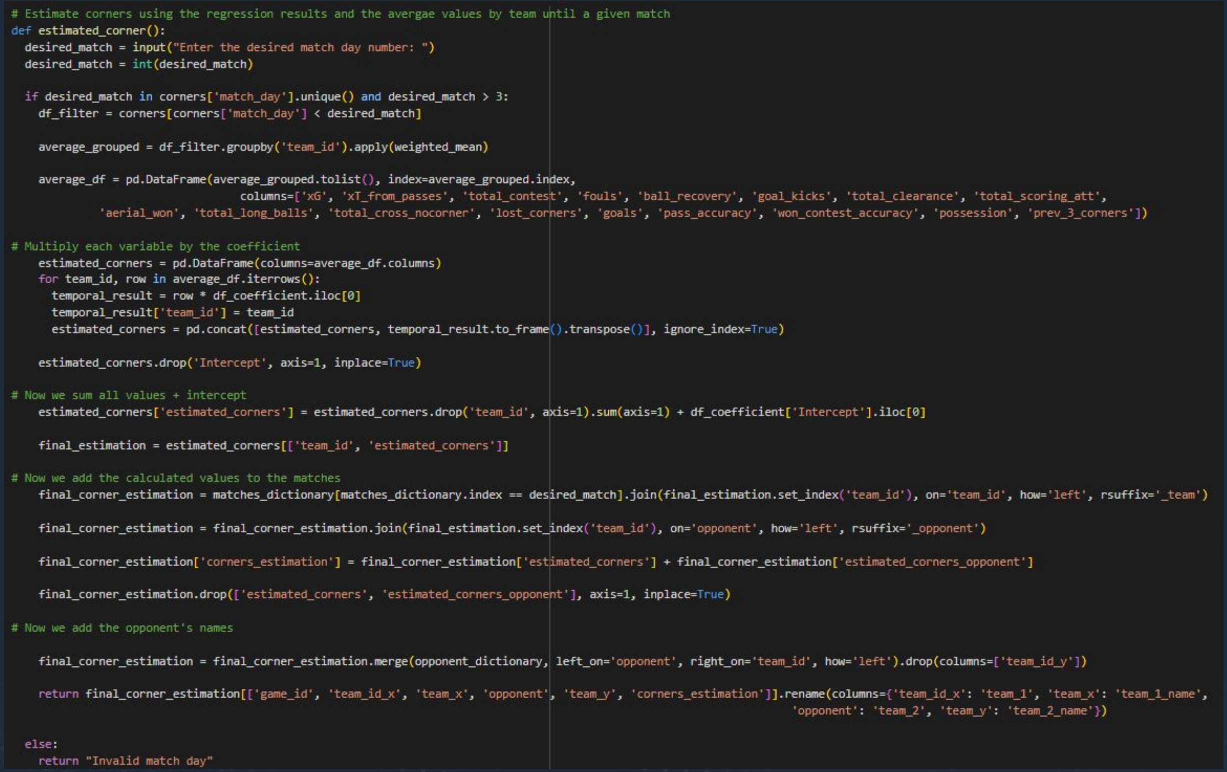
```python
# Weighted mean function
def weighted_mean(group):
    weights = np.exp(-0.1 * (len(group) - group['match_day'].values))
    weighted_values = group[['xG', 'xT_from_passes', 'total_contest', 'fouls', 'ball_recovery', 'goal_kicks', 'total_clearance', 'total_scoring_att',
            'aerial_won', 'total_long_balls', 'total_cross_nocorner', 'lost_corners', 'goals', 'pass_accuracy', 'won_contest_accuracy', 'possession', 'prev_3_corners']].values
    weighted_mean = np.average(weighted_values, axis=0, weights=weights)
    return weighted_mean
```

Model implementation

For creating this estimator, we did the following steps:

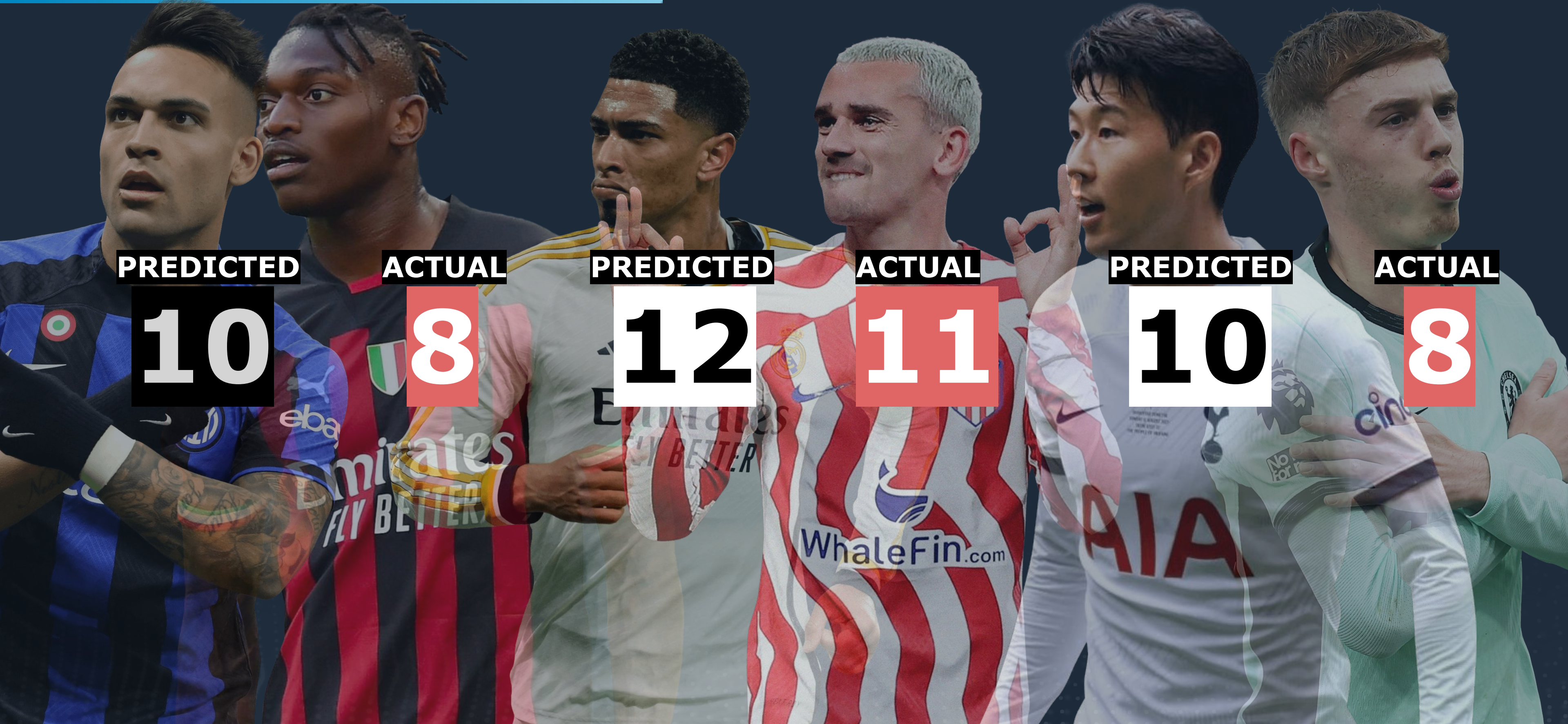
3. Created a final function that reproduces a regression, applying the intercept and coefficients to the variables already calculated with the weighted average.

This function is complex, so a future improvement can be separate internal steps into new functions, so it is more readable and organized.



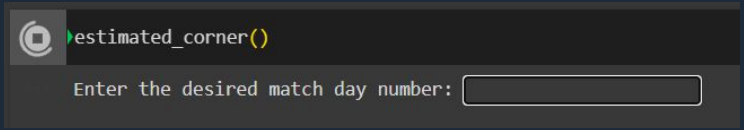
```python
# Estimate corners using the regression results and the avergae values by team until a given match
def estimated_corner():
  desired_match = input("Enter the desired match day number: ")
  desired_match = int(desired_match)

  if desired_match in corners['match_day'].unique() and desired_match > 3:
    df_filter = corners[corners['match_day'] < desired_match]

    average_grouped = df_filter.groupby('team_id').apply(weighted_mean)

    average_df = pd.DataFrame(average_grouped.tolist(), index=average_grouped.index,
                              columns=['xG', 'xT_from_passes', 'total_contest', 'fouls', 'ball_recovery', 'goal_kicks', 'total_clearance', 'total_scoring_att',
             'aerial_won', 'total_long_balls', 'total_cross_nocorner', 'lost_corners', 'goals', 'pass_accuracy', 'won_contest_accuracy', 'possession', 'prev_3_corners'])

# Multiply each variable by the coefficient
    estimated_corners = pd.DataFrame(columns=average_df.columns)
    for team_id, row in average_df.iterrows():
      temporal_result = row * df_coefficient.iloc[0]
      temporal_result['team_id'] = team_id
      estimated_corners = pd.concat([estimated_corners, temporal_result.to_frame().transpose()], ignore_index=True)

    estimated_corners.drop('Intercept', axis=1, inplace=True)

# Now we sum all values + intercept
    estimated_corners['estimated_corners'] = estimated_corners.drop('team_id', axis=1).sum(axis=1) + df_coefficient['Intercept'].iloc[0]

    final_estimation = estimated_corners[['team_id', 'estimated_corners']]

# Now we add the calculated values to the matches
    final_corner_estimation = matches_dictionary[matches_dictionary.index == desired_match].join(final_estimation.set_index('team_id'), on='team_id', how='left', rsuffix='_team')

    final_corner_estimation = final_corner_estimation.join(final_estimation.set_index('team_id'), on='opponent', how='left', rsuffix='_opponent')

    final_corner_estimation['corners_estimation'] = final_corner_estimation['estimated_corners'] + final_corner_estimation['estimated_corners_opponent']

    final_corner_estimation.drop(['estimated_corners', 'estimated_corners_opponent'], axis=1, inplace=True)

# Now we add the opponent's names

    final_corner_estimation = final_corner_estimation.merge(opponent_dictionary, left_on='opponent', right_on='team_id', how='left').drop(columns=['team_id_y'])

    return final_corner_estimation[['game_id', 'team_id_x', 'team_x', 'opponent', 'team_y', 'corners_estimation']].rename(columns={'team_id_x': 'team_1', 'team_x': 'team_1_name',
                                                                                                                                          'opponent': 'team_2', 'team_y': 'team_2_name'})

  else:
    return "Invalid match day"
```

Predictions v Reality



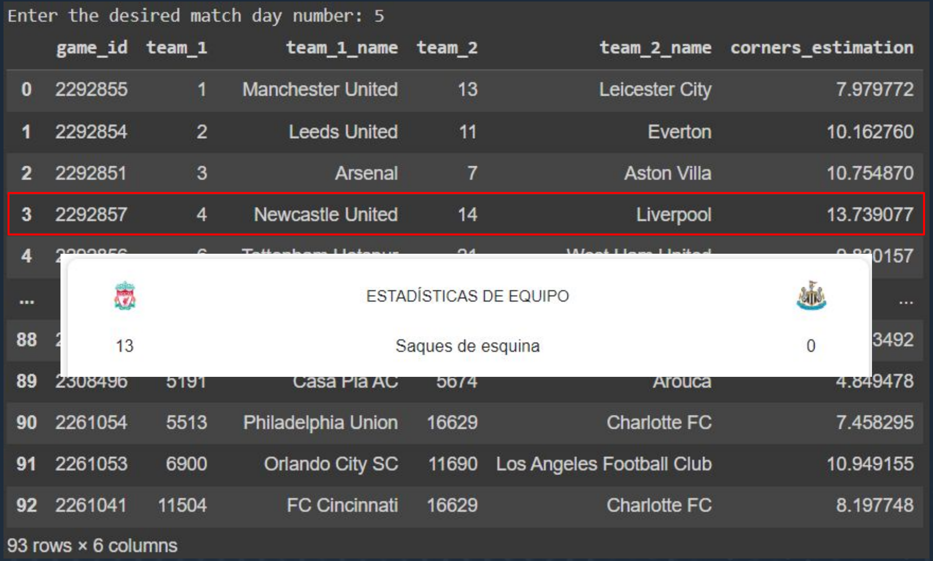
Model implementation

Some examples:



```
estimated_corner()

Enter the desired match day number: 
```

Match 5 estimation



```
Enter the desired match day number: 5
```

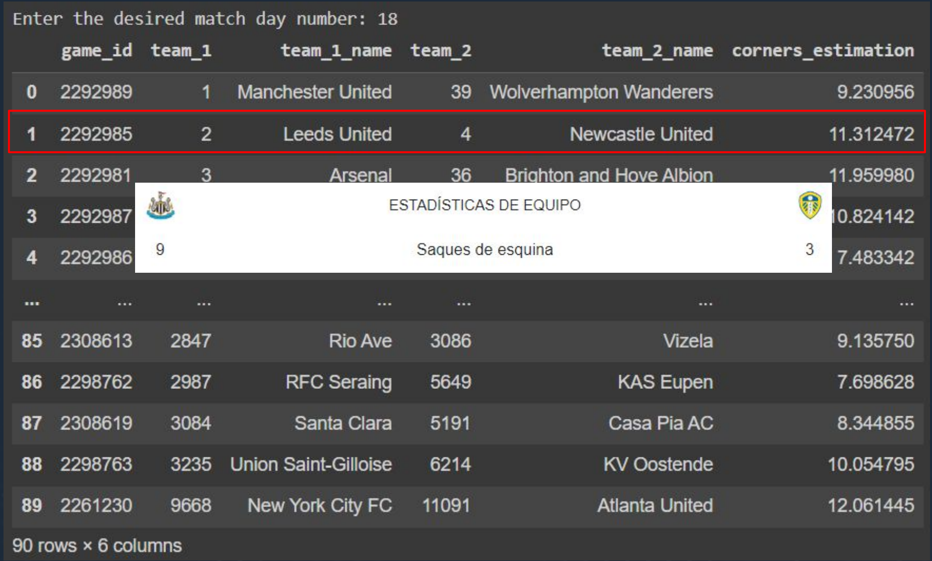
	game_id	team_1	team_1_name	team_2	team_2_name	corners_estimation
0	2292855	1	Manchester United	13	Leicester City	7.979772
1	2292854	2	Leeds United	11	Everton	10.162760
2	2292851	3	Arsenal	7	Aston Villa	10.754870
3	2292857	4	Newcastle United	14	Liverpool	13.739077
4	2292856	6	Tottenham Hotspur	21	West Ham United	9.820157
...	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	...
88	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]3492
89	2308496	5191	Casa Pia AC	5674	Arouca	4.849478
90	2261054	5513	Philadelphia Union	16629	Charlotte FC	7.458295
91	2261053	6900	Orlando City SC	11690	Los Angeles Football Club	10.949155
92	2261041	11504	FC Cincinnati	16629	Charlotte FC	8.197748

93 rows × 6 columns

ESTADÍSTICAS DE EQUIPO

13	Saques de esquina	0

Match 18 estimation



```
Enter the desired match day number: 18
```

	game_id	team_1	team_1_name	team_2	team_2_name	corners_estimation
0	2292989	1	Manchester United	39	Wolverhampton Wanderers	9.230956
1	2292985	2	Leeds United	4	Newcastle United	11.312472
2	2292981	3	Arsenal	36	Brighton and Hove Albion	11.959980
3	2292987	[illegible]	[illegible]	[illegible]	[illegible]	10.824142
4	2292986	[illegible]	[illegible]	[illegible]	[illegible]	7.483342
...	...	...	...	...	...	...
85	2308613	2847	Rio Ave	3086	Vizela	9.135750
86	2298762	2987	RFC Seraing	5649	KAS Eupen	7.698628
87	2308619	3084	Santa Clara	5191	Casa Pia AC	8.344855
88	2298763	3235	Union Saint-Gilloise	6214	KV Oostende	10.054795
89	2261230	9668	New York City FC	11091	Atlanta United	12.061445

90 rows × 6 columns

ESTADÍSTICAS DE EQUIPO

9	Saques de esquina	3

THANK YOU!

🇮🇹 **Antonio Lupo**

n/antonio-lupo

github.com/AntonioWolf01

🇹🇳 **Samaher Brahem**

in/samaherbrahem

github.com/SamaherUNIMI

samaherbrahem.com

🇦🇷 **Franco Bonifacini**

in/franco-bonifacini

github.com/Boni1995