Udacity - Data Science NanoDegree

Investigate a Dataset

Submission By: 55thSwiss

We'll be analyzing the soccer data set [1], mostly because it could be used for predictive ML and it's sports related, both of which interest me. All analysis and computation have been done in this notebook.

[1] https://www.kaggle.com/hugomathien/soccer (https://www.kaggle.com/hugomathien/soccer)

Intro

One of the most enjoyable parts about watching sports is watching a team win. This usually has to do with the fact that winning teams score more, but in this analysis I'm going to take a look at what makes a winning team, statistically, different than a losing team beyond just the goals.

We're going to manipulate the data into two basic categories, data about individual teams and data about how these teams did against one another, then we can compare what the make-up of these teams are that played one another.

Import Dependencies

```
In [1]: %matplotlib inline
        import pandas as pd
        import numpy as np
        import matplotlib as mpl
        import matplotlib.pyplot as plt
        import matplotlib.pylab as pylab
        import matplotlib.ticker as ticker
        import seaborn as sns
        from math import pi
        import sqlite3
        import datetime
        import warnings
        warnings.filterwarnings("ignore")
        pd.set_option('display.max_columns', None)
        pd.set_option('display.max_rows', None)
```

Import Data

```
In [2]: with sqlite3.connect('database.sqlite') as con:
            countries = pd.read_sql_query("SELECT * from Country", con)
            matches = pd.read_sql_query("SELECT * from Match", con)
            leagues = pd.read_sql_query("SELECT * from League", con)
            teams = pd.read_sql_query("SELECT * from Team", con)
            player = pd.read_sql_query("SELECT * from Player",con)
            player_attributes = pd.read_sql_query("SELECT * from Player_Attributes",co
        n)
            sequence = pd.read_sql_query("SELECT * from sqlite_sequence",con)
            team_attributes = pd.read_sql_query("SELECT * from Team_Attributes",con)
```

Quick Look

Since we're interested in what makes a winning team different than a losing one, a large portion of this data is
not going to be utilized, namely the 'player' and 'player_attributes' data sets. Normally I wouldn't want to
disregard such a large accumulation of data but for the sake of not making this about discrepancies, anomalies,
or general focus on the correlation between one data set and another it will be assumed the accumulation of
individual player attributes into a single 'team' is therefore represented in the 'team_attributes' data set.

```
In [3]: teams.nunique()
```

```
Out[3]: id                 299
        team_api_id        299
        team_fifa_api_id   285
        team_long_name     296
        team_short_name    259
        dtype: int64
```

```
In [4]: team_attributes.nunique()
```

```
Out[4]: id                                  1458
        team_fifa_api_id                     285
        team_api_id                          288
        date                                   6
        buildUpPlaySpeed                      57
        buildUpPlaySpeedClass                  3
        buildUpPlayDribbling                  49
        buildUpPlayDribblingClass              3
        buildUpPlayPassing                    58
        buildUpPlayPassingClass                3
        buildUpPlayPositioningClass            2
        chanceCreationPassing                 50
        chanceCreationPassingClass             3
        chanceCreationCrossing                56
        chanceCreationCrossingClass            3
        chanceCreationShooting                57
        chanceCreationShootingClass            3
        chanceCreationPositioningClass         2
        defencePressure                       48
        defencePressureClass                   3
        defenceAggression                     47
        defenceAggressionClass                 3
        defenceTeamWidth                      43
        defenceTeamWidthClass                  3
        defenceDefenderLineClass               2
        dtype: int64
```

```
In [5]: countries.nunique()
```

```
Out[5]: id      11
        name    11
        dtype: int64
```

```
In [6]: leagues.nunique()
```

```
Out[6]: id            11
        country_id    11
        name          11
        dtype: int64
```

```
In [7]: matches[['country_id',
               'league_id',
               'season',
               'date',
               'match_api_id',
               'home_team_api_id',
               'away_team_api_id']].nunique()
```

```
Out[7]: country_id          11
        league_id           11
        season               8
        date              1694
        match_api_id     25979
        home_team_api_id   299
        away_team_api_id   299
        dtype: int64
```

Merge Data

Team and Attributes

```
In [8]: # combine the df's on already existing features
        teamsDF = teams.merge(team_attributes, on=['team_api_id', 'team_fifa_api_id'])
        # get rid of some creationed columns from the merge
        teamsDF.drop(['id_x', 'id_y'], axis=1, inplace=True)
        # We only want one entry per team per year, so we need to eliminate duplicates
         based on the date and team name
        teamsDF.drop_duplicates(subset=['date', 'team_long_name'], inplace = True)
        teamsDF.sample(3)
```

Out[8]:

	team_api_id	team_fifa_api_id	team_long_name	team_short_name	date	buildU
508	8178	32.0	Bayer 04 Leverkusen	LEV	2014-09-19 00:00:00	69
992	8569	110744.0	GKS Bełchatów	BEL	2011-02-22 00:00:00	47
573	9788	23.0	Borussia Mönchengladbach	GLA	2011-02-22 00:00:00	63
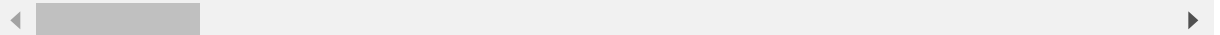
```
In [9]: print(teams.shape)
        print(team_attributes.shape)
        print(teamsDF.shape)

        (299, 5)
        (1458, 25)
        (1450, 26)
```

Country, League, and Match

```
In [10]: # get the initial merge
         leaguesDF = countries.merge(leagues, on=['id'])
         # rename
         leaguesDF = leaguesDF.rename(columns={'name_x':'Country', 'name_y':'League'})
         leaguesDF.sample(3)
```

Out[10]:

	id	Country	country_id	League
3	7809	Germany	7809	Germany 1. Bundesliga
10	24558	Switzerland	24558	Switzerland Super League
9	21518	Spain	21518	Spain LIGA BBVA

```
In [11]: # now import that 'matches' data set
         leaguesDF = leaguesDF.merge(matches, on = ['country_id'])
         leaguesDF.sample(3)
```

Out[11]:

	id_x	Country	country_id	League	id_y	league_id	season	stage	
14122	13274	Netherlands	13274	Netherlands Eredivisie	14123	13274	2010/2011	33	2 0 0
9191	7809	Germany	7809	Germany 1. Bundesliga	9192	7809	2012/2013	25	2 0 0
12719	10257	Italy	10257	Italy Serie A	12720	10257	2014/2015	28	2 0 0

◄ ▬ ▶

In [12]:
```python
a, b, c, d = countries.shape, leagues.shape, matches.shape, leaguesDF.shape
print(a)
print(b)
print(c)
print(d)
```

```
(11, 2)
(11, 3)
(25979, 115)
(25979, 118)
```

In [13]:
```python
# capture only two columns
temp = teamsDF[['team_api_id', 'team_long_name']]
# rename the column to match other df for key
temp = temp.rename(columns={'team_api_id':'home_team_api_id'})
# make sure there's no duplicates
temp.drop_duplicates(subset=['home_team_api_id', 'team_long_name'], inplace =
True)
# merge them together on 'home_team_api_id'
leaguesDF = leaguesDF.merge(temp, on=['home_team_api_id'], how='left')
# rename again for away columns
temp = temp.rename(columns={'home_team_api_id':'away_team_api_id'})
# merge again
leaguesDF = leaguesDF.merge(temp, on=['away_team_api_id'], how='left')
# drop some useless features
leaguesDF.drop(['id_x', 'id_y', 'country_id', 'league_id', 'stage'], axis=1, i
nplace=True)
# create a copy of 'LeagueDF' to simplify the information even more
leaguesFinal = leaguesDF
leaguesFinal = leaguesFinal[['Country', 'League', 'season', 'date', 'match_api
_id', 'team_long_name_x',
                            'team_long_name_y', 'home_team_goal', 'away_team_goal'
]]
```

In [14]:
```python
# rename for clarity
leaguesFinal = leaguesFinal.rename(columns={'team_long_name_x':'Home Team', 't
eam_long_name_y':'Away Team'})
leaguesFinal.sample(3)
```

Out[14]:

	Country	League	season	date	match_api_id	Home Team	Away Team	hon
17177	Poland	Poland Ekstraklasa	2014/2015	2014-10-05 00:00:00	1722172	Piast Gliwice	Legia Warszawa	3
10926	Italy	Italy Serie A	2009/2010	2010-05-02 00:00:00	705237	Atalanta	Bologna	1
23055	Spain	Spain LIGA BBVA	2012/2013	2012-11-04 00:00:00	1260150	RC Deportivo de La Coruña	RCD Mallorca	1

In [15]:
```python
# functions to decide the winner and loser of each match and append the df wit
h that feature

def win(leaguesFinal):
    if leaguesFinal['home_team_goal'] > leaguesFinal['away_team_goal']:
        return leaguesFinal['Home Team']
    elif leaguesFinal['away_team_goal'] > leaguesFinal['home_team_goal']:
        return leaguesFinal['Away Team']
    elif leaguesFinal['home_team_goal'] == leaguesFinal['away_team_goal']:
        return "DRAW"

def loss(leaguesFinal):
    if leaguesFinal['home_team_goal'] < leaguesFinal['away_team_goal']:
        return leaguesFinal['Home Team']
    elif leaguesFinal['away_team_goal'] < leaguesFinal['home_team_goal']:
        return leaguesFinal['Away Team']

leaguesFinal["win"] = leaguesFinal.apply(lambda leaguesFinal:win(leaguesFinal
),axis=1)
leaguesFinal["loss"] = leaguesFinal.apply(lambda leaguesFinal:loss(leaguesFina
l),axis=1)
```

In [16]: `leaguesFinal.shape`

Out[16]: (25979, 11)

In [17]: `leaguesFinal.sample(3)`

Out[17]:

	Country	League	season	date	match_api_id	Home Team	Away Team	home
1638	Belgium	Belgium Jupiler League	2015/2016	2016-02-14 00:00:00	1980030	KVC Westerlo	Standard de Liège	2
8363	Germany	Germany 1. Bundesliga	2009/2010	2010-05-08 00:00:00	674582	1. FSV Mainz 05	FC Schalke 04	0
802	Belgium	Belgium Jupiler League	2011/2012	2011-11-19 00:00:00	1032802	KV Mechelen	SV Zulte-Waregem	1

Feature Engineering

Team Record (Wins / Losses) DataFrame

Draws are being diregarded in this analysis because they do not help to determine what makes a winning team different than a losing team.

In [18]:
```python
# get indices
seasons = leaguesFinal['season'].unique()
teams = teamsDF['team_long_name'].unique()
# create an empty list
df = []

# first separate by season
for i in seasons:
    season = leaguesFinal['season'] == i
    season = leaguesFinal[season]
    # then count the games won and games lost
    for j in teams:
        team_season_wins = season['win'] == j
        team_season_win_record = team_season_wins[team_season_wins].count()
        team_season_loss = season['loss'] == j
        team_season_loss_record = team_season_loss[team_season_loss].count()
        df.append((j, i, team_season_win_record, team_season_loss_record))
# create the new df and feature names
df = pd.DataFrame(df, columns=('Team', 'Seasons', 'Wins', 'Losses'))
# rename the team column to use as a key
df = df.rename(columns={'Team':'Home Team'})
# take just the information we want to merge ('League') plus a key column for
 merging
df2 = leaguesFinal[['Home Team', 'League']]
# clean it up, we only want one team name per league, there are no dates assoc
iated with df2
df2.drop_duplicates(subset = ['Home Team'],inplace = True)
# the merge
df = df.merge(df2, on = 'Home Team')
# change the feature name back
df = df.rename(columns={'Home Team':'Team'})
# create an identifiable df name
teamRecords = df[['League', 'Team', 'Seasons', 'Wins', 'Losses']]
# drop some outlier rows with odd data,
# seemed to consistantly contain '0' for either win or loss
teamRecords = teamRecords[teamRecords.Wins != 0]
teamRecords = teamRecords[teamRecords.Losses != 0]
teamRecords.sample(5)
```

Out[18]:

	League	Team	Seasons	Wins	Losses
1087	Italy Serie A	Sampdoria	2015/2016	10	18
1551	Poland Ekstraklasa	Zagłębie Lubin	2015/2016	12	9
2224	Switzerland Super League	FC Vaduz	2008/2009	5	24
1512	Poland Ekstraklasa	Jagiellonia Białystok	2008/2009	9	14
2197	Switzerland Super League	FC Basel	2013/2014	19	2

Team Goals by Season

```
In [19]: # create another list
         df = []
         # group by 'home' or 'away' and season and sum the goal column
         home_goals = leaguesFinal.groupby(('Home Team', 'season'))['home_team_goal'].s
         um()
         away_goals = leaguesFinal.groupby(('Away Team', 'season'))['away_team_goal'].s
         um()
         # lose the win and loss title for 'Team' to merge
         a = home_goals.rename_axis(['Team','season'])
         b = away_goals.rename_axis(['Team','season'])
         # fill any NaN values with 0 goals
         df = (a.add(b, fill_value=0)).reset_index(name='Goals')
         df = df.rename(columns={'season':'Seasons'})
         # the merge
         teamRecords = teamRecords.merge(df, on = ['Team', 'Seasons'], how = 'left')
         # organize for consistancy
         teamRecords.sort_values(['League', 'Team', 'Seasons'], ascending = True, inpla
         ce = True)
         teamRecords.shape
```

Out[19]: (1453, 6)

In [20]: `teamRecords.sample(3)`

Out[20]:

	League	Team	Seasons	Wins	Losses	Goals
396	France Ligue 1	Montpellier Hérault SC	2009/2010	20	9	50
81	Belgium Jupiler League	KVC Westerlo	2008/2009	15	12	42
1052	Portugal Liga ZON Sagres	Académica de Coimbra	2015/2016	5	19	32

League Winners

create a df of the team with the best record from each 'League' for each 'Seasons'

In [21]:
```python
# create the df to work with
leagueWinners_season = teamRecords
# organize for what matters
leagueWinners_season.sort_values(['League', 'Seasons', 'Wins'], ascending = False, inplace = True)
# we don't care about the bottom of the barrel teams, they won't be a league winner
leagueWinners_season = leagueWinners_season[leagueWinners_season.Wins > 10]
# grab the first row in each combination of 'League' and 'Season'
leagueWinners_season = leagueWinners_season.groupby(['League', 'Seasons']).first()
# display winning teams of each league by season
leagueWinners_season.head(leagueWinners_season['Team'].count())
```

Out[21]:

League	Seasons	Team	Wins	Losses	Goals
Belgium Jupiler League	**2008/2009**	RSC Anderlecht	24	5	75
	2009/2010	RSC Anderlecht	22	3	62
	2010/2011	KRC Genk	19	4	64
	2011/2012	RSC Anderlecht	20	3	61
	2012/2013	RSC Anderlecht	20	3	69
	2014/2015	Club Brugge KV	17	3	69
	2015/2016	Club Brugge KV	21	8	64
England Premier League	**2008/2009**	Manchester United	28	4	68
	2009/2010	Chelsea	27	6	103
	2010/2011	Manchester United	23	4	78
	2011/2012	Manchester City	28	5	93
	2012/2013	Manchester United	28	5	86
	2013/2014	Manchester City	27	6	102
	2014/2015	Chelsea	26	3	73
	2015/2016	Leicester City	23	3	68
France Ligue 1	**2008/2009**	Girondins de Bordeaux	24	6	64
	2009/2010	Olympique de Marseille	23	6	69
	2010/2011	LOSC Lille	21	4	68
	2011/2012	Montpellier Hérault SC	25	6	68
	2012/2013	Paris Saint-Germain	25	5	69
	2013/2014	Paris Saint-Germain	27	3	84
	2014/2015	Paris Saint-Germain	24	3	83
	2015/2016	Paris Saint-Germain	30	2	102
Germany 1. Bundesliga	**2008/2009**	VfL Wolfsburg	21	7	80
	2009/2010	FC Bayern Munich	20	4	72
	2010/2011	Borussia Dortmund	23	5	67
	2011/2012	Borussia Dortmund	25	3	80
	2012/2013	FC Bayern Munich	29	1	98
	2013/2014	FC Bayern Munich	29	2	94
	2014/2015	FC Bayern Munich	25	5	80
	2015/2016	FC Bayern Munich	28	2	80

League	Seasons	Team	Wins	Losses	Goals
Italy Serie A	**2008/2009**	Inter	25	4	70
	2009/2010	Inter	24	4	75
	2010/2011	Milan	24	4	65
	2011/2012	Milan	23	6	72
	2012/2013	Juventus	27	5	71
	2013/2014	Juventus	33	2	80
	2014/2015	Juventus	26	3	72
	2015/2016	Juventus	29	5	75
Netherlands Eredivisie	**2008/2009**	AZ	25	4	66
	2009/2010	Ajax	27	3	106
	2010/2011	Ajax	22	5	72
	2011/2012	Ajax	23	4	93
	2012/2013	Ajax	22	2	83
	2013/2014	Ajax	20	3	69
	2014/2015	PSV	29	4	92
	2015/2016	PSV	26	2	88
Poland Ekstraklasa	**2008/2009**	Wisła Kraków	19	4	53
	2009/2010	Lech Poznań	19	3	51
	2010/2011	Wisła Kraków	17	8	44
	2011/2012	Śląsk Wrocław	17	8	47
	2012/2013	Legia Warszawa	20	3	59
	2013/2014	Legia Warszawa	20	7	60
	2014/2015	Legia Warszawa	17	8	57
	2015/2016	Legia Warszawa	17	4	58
Portugal Liga ZON Sagres	**2008/2009**	FC Porto	21	2	61
	2009/2010	SL Benfica	24	2	78
	2010/2011	SL Benfica	20	7	61
	2011/2012	FC Porto	23	1	69
	2012/2013	SL Benfica	24	1	77
	2013/2014	SL Benfica	23	2	58
	2014/2015	SL Benfica	27	3	86

League	Seasons	Team	Wins	Losses	Goals
	2015/2016	SL Benfica	29	4	88
Scotland Premier League	**2008/2009**	Rangers	26	4	77
	2009/2010	Rangers	26	3	82
	2010/2011	Rangers	30	5	88
	2011/2012	Celtic	30	5	84
	2012/2013	Celtic	24	7	92
	2013/2014	Celtic	31	1	102
	2014/2015	Celtic	29	4	84
	2015/2016	Celtic	26	4	93
Spain LIGA BBVA	**2008/2009**	FC Barcelona	27	5	105
	2009/2010	FC Barcelona	31	1	98
	2010/2011	FC Barcelona	30	2	95
	2011/2012	Real Madrid CF	32	2	121
	2012/2013	FC Barcelona	32	2	115
	2013/2014	Atlético Madrid	28	4	77
	2014/2015	FC Barcelona	30	4	110
	2015/2016	FC Barcelona	29	5	112
Switzerland Super League	**2008/2009**	FC Zürich	24	5	80
	2009/2010	BSC Young Boys	25	9	78
	2010/2011	FC Basel	21	5	76
	2011/2012	FC Basel	22	4	78
	2012/2013	FC Basel	21	6	61
	2013/2014	FC Basel	19	2	70
	2014/2015	FC Basel	24	6	84
	2015/2016	FC Basel	26	5	88

League Losers

In [22]:
```python
# create the df to work with
leagueLosers_season = teamRecords
# organize for what matters
leagueLosers_season.sort_values(['League', 'Seasons', 'Losses'], ascending = False, inplace = True)
# we don't care about the bottom of the barrel teams, they won't be a league losers
leagueLosers_season = leagueLosers_season[leagueLosers_season.Losses > 10]
# grab the first row in each combination of 'League' and 'Season'
leagueLosers_season = leagueLosers_season.groupby(['League', 'Seasons']).first()
# display losing teams of each league by season
leagueLosers_season.head(leagueLosers_season['Team'].count())
```

Out[22]:

League	Seasons	Team	Wins	Losses	Goals
Belgium Jupiler League	**2008/2009**	RAEC Mons	3	21	31
	2009/2010	Sporting Lokeren	5	20	22
	2010/2011	Sporting Charleroi	4	19	20
	2011/2012	KVC Westerlo	5	20	29
	2012/2013	KSV Cercle Brugge	3	22	30
	2014/2015	Waasland-Beveren	7	18	30
	2015/2016	Sint-Truidense VV	8	16	28
England Premier League	**2008/2009**	West Bromwich Albion	8	22	36
	2009/2010	Burnley	8	24	42
	2010/2011	Wolverhampton Wanderers	11	20	46
	2011/2012	Blackburn Rovers	8	23	48
	2012/2013	Reading	6	22	43
	2013/2014	Fulham	9	24	40
	2014/2015	Queens Park Rangers	8	24	42
	2015/2016	Aston Villa	3	27	27
France Ligue 1	**2008/2009**	Le Havre AC	7	26	30
	2009/2010	Grenoble Foot 38	5	25	31
	2010/2011	AC Arles-Avignon	3	24	21
	2011/2012	Dijon FCO	9	20	38
	2012/2013	Stade Brestois 29	8	25	32
	2013/2014	Valenciennes FC	7	23	37
	2014/2015	Évian Thonon Gaillard FC	11	23	41
	2015/2016	ES Troyes AC	3	26	28
Germany 1. Bundesliga	**2008/2009**	Karlsruher SC	8	21	30
	2009/2010	Hertha BSC Berlin	5	20	34
	2010/2011	FC St. Pauli	8	21	35
	2011/2012	1. FC Köln	8	20	39
	2012/2013	SpVgg Greuther Fürth	4	21	26
	2013/2014	Hamburger SV	7	21	51
	2014/2015	Hamburger SV	9	17	25
	2015/2016	Hannover 96	7	23	31

			Team	Wins	Losses	Goals
League	**Seasons**					
Italy Serie A	**2008/2009**	Torino		8	20	37
	2009/2010	Livorno		7	23	27
	2010/2011	Bari		5	24	27
	2011/2012	Cesena		4	22	24
	2012/2013	Pescara		6	28	27
	2013/2014	Livorno		6	25	39
	2014/2015	Parma		6	24	33
	2015/2016	Frosinone		8	23	35
Netherlands Eredivisie	**2008/2009**	ADO Den Haag		8	18	41
	2009/2010	RKC Waalwijk		5	29	30
	2010/2011	VVV-Venlo		6	25	34
	2011/2012	Excelsior		4	23	28
	2012/2013	Willem II		5	21	33
	2013/2014	Roda JC Kerkrade		7	19	44
	2014/2015	FC Dordrecht		4	22	24
	2015/2016	SC Cambuur		3	22	33
Poland Ekstraklasa	**2008/2009**	Lechia Gdańsk		9	16	30
	2009/2010	Odra Wodzisław		7	17	27
	2010/2011	Cracovia		8	17	37
	2011/2012	Widzew Łódź		5	16	23
	2012/2013	Pogoń Szczecin		10	15	29
	2013/2014	Zagłębie Lubin		7	15	31
	2014/2015	Zawisza Bydgoszcz		8	17	32
	2015/2016	Jagiellonia Białystok		10	15	37
Portugal Liga ZON Sagres	**2008/2009**	Vitória Setúbal		7	18	21
	2009/2010	Leixões SC		5	19	25
	2010/2011	Naval 1° de Maio		5	17	26
	2011/2012	União de Leiria, SAD		5	21	25
	2012/2013	Vitória Setúbal		7	18	30
	2013/2014	FC Paços de Ferreira		6	18	28
	2014/2015	FC Penafiel		5	22	29

League	Seasons	Team	Wins	Losses	Goals
	2015/2016	CS Marítimo	10	19	45
Scotland Premier League	2008/2009	Hamilton Academical FC	12	21	30
	2009/2010	Kilmarnock	8	21	29
	2010/2011	Aberdeen	11	22	39
	2011/2012	Dunfermline Athletic	5	23	40
	2012/2013	Dundee FC	7	22	28
	2013/2014	Kilmarnock	11	21	45
	2014/2015	St. Mirren	9	26	30
	2015/2016	Dundee United	8	23	45
Spain LIGA BBVA	2008/2009	Real Sporting de Gijón	14	23	47
	2009/2010	CD Tenerife	9	20	40
	2010/2011	Real Sociedad	14	21	49
	2011/2012	Rayo Vallecano	13	21	53
	2012/2013	Real Zaragoza	9	22	37
	2013/2014	Real Betis Balompié	6	25	36
	2014/2015	Córdoba CF	3	24	22
	2015/2016	Levante UD	8	22	37
Switzerland Super League	2008/2009	FC Vaduz	5	24	28
	2009/2010	AC Bellinzona	7	25	42
	2010/2011	FC St. Gallen	8	21	34
	2011/2012	Grasshopper Club Zürich	7	22	32
	2012/2013	Servette FC	6	22	32
	2013/2014	FC Lausanne-Sports	7	26	38
	2014/2015	FC Vaduz	7	19	28
	2015/2016	FC St. Gallen	10	18	41

Team Attributes Catagories

In [23]: `teamsDF.sample(3)`

Out[23]:

	team_api_id	team_fifa_api_id	team_long_name	team_short_name	date	buildU
200	9879	144.0	Fulham	FUL	2011-02-22 00:00:00	50
189	10194	1806.0	Stoke City	STK	2012-02-22 00:00:00	75
986	2182	873.0	Lech Poznań	POZ	2011-02-22 00:00:00	64
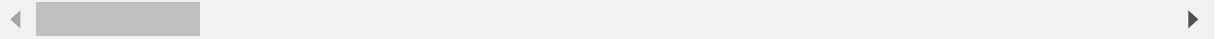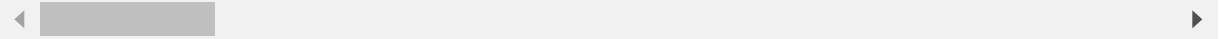
In [24]:
```python
# left the 'date' column as an object to easily slice off the 00:00:00 from th
e feature
teamsDF['date'] = teamsDF['date'].map(lambda x: x.rstrip(' 00:'))
# sort for testing
teamsDF.sort_values(['team_long_name', 'date'], inplace = True)
# manipulation df
df = teamsDF
# empty list
lst = []

for d in df['date']:
    # create variables from the year, month, and day in 'date' feature
    datee = datetime.datetime.strptime(d, '%Y-%m-%d')
    # assigment
    currentYear = datee.year
    # decide what season it is
    if datee.month < 7: # https://en.m.wikipedia.org/wiki/Domestic_association
_football_season
        season = str(currentYear - 1) + '/' + str(currentYear)
    else:
        season = str(currentYear) + '/' + str(currentYear + 1)
    # compile the list with a reference to the original df(d) for merging
    lst.append((d, season))
# create the second df for merging
df2 = pd.DataFrame(lst, columns = ['date', 'Seasons'])
# merge on original 'date' feature
df = df.merge(df2, on = 'date')
# merged df was huge, creating dulipcates
# this manipulation df was made only to merge with 'leagueWinners_season', so
 we don't care about losing edge features / data
df.drop_duplicates(subset = ['date', 'team_long_name', 'Seasons'],inplace = Tr
ue)
# checking
df.sort_values(['team_long_name', 'date'], inplace = True)
# format feature name for merge
df = df.rename(columns = {'team_long_name' : 'Team'})
# create final df with 'leagueWinners' and all their attributes
leagueWinners_attributes = leagueWinners_season.merge(df, on = ['Seasons', 'Te
am'], how = 'left')
leagueLosers_attributes = leagueLosers_season.merge(df, on = ['Seasons', 'Tea
m'], how = 'left')
# get rid of unneeded features for the radar charts
leagueWinners_attributes.drop(['team_api_id', 'team_fifa_api_id', 'team_short_
name', 'date'], axis=1, inplace=True)
leagueLosers_attributes.drop(['team_api_id', 'team_fifa_api_id', 'team_short_n
ame', 'date'], axis=1, inplace=True)
leagueWinners_attributes.sample(3)
```

Out[24]:

	Seasons	Team	Wins	Losses	Goals	buildUpPlaySpeed	buildUpPlaySpeedClas
57	2010/2011	SL Benfica	20	7	61	65.0	Balanced
26	2011/2012	Borussia Dortmund	25	3	80	71.0	Fast
19	2012/2013	Paris Saint-Germain	25	5	69	NaN	NaN

create a function that compares the winning team attributes vs losing team attributes in

three major categories: Build Up, Offense, and Defense

list features that'll be compared

means = ['buildUpPlaySpeed', 'buildUpPlayDribbling', 'buildUpPlayPassing', 'chanceCreationPassing', 'chanceCreationCrossing', 'chanceCreationShooting', 'defencePressure', 'defenceAggression', 'defenceTeamWidth']

fill the NaN values with the median. Had the best results compared with mean and mode

for m in means: leagueLosers_attributes[m].fillna((leagueLosers_attributes[m].median()), inplace=True) leagueWinners_attributes[m].fillna((leagueWinners_attributes[m].median()), inplace=True)

create a distinguishing feature for the 'hue'

leagueWinners_attributes['Side'] = 'Winners' leagueLosers_attributes['Side'] = 'Losers'

learned this simple new way to concat haha

frames = [leagueLosers_attributes, leagueWinners_attributes] df = pd.concat(frames)

separate the three frames

df1 = df[['buildUpPlaySpeed', 'buildUpPlayDribbling', 'buildUpPlayPassing', 'Side']].reset_index() title1 = 'League Winner vs Loser Build Up Play Attributes' df1.drop(['index'], axis=1, inplace=True) df2 = df[['chanceCreationPassing', 'chanceCreationCrossing', 'chanceCreationShooting', 'Side']].reset_index() title2 = 'League Winner vs Loser Offense Attributes' df2.drop(['index'], axis=1, inplace=True) df3 = df[['defencePressure', 'defenceAggression', 'defenceTeamWidth', 'Side']].reset_index() title3 = 'League Winner vs Loser Defense Attributes' df3.drop(['index'], axis=1, inplace=True)

create the function for viewing

def pair_plot(df, title):

```
    # plot chart
    sns.pairplot(df, kind = 'scatter', hue = 'Side')
    plt.title(title, y = 3.4, x = -1.31, fontsize = 18)
    plt.show()
```

Visualization and Analysis

In [25]:
```python
goals_per_year = []
seasons = leaguesFinal['season'].unique()

for i in range(0,8):
    mask = leaguesFinal['season'] == seasons[i]
    goals = leaguesFinal[mask]['home_team_goal'].sum() + leaguesFinal[mask]['a
way_team_goal'].sum()
    goals_per_year.append(goals)

df = pd.DataFrame([goals_per_year]).transpose()
df['Season'] = ['2008/2009', '2009/2010', '2010/2011', '2011/2012', '2012/201
3', '2013/2014', '2014/2015', '2015/2016']
df = df.rename(columns = {0 : 'Goals'})

#
sns.set_style("darkgrid")
plt.figure(figsize=(16, 4))
plt.plot(df['Season'], df['Goals'], color = 'royalblue')
plt.ylabel('Goals', fontsize = 14)
plt.xlabel('Season', fontsize = 14)
plt.title('Goals Scored in All Leagues by Year', fontsize = 16)
mpl.rcParams['agg.path.chunksize'] = 10000
```


Besides the 2013/2014 season, European soccer has seen a gradual increase in scoring every year. I'm unaware of historical rule changes and things like that which could contribute to this besides the overall offensive talent of a team increasing, but this is very similar to most other sports where the scoring has been rising consistently.

In [26]:
```python
d = teamRecords.sort_values(by = 'Wins', ascending=False)
plt.figure(figsize=(14,5))
sns.barplot('Team', 'Wins', data = d[:50], color = 'b', label = 'Wins')
sns.barplot('Team', 'Losses', data = d[:50], color = 'r', label = 'Losses')
plt.xticks(rotation = 70, fontsize = 12)
plt.xlabel('Teams', fontsize = 14)
plt.ylabel('Games', fontsize = 14)
plt.legend(loc="best")
plt.title('Top Records by Team', fontsize = 16)
plt.show()
```
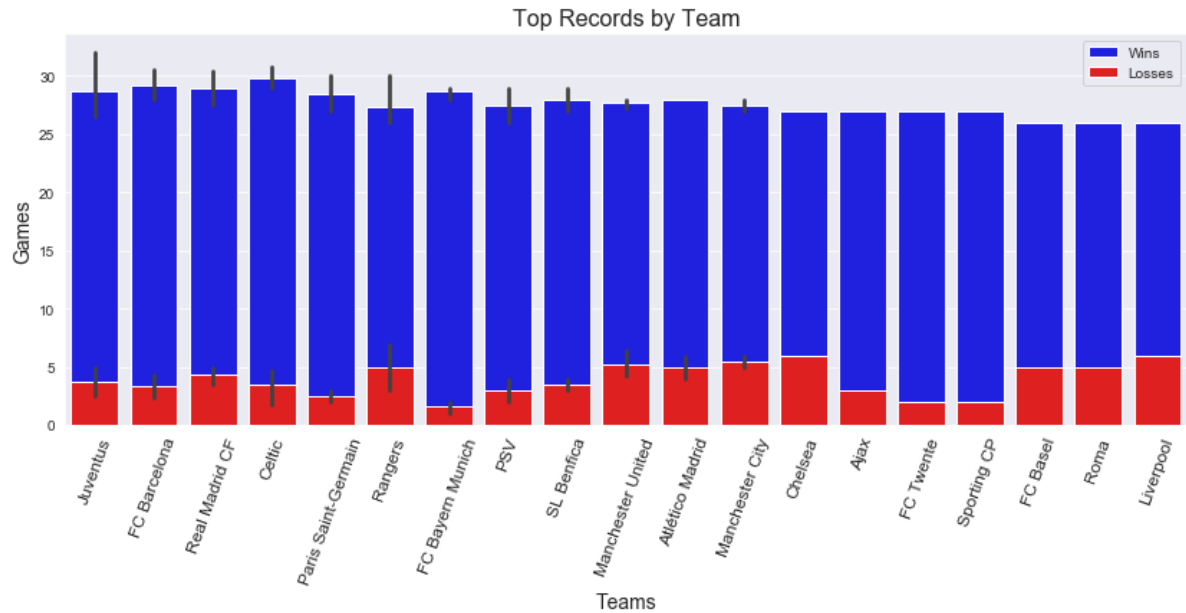

It's no surprise the teams at the very top of their league have a supurb win to loss ration. This would also be that teams best year if the team had more than one winning season.

```
In [27]:  sns.jointplot(leagueWinners_season['Wins'], leagueWinners_season['Goals'], kin
          d = 'reg', color = 'b', height = 8)
          plt.title('Season Winner\'s Goal and Win Regression', y = 1.25, fontsize = 18)
          plt.xlabel('Wins', fontsize=14)
          plt.ylabel('Goals', fontsize=14)
          plt.show()
```


Season Winner's Goal and Win Regression
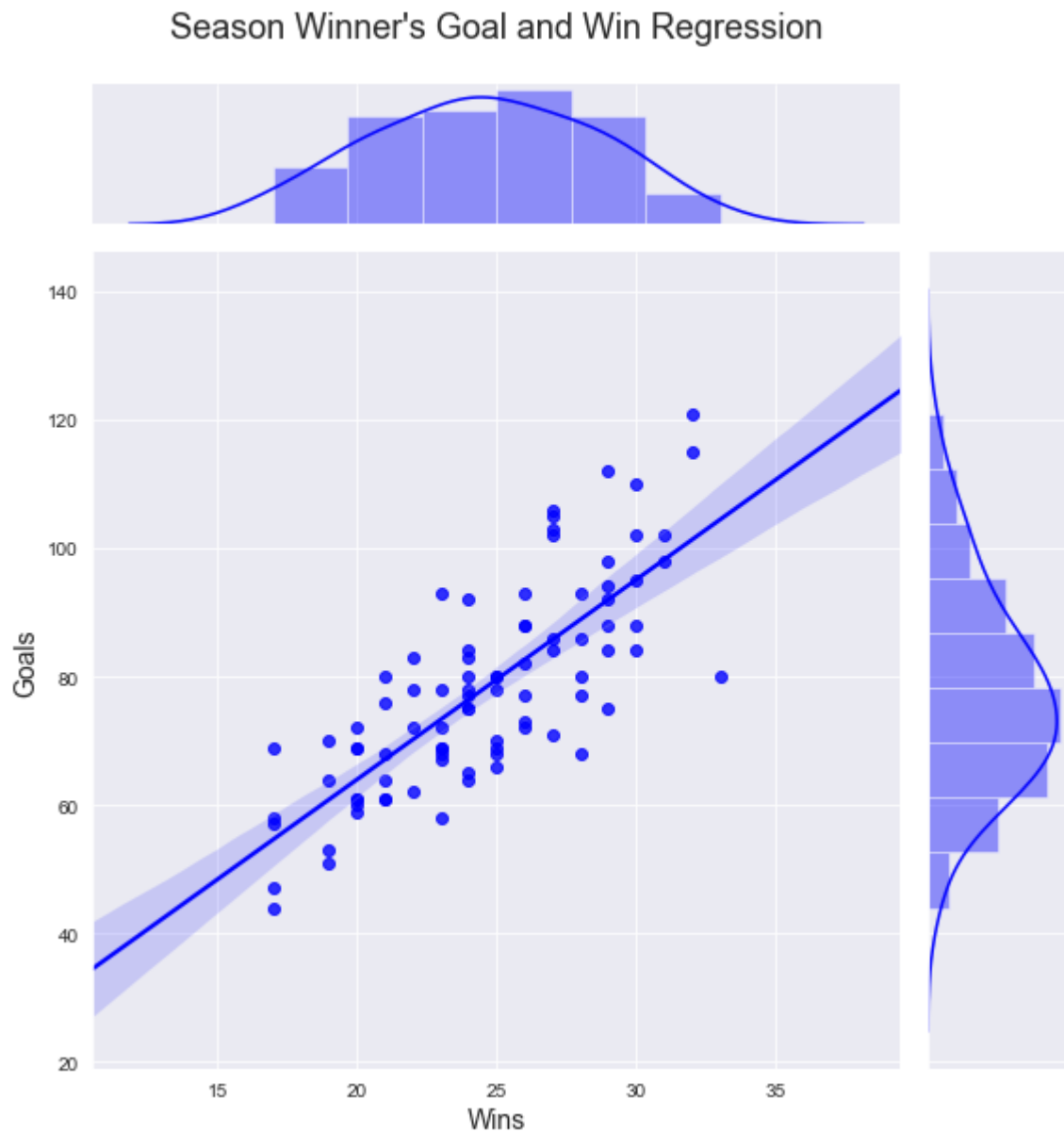
We can see the extremely positive correlation between goals and wins for the best teams in each league. The regression line is nearly a 45 degree angle.

```
In [28]: leagueWinners_season['Side'] = 'Winners'
         leagueLosers_season['Side'] = 'Losers'
         scatter = leagueWinners_season.merge(leagueLosers_season, how = 'outer')
         scatter = scatter[['Wins', 'Goals', 'Side']]
         plt.figure(figsize=(10,10))
         plt.title('Goals to Win Ratio for Highest Winning vs Highest Losing Teams', fo
         ntsize = 16)
         plt.xlabel('Wins', fontsize=14)
         plt.ylabel('Goals', fontsize=14)
         ax = sns.scatterplot(x = 'Wins', y = 'Goals', hue = 'Side', data = scatter)
```


This is the bigger picture of the last figure which now contains the closing team's goal to win ration. We can imagine with the short gap between the two clusters that the main body of all the teams in the middle would fall here with leading and trailing data points over lapping with these two clusters. The literal complete disconnect between the two clusters shows how closely related scoring and winning is.

In [29]:
```python
# create a function that compares the winning team attributes vs losing team a
ttributes in
# three major categories: Build Up, Offense, and Defense

# list features that'll be compared
means = ['buildUpPlaySpeed', 'buildUpPlayDribbling', 'buildUpPlayPassing',
                'chanceCreationPassing', 'chanceCreationCrossing', 'chanceCrea
tionShooting',
                'defencePressure', 'defenceAggression', 'defenceTeamWidth']

# fill the NaN values with the median. Had the best results compared with mean
 and mode
for m in means:
    leagueLosers_attributes[m].fillna((leagueLosers_attributes[m].median()), i
nplace=True)
    leagueWinners_attributes[m].fillna((leagueWinners_attributes[m].median()),
 inplace=True)


# create a distinguishing feature for the 'hue'
leagueWinners_attributes['Side'] = 'Winners'
leagueLosers_attributes['Side'] = 'Losers'
# learned this simple new way to concat haha
frames = [leagueLosers_attributes, leagueWinners_attributes]
df = pd.concat(frames)
# separate the three frames
df1 = df[['buildUpPlaySpeed', 'buildUpPlayDribbling', 'buildUpPlayPassing', 'S
ide']].reset_index()
title1 = 'League Winner vs Loser Build Up Play Attributes'
df1.drop(['index'], axis=1, inplace=True)
df2 = df[['chanceCreationPassing', 'chanceCreationCrossing', 'chanceCreationSh
ooting', 'Side']].reset_index()
title2 = 'League Winner vs Loser Offense Attributes'
df2.drop(['index'], axis=1, inplace=True)
df3 = df[['defencePressure', 'defenceAggression', 'defenceTeamWidth', 'Side']]
.reset_index()
title3 = 'League Winner vs Loser Defense Attributes'
df3.drop(['index'], axis=1, inplace=True)

# create the function for viewing
def pair_plot(df, title):
    # plot chart
    sns.pairplot(df, kind = 'reg', hue = 'Side', height = 4)
    plt.title(title, y = 2.2, x = -0.8, fontsize = 18)
    plt.show()
```

In [30]: `pair_plot(df1, title1)`

League Winner vs Loser Build Up Play Attributes
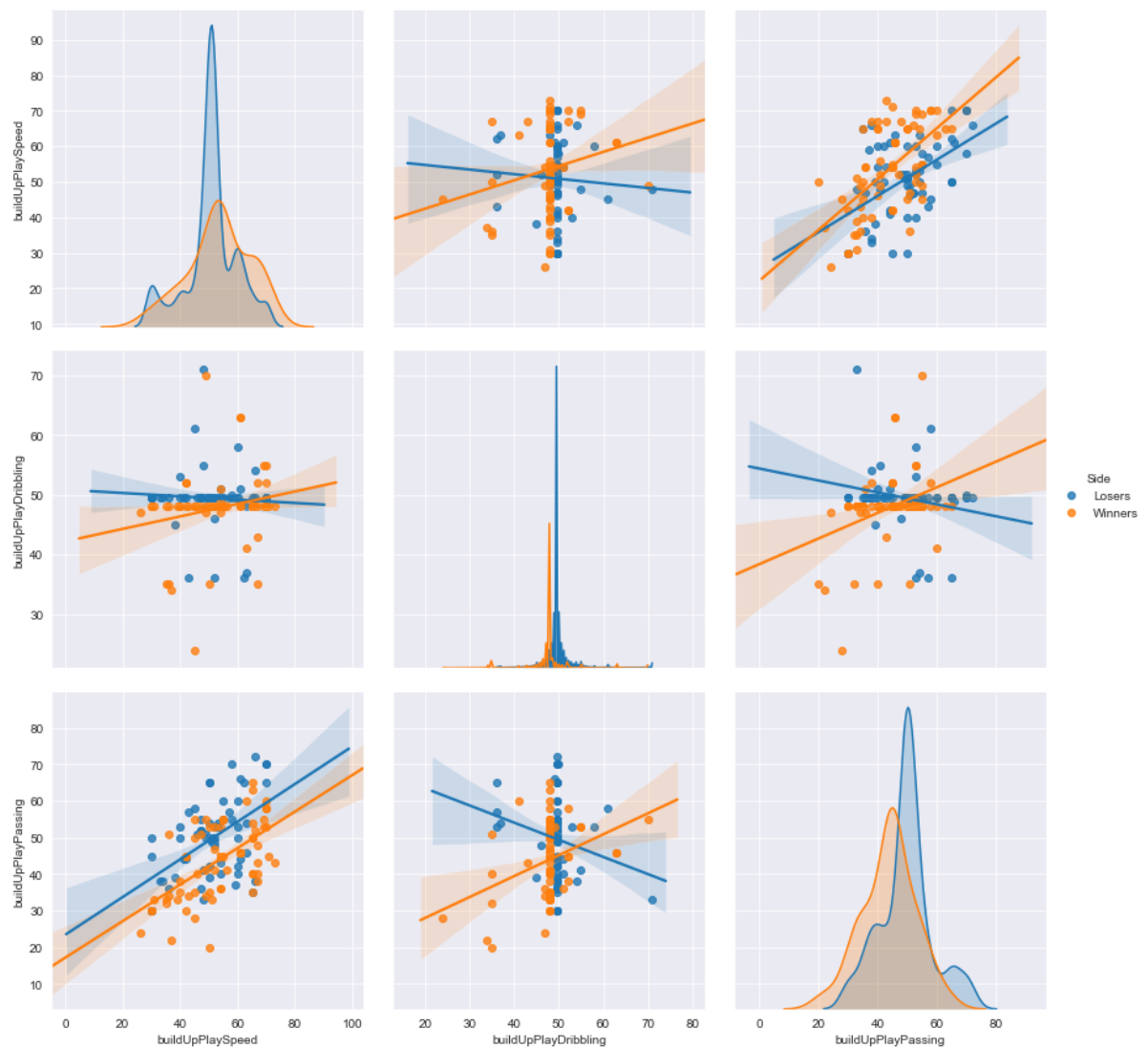
There is only a single combination of categories (Speed and Passing) which both the winning and losing teams have a positive correlation, although the winning team is still stronger. Including that frame, the winning team has positive correlations with every single combination of Build Up statistic.

In [31]: `pair_plot(df2, title2)`

League Winner vs Loser Offense Attributes
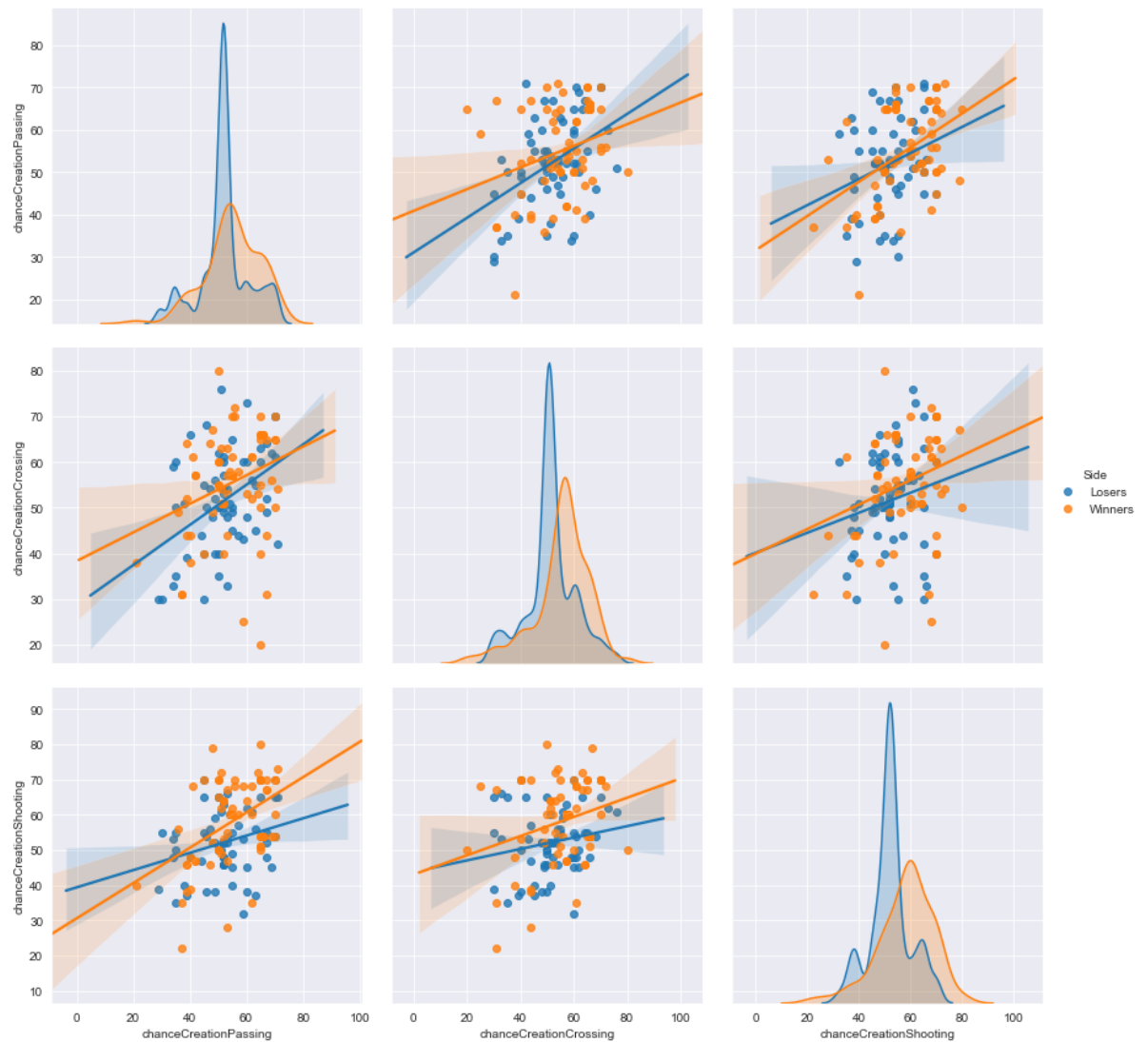
Offensively this comparison was interesting because the losing teams had a more positive correlation in categories like Passing vs Crossing, although the winning teams still maintained a higher average across all teams. This seems to be a consistent trend, the winning team is on average higher than the losing teams, regardless of correlation.

In [32]: `pair_plot(df3, title3)`

League Winner vs Loser Defense Attributes
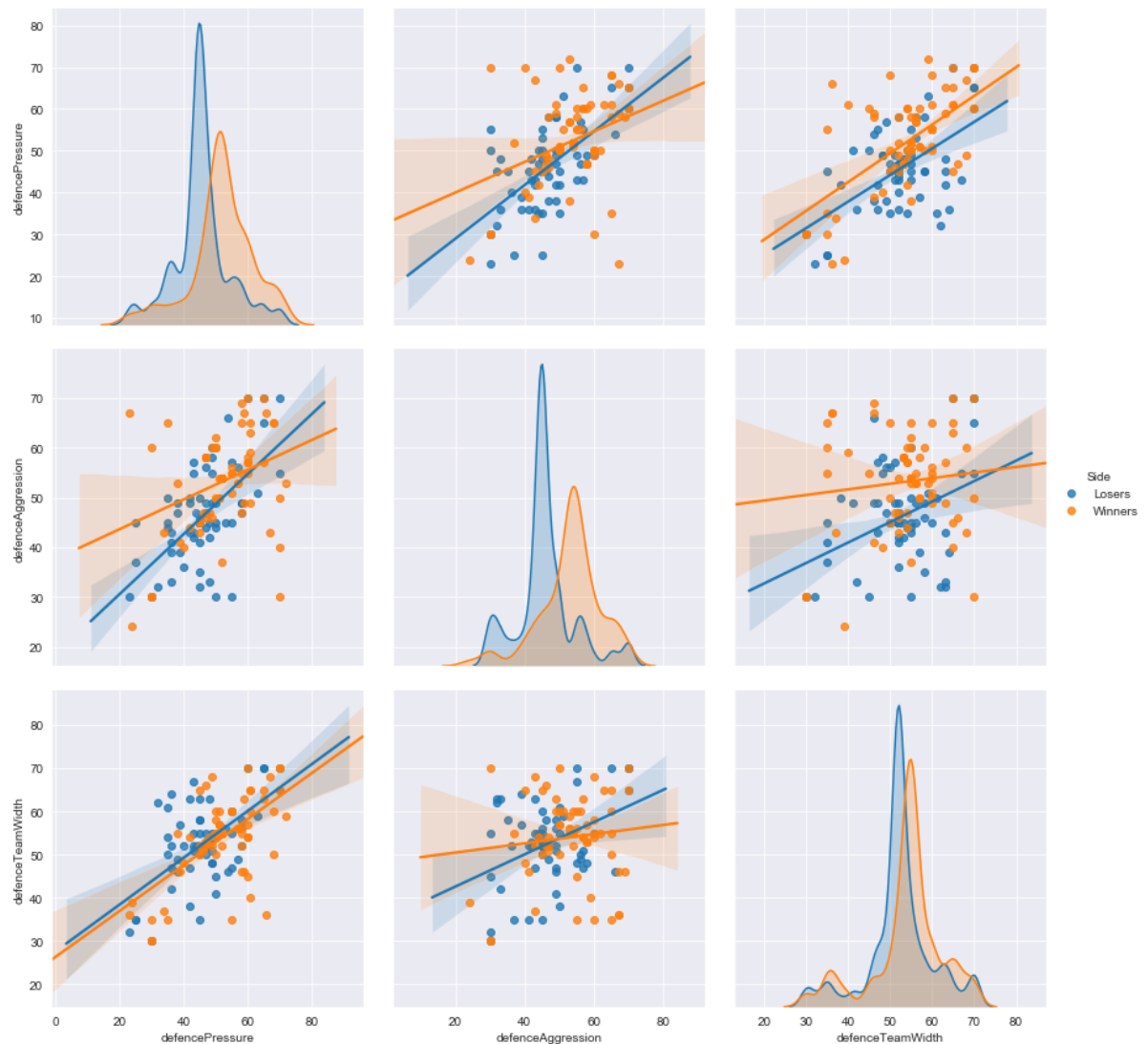
The most interesting stat in this frame for me is the 'Width'. I have no idea what this stat is mean to represent relative to a team's defensive rating but it has a direct correlation to the team's defensive pressure. The winning and losing distributions are nearly identical, and the correlation between defensive pressure and the width for both winning and losing teams are almost parallel. Interestingly similar, again like Offense the losing teams seem to have a tighter correlation between categories, but on average the winning teams have much higher statistics.

Lastly we can see in all three frames that there are a least a couple teams that are on par statistically with the best winning teams in almost every category, yet still managed to lose enough games to get grouped into the worst of the worst (thin high peaks on the losing distribution charts).

Conclusion

We can see that just going by face value wins and losses is not really enough to judge the difference between a first place team and a last place team. In many circumstances there's a lot of grey area, with losing teams ranking in indivudal categories right on par with a winning club, but it's the high overall average - across all categories - that elevates the winning teams.

One particular area which I believe leads to the inevitable higher goal scoring of the winning teams is the higher average and more positive correlation of Build Up Play Speed vs Build Up Play Dribbling and Build Up Play Speed vs Build Up Play Passing. In both comparisons the winning teams did significantly better. I think these stats translate into a much stronger transition game for the winning teams, meaning when they get the ball and are setting up and moving down field, they do this much more quickly and effectively than the other team, leading to more time in the offensive zone.