Caroline Gilhool

2019-2020 NFL Season Data Manipulation & Betting Predictions

Introduction

I will be analyzing 2019-2020 NFL Data in order to compare teams' performance as well as determine whether certain factors, such as game location (home vs away), stadium roof type (fixed, open, or retractable), etc. impacts a team's performance. After collecting this data, I will merge it with NFL betting odds data for the upcoming week's games in order to predictively model the best bets against the spread and over/under lines.

Dimension 1: Data Gathering

I create the following dataframes:

- **stadiums** : I use pandas to read NFL stadium data from Wikipedia into a table. I will use this data to later determine if games were played under fixed, open, or retractable roofs and whether that impacts NFL game results.
- **result**: I do a get request to cbssports.com and use BeautifulSoup to scrape all the NFL team names as well as the URLs to their game results. I then loop through a list of the urls and use pandas to read the game results into an individual table for each team. I then concatenate these tables using the concat function and set the index to be the team name.
- **upcoming_week** : I use pandas to read the current week's upcoming NFL games from CBS. I put games that have yet to happen into a dataframe. For example, if it is a Wednesday the table will contain the upcoming Thursday night, Sunday, and Monday night games. However, if it is a Monday morning, the table will only contain the Monday night game that has not yet occurred.

```
  NOTE: CBS link must be updated to include the current NFL week. EX:
 This week (12/10/19) is WEEK 15.
  Therefore, the link I am using is https://www.cbssports.com/nfl/sche
dule/2019/regular/15/ with 15
  included at the end of this URL. From 12/17-12/23, the link should
  be https://www.cbssports.com/nfl/schedule/2019/regular/16/.
```

- **bets**: I use pandas to read betting odds data for the week from Odds Shark. I collect of all of the teams favored to win, the spread, spread price, O/U, and O/U price into a dataframe called bets that I later merge with result by matching based on if the "Team Favored To Win" in bets is equal to the 'Home' or 'Away' team in result.

Dimension 2: Data Cleaning

result table columns added based on cleaning 'Result' and 'OPP' columns:

- **Outcome** (W, L, or T)
- **Offense** (points scored)
- **Defense** (points allowed)
- **Location** (Home or Away)

result table columns cleaned:

- **OPP** (clean up opponent name in "OPP" column to have all team names be written consistently)

upcoming_week table columns cleaned:

- **Away** (clean up team names in "Away" column to have all team names be written consistently)
- **Home** (clean up team names in "Home" column to have all team names be written consistently)

bets table columns cleaned:

- **Team Favored to Win** (clean up team names in "Team Favored to Win" column to have all team names be written consistently)
- **O/U** (display numeric value)

Dimension 3: Data Manipulation

averages dataframe creation from result dataframe:

- I create an averages dataframe by using group by and data aggregation to include average offense and defensive scores by team. After applying my two statistical functions calc_off_adjusted and calc_def_adjusted to the result dataframe (see Dimension 4: Data Reporting), I update the averages dataframe to include offensive adjusted scores and defensive adjusted scores by team using data aggregation.

result dataframe updated to include information from stadiums dataframe:

- I utilize the function get_roof_type to create the column "Roof Type" based on the away team's stadium data located in the stadiums dataframe.

upcoming_week_predictions dataframe created by merging upcoming_week with bets:

- I perform two merges and append the dataframes together since the bets dataframe only contains one team name ("Team Favored to Win") and it is unclear weather the team favored is playing at home or away:

  1) merge based off upcoming_week['Home'] column and bets['Team']

  2) merge based off upcoming_week['Away'] column and bets['Team']

- This dataframe allows the viewer to make betting predictions for the week based on betting odds and predicted numbers.

result_pivot created from a pivot table of the result dataframe:

- I analyze how many points teams score on average on offense and allow on average on defense based on whether they are playing in an indoor or outdoor stadium (i.e., if roof is fixed, open, or retractable).

records created from a pivot table from result dataframe:

- I create a pivot table depicting how many wins, losses, and ties each NFL team has had.

scores_by_location created from a pivot table from result dataframe:

- I create a pivot table depicting average team scores by location (home or away).

Dimension 4: Data Reporting

I apply multiple functions to my dataframes in order to analyze current team statistics and predict accurate future team statistics and game results.

I apply the following functions to my **result** dataframe in order to demonstrate findings in NFL 2019 season results:

- **calc_off_adjusted**: calculate a team's offensive adjusted score ("Offense_Adj") for each game of the 2019 season by dividing how many points the team scored on offense by their opponent's average defensive points allowed
  - If > 1: team played well on offense
  - If < 1: team played badly on offense
- **calc_def_adjusted**: calculate a team's defensive adjusted score ("Defense_Adj") for each game of the 2019 season by dividing how many points the team allowed on defense by the amount of points their opponent scores on average on offense
  - If > 1: team played badly on defense
  - If < 1: team played well on defense

I apply the following functions to my **upcoming_week** dataframe in order to predict the next week's NFL games results:

- **predict_home**: calculate the home team's predicted points scored

  1) calculate the average amount of points the home team will score on offense by averaging the amount of points the home team scores on average on offense with the average amount of points the away team's defense allows

  2) calculate the home team's average adjusted amount of points on offense by averaging the home team's average amount of points adjusted on offense with the average adjusted amount of points the away team's defense allows

  3) return the product of the average and average adjusted calculations as the home team's final predicted offensive score

- **predict_away**: calculate the away team's predicted points scored

  1) calculate the average amount of points the away team will score on offense by averaging the amount of points the away team scores on average on offense with the average amount of points the home team's defense allows

  2) calculate the away team's average adjusted amount of points on offense by averaging the away team's average amount of points adjusted on offense with the average adjusted amount of points the home team's defense allows

  3) return the product of the average and average adjusted calculations as the away team's final predicted offensive score

- **predict_total_points**: return the sum of the predicted home and away total points scored on offense predictions
- **predict_difference**: return the difference of the predicted home and away total points scored on offense predictions

I create the following data visualizations to clearly depict team performance and smart betting decisions:

- **Vertical bar plots** - analyze data from upcoming_week_predictions table by comparing the absolute value of the predicted difference in teams' scores ("Predicted_Difference") to the absolute value of the

spread ("Spread") as well as the teams' predicted total combined score ("Predicted_Total") to the O/U ("O/U")
    - imported seaborn to create better looking bar plots using seaborn styles
- **Pivot table vertical bar plot** - bar chart of records pivot table
    - imported seaborn to create better looking bar plots using seaborn styles
- **Pivot table line plot** - scores_by_location pivot table
    - imported mathplotlib.pyplot as plt
    - plot initially repeated colors so could not differentiate teams from each other -- used seaborn color palette to import 32 unique colors for every individual NFL team
    - use savefig to save line plot as a png image

Goal

This notebook demonstrates how to create a dataframe that analyzes 2019-2020 NFL season data to date in order to predictively model the upcoming week's NFL games' results and compare predictive game outcomes with betting odds.

Run this notebook to determine which NFL spreads and Over/Under bets are smart to gamble on for the week.

1. Create Dataframes

In [1]:

```
import pandas as pd
import requests
from bs4 import BeautifulSoup
import re
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

Data gathering: stadiums dataframe

In [2]:

```
# creating stadiums dataframe from Wikipedia
stadiums = pd.read_html("https://en.wikipedia.org/wiki/List_of_current_National_
Football_League_stadiums")
stadiums = stadiums[1]

# fixing stadiums to have two individual rows for the New York Giants and New Yo
rk Jets since both teams
# play in the Metlife Stadium and I will later merge off of 'Team(s)'
stadiums = stadiums.append({'Image' : None , 'Name' : "MetLife Stadium", 'Capaci
ty': 82500, 'Location': 'East Rutherford, New Jersey', 'Surface': 'UBU Sports Sp
eed Series S5-M Synthetic Turf[33]', 'Roof type' : 'Open', 'Team(s)': 'New York
 Giants', 'Opened': 2010, 'Ref(s)' : '[34]' } , ignore_index=True)
stadiums = stadiums.append({'Image' : None , 'Name' : "MetLife Stadium", 'Capaci
ty': 82500, 'Location': 'East Rutherford, New Jersey', 'Surface': 'UBU Sports Sp
eed Series S5-M Synthetic Turf[33]', 'Roof type' : 'Open', 'Team(s)': 'New York
 Jets', 'Opened': 2010, 'Ref(s)' : '[34]' } , ignore_index=True)

# stadiums dataframe
stadiums
```

Out[2]:

	Image	Name	Capacity	Location	Surface	Roof type	Team(s)	Op
0	NaN	Arrowhead Stadium	76416	Kansas City, Missouri	Bermuda grass	Open	Kansas City Chiefs	
1	NaN	AT&T Stadium	80000	Arlington, Texas	Hellas Matrix Turf	Retractable	Dallas Cowboys	
2	NaN	Bank of America Stadium	75523	Charlotte, North Carolina	Bermuda grass	Open	Carolina Panthers	
3	NaN	CenturyLink Field	69000	Seattle, Washington	FieldTurf Revolution 360[8]	Open	Seattle Seahawks	
4	NaN	Dignity Health Sports Park	27000	Carson, California	Bermuda grass	Open	Los Angeles Chargers	
5	NaN	Empower Field at Mile High	76125	Denver, Colorado	Kentucky bluegrass	Open	Denver Broncos	
6	NaN	FedExField	82000	Landover, Maryland	Bermuda grass	Open	Washington Redskins	
7	NaN	FirstEnergy Stadium	67895	Cleveland, Ohio	Kentucky bluegrass	Open	Cleveland Browns	
8	NaN	Ford Field	65000	Detroit, Michigan	FieldTurf Classic HD[15]	Fixed	Detroit Lions	
9	NaN	Gillette Stadium	66829	Foxborough, Massachusetts	FieldTurf CORE[17]	Open	New England Patriots	
10	NaN	Hard Rock Stadium	65326	Miami Gardens, Florida	Platinum TE Paspalum	Open	Miami Dolphins	
11	NaN	Heinz Field	68400	Pittsburgh, Pennsylvania	Kentucky bluegrass	Open	Pittsburgh Steelers	
12	NaN	Lambeau Field	81441	Green Bay, Wisconsin	Desso GrassMaster[21]	Open	Green Bay Packers	
13	NaN	Levi's Stadium	68500	Santa Clara, California	Bermuda grass / Perennial Ryegrass mixture	Open	San Francisco 49ers	
14	NaN	Lincoln Financial Field	69596	Philadelphia, Pennsylvania	Desso GrassMaster[24]	Open	Philadelphia Eagles	
15	NaN	Los Angeles Memorial Coliseum	78500	Los Angeles, California	Bermuda grass	Open	Los Angeles Rams	
16	NaN	Lucas Oil Stadium	67000	Indianapolis, Indiana	Shaw Sports Momentum Pro	Retractable	Indianapolis Colts	
17	NaN	M&T Bank Stadium	71008	Baltimore, Maryland	Bermuda grass	Open	Baltimore Ravens	
18	NaN	Mercedes-Benz Superdome	73208	New Orleans, Louisiana	FieldTurf Revolution 360[29]	Fixed	New Orleans Saints	

	Image	Name	Capacity	Location	Surface	Roof type	Team(s)	Op
19	NaN	Mercedes-Benz Stadium	71000	Atlanta, Georgia	FieldTurf Revolution[31]	Retractable	Atlanta Falcons	
20	NaN	MetLife Stadium	82500	East Rutherford, New Jersey	UBU Sports Speed Series S5-M Synthetic Turf[33]	Open	New York GiantsNew York Jets	
21	NaN	New Era Field	71608	Orchard Park, New York	A-Turf Titan 50[35]	Open	Buffalo Bills	
22	NaN	Nissan Stadium	69143	Nashville, Tennessee	Bermuda grass	Open	Tennessee Titans	
23	NaN	NRG Stadium	72220	Houston, Texas	Hellas Matrix Turf[37]	Retractable	Houston Texans	
24	NaN	Paul Brown Stadium	65515	Cincinnati, Ohio	UBU Speed Series S5-M Synthetic Turf	Open	Cincinnati Bengals	
25	NaN	Raymond James Stadium	65890	Tampa, Florida	Bermuda grass	Open	Tampa Bay Buccaneers	
26	NaN	RingCentral Coliseum	56057	Oakland, California	Bermuda grass	Open	Oakland Raiders	
27	NaN	Soldier Field	61500	Chicago, Illinois	Kentucky bluegrass	Open	Chicago Bears	192
28	NaN	State Farm Stadium	63400	Glendale, Arizona	Bermuda grass	Retractable	Arizona Cardinals	
29	NaN	TIAA Bank Field	69132	Jacksonville, Florida	Bermuda grass	Open	Jacksonville Jaguars	
30	NaN	U.S. Bank Stadium	66655	Minneapolis, Minnesota	UBU Speed Series S5-M Synthetic Turf[45]	Fixed	Minnesota Vikings	
31	None	MetLife Stadium	82500	East Rutherford, New Jersey	UBU Sports Speed Series S5-M Synthetic Turf[33]	Open	New York Giants	
32	None	MetLife Stadium	82500	East Rutherford, New Jersey	UBU Sports Speed Series S5-M Synthetic Turf[33]	Open	New York Jets	

Data gathering: result dataframe

In [3]:

```
# building result dataframe to compile all results from 2019-2020 NFL Season so
 far
html = requests.get("https://www.cbssports.com/nfl/teams/").text
soup = BeautifulSoup(html, 'html.parser')
soup

teams =  soup.find_all(class_='TeamName')
teamNames = []
urls = []
for t in teams:
    urls.append(t.a['href'])
    teamNames.append(t.text)

count = 0
l_tables = []
for u in urls:
    start = "https://www.cbssports.com"
    end = "schedule/"
    tables = pd.read_html(start + u + end)
    table = tables[1]
    table['team'] = teamNames[count]
    l_tables.append(table)
    count = count+1

result = pd.concat(l_tables)
result.set_index('team').head(20)
```

Out[3]:

team	Wk	Date	OPP	Result	Record
Buffalo Bills	1	Sep 8, 2019	@ N.Y. Jets	W 17-16	1-0
Buffalo Bills	2	Sep 15, 2019	@ N.Y. Giants	W 28-14	2-0
Buffalo Bills	3	Sep 22, 2019	vs Cincinnati	W 21-17	3-0
Buffalo Bills	4	Sep 29, 2019	vs New England	L 16-10	3-1
Buffalo Bills	5	Oct 6, 2019	@ Tennessee	W 14-7	4-1
Buffalo Bills	6	—	BYE	—	—
Buffalo Bills	7	Oct 20, 2019	vs Miami	W 31-21	5-1
Buffalo Bills	8	Oct 27, 2019	vs Philadelphia	L 31-13	5-2
Buffalo Bills	9	Nov 3, 2019	vs Washington	W 24-9	6-2
Buffalo Bills	10	Nov 10, 2019	@ Cleveland	L 19-16	6-3
Buffalo Bills	11	Nov 17, 2019	@ Miami	W 37-20	7-3
Buffalo Bills	12	Nov 24, 2019	vs Denver	W 20-3	8-3
Buffalo Bills	13	Nov 28, 2019	@ Dallas	W 26-15	9-3
Buffalo Bills	14	Dec 8, 2019	vs Baltimore	L 24-17	9-4
Miami Dolphins	1	Sep 8, 2019	vs Baltimore	L 59-10	0-1
Miami Dolphins	2	Sep 15, 2019	vs New England	L 43-0	0-2
Miami Dolphins	3	Sep 22, 2019	@ Dallas	L 31-6	0-3
Miami Dolphins	4	Sep 29, 2019	vs L.A. Chargers	L 30-10	0-4
Miami Dolphins	5	—	BYE	—	—
Miami Dolphins	6	Oct 13, 2019	vs Washington	L 17-16	0-5

Data gathering: upcoming_week dataframe

IMPORTANT NOTE: CBS link must be updated to include the current NFL week. EX: This week (12/10/19) is WEEK 15. Therefore, the link I am using is https://www.cbssports.com/nfl/schedule/2019/regular/15/ (https://www.cbssports.com/nfl/schedule/2019/regular/15/) with 15 included at the end of this URL. From 12/17 - 12/23, the link should be https://www.cbssports.com/nfl/schedule/2019/regular/16/ (https://www.cbssports.com/nfl/schedule/2019/regular/16/).

In [4]:

```
# create upcoming_week dataframe - there are Thursday, Sunday, and Monday night
 games but I just
# want to look at the week's games that have not yet occured in order to make pr
edictions

#

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

# IMPORTANT NOTE: CBS link must be updated to include the current NFL week. EX:
 This week (12/10/19) is WEEK 15.
# Therefore, the link I am using is https://www.cbssports.com/nfl/schedule/2019/
regular/15/ with 15
# included at the end of this URL. From 12/17 - 12/23, the link should
# be https://www.cbssports.com/nfl/schedule/2019/regular/16/.
#

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


nfl_games = pd.read_html("https://www.cbssports.com/nfl/schedule/2019/regular/1
5/")
next_games = []
for n in nfl_games:
    if 'Result' in n.columns:
        pass
    else:
        next_games.append(n)

upcoming_week = pd.concat(next_games)
upcoming_week
```

Out[4]:

	Away	Home	Time	TV	Venue	Buy Tickets
0	N.Y. Jets	Baltimore	8:20 pm	NFLN	M&T Bank Stadium	Tickets Starting at $48.00
0	Miami	N.Y. Giants	1:00 pm	NaN	MetLife Stadium	Tickets Starting at $24.29
1	Houston	Tennessee	1:00 pm	NaN	Nissan Stadium	Tickets Starting at $55.48
2	New England	Cincinnati	1:00 pm	NaN	Paul Brown Stadium	Tickets Starting at $51.03
3	Seattle	Carolina	1:00 pm	FOX	Bank of America Stadium	Tickets Starting at $40.00
4	Tampa Bay	Detroit	1:00 pm	FOX	Ford Field	Tickets Starting at $32.00
5	Denver	Kansas City	1:00 pm	NaN	Arrowhead Stadium	Tickets Starting at $48.00
6	Philadelphia	Washington	1:00 pm	FOX	FedEx Field	Tickets Starting at $40.00
7	Chicago	Green Bay	1:00 pm	FOX	Lambeau Field	Tickets Starting at $121.00
8	Jacksonville	Oakland	4:05 pm	NaN	RingCentral Coliseum	Tickets Starting at $152.52
9	Cleveland	Arizona	4:05 pm	NaN	State Farm Stadium	Tickets Starting at $60.00
10	Minnesota	L.A. Chargers	4:05 pm	NaN	Dignity Health Sports Park	Tickets Starting at $209.95
11	Atlanta	San Francisco	4:25 pm	FOX	Levi's Stadium	Tickets Starting at $95.00
12	L.A. Rams	Dallas	4:25 pm	FOX	AT&T Stadium	Tickets Starting at $39.00
13	Buffalo	Pittsburgh	8:20 pm	NBC	Heinz Field	Tickets Starting at $94.00
0	Indianapolis	New Orleans	8:15 pm	ESPN	Mercedes-Benz Superdome	Tickets Starting at $73.99

Data gathering: bets dataframe

In [5]:

```
# bets dataframe: betting_odds from Odds Shark
betting_odds = pd.read_html("https://www.oddsshark.com/nfl/consensus-picks")
odds = pd.DataFrame()
col = ['Unnamed: 0', 'ATS Consensus', 'ATS Consensus.1','Spread','Price','O/U Co
nsensus','O/U Consensus.1','Price.1']
odds = odds.reindex(columns=col)
all_odds = []

for b in betting_odds:
    if (b['Spread'][0] == 'Ev') == True:
        odds = (b.iloc[[0]])
        all_odds.append(odds)
    else:
        b['Spread'] = pd.to_numeric(b['Spread'])
        if b['Spread'][0] < 0:
            odds = (b.iloc[[0]])
            all_odds.append(odds)
        else:
            odds = (b.iloc[[1]])
            all_odds.append(odds)

bets = pd.concat(all_odds)
bets
```

Out[5]:

	Unnamed: 0	ATS Consensus	ATS Consensus.1	Spread	Price	O/U Consensus	O/U Consensus.1	Price.1
1	NaN	BAL Baltimore	54%	-17.0	-110	O/U44	48%	-110
1	NaN	GB Green Bay	51%	-4.0	-110	O/U40.5	37%	-110
1	NaN	KC Kansas City	53%	-10.0	-105	O/U45	32%	-110
1	NaN	TEN Tennessee	46%	-3.0	-120	O/U51	37%	-110
0	Matchup	SEA Seattle	69%	-6.5	-110	O/U49	52%	-110
0	Matchup	NE New England	64%	-10.0	-110	O/U41.5	60%	-115
0	Matchup	TB Tampa Bay	56%	-3.5	-110	O/U46	68%	-110
1	NaN	NYG NY Giants	46%	-3.5	-105	O/U46.5	56%	-110
0	Matchup	PHI Philadelphia	57%	-5.0	-110	O/U39	53%	-110
1	NaN	OAK Oakland	59%	-6.5	-110	O/U45.5	35%	-110
0	Matchup	CLE Cleveland	60%	-2.5	-110	O/U49	67%	-110
0	Matchup	MIN Minnesota	61%	-2.5	-110	O/U45.5	62%	-110
1	NaN	SF San Francisco	55%	-10.5	-110	O/U48	30%	-110
0	Matchup	LAR LA Rams	66%	-1.5	-110	O/U49	55%	-110
1	NaN	PIT Pittsburgh	34%	-1.5	-110	O/U35.5	37%	-110
1	NaN	NO New Orleans	57%	-9.0	-110	O/U46.5	36%	-110

2. Clean Dataframes

result dataframe

I write my own functions and use .apply to apply them to the dataframe as well as use .str functions to clean the dataframe.

The 'Result" column of result contains a string like "W 17-16" which must be separated into "Outcome" (W, L, or T), "Offense" (offensive points scored), and "Defense" (defensive points allowed).

In [6]:

```
# Outcome column
result['Outcome'] = result['Result'].str.extract(r'(\w)')
result.set_index('team').head(20)
```

Out[6]:

team	Wk	Date	OPP	Result	Record	Outcome
Buffalo Bills	1	Sep 8, 2019	@ N.Y. Jets	W 17-16	1-0	W
Buffalo Bills	2	Sep 15, 2019	@ N.Y. Giants	W 28-14	2-0	W
Buffalo Bills	3	Sep 22, 2019	vs Cincinnati	W 21-17	3-0	W
Buffalo Bills	4	Sep 29, 2019	vs New England	L 16-10	3-1	L
Buffalo Bills	5	Oct 6, 2019	@ Tennessee	W 14-7	4-1	W
Buffalo Bills	6	—	BYE	—	—	NaN
Buffalo Bills	7	Oct 20, 2019	vs Miami	W 31-21	5-1	W
Buffalo Bills	8	Oct 27, 2019	vs Philadelphia	L 31-13	5-2	L
Buffalo Bills	9	Nov 3, 2019	vs Washington	W 24-9	6-2	W
Buffalo Bills	10	Nov 10, 2019	@ Cleveland	L 19-16	6-3	L
Buffalo Bills	11	Nov 17, 2019	@ Miami	W 37-20	7-3	W
Buffalo Bills	12	Nov 24, 2019	vs Denver	W 20-3	8-3	W
Buffalo Bills	13	Nov 28, 2019	@ Dallas	W 26-15	9-3	W
Buffalo Bills	14	Dec 8, 2019	vs Baltimore	L 24-17	9-4	L
Miami Dolphins	1	Sep 8, 2019	vs Baltimore	L 59-10	0-1	L
Miami Dolphins	2	Sep 15, 2019	vs New England	L 43-0	0-2	L
Miami Dolphins	3	Sep 22, 2019	@ Dallas	L 31-6	0-3	L
Miami Dolphins	4	Sep 29, 2019	vs L.A. Chargers	L 30-10	0-4	L
Miami Dolphins	5	—	BYE	—	—	NaN
Miami Dolphins	6	Oct 13, 2019	vs Washington	L 17-16	0-5	L

In [7]:

```
# Offense column

def get_offense(x):
    if (x['Outcome'] == 'W') == True:
        pos = re.search(r'\d+', x['Result']).group()
        return pos
    if (x['Outcome'] == 'L') == True:
        pos = re.search(r'\-+\d+',  x['Result']).group()
        return pos
    if (x['Outcome'] == 'T') == True:
        pos = re.search(r'\d+', x['Result']).group()
        return pos
    else:
        return

result['Offense'] = result.apply(get_offense, axis=1)
result['Offense'] = result['Offense'].str.replace(r'\-', '')

# Defense column

def get_defense(x):
    if (x['Outcome'] == 'W') == True:
        pos = re.search(r'\-+\d+',  x['Result']).group()
        return pos
    if (x['Outcome'] == 'L') == True:
        pos = re.search(r'\d+', x['Result']).group()
        return pos
    if (x['Outcome'] == 'T') == True:
        pos = re.search(r'\d+', x['Result']).group()
        return pos
    else:
        return

result['Defense'] = result.apply(get_defense, axis=1)
result['Defense'] = result['Defense'].str.replace(r'\-', '')

# Location column

def get_location(x):
    res = re.search(r'^[@]', x['OPP'])
    if res:
        pos = 'Away'
        return pos
    else:
        res2 = re.search(r'^[vs]', x['OPP'])
        if res2:
            pos = 'Home'
            return pos
        else:
            return

result['Location'] = result.apply(get_location, axis=1)
```

The "OPP" column of result contains a string like "@ N.Y. Jets" or "vs Cincinnati" which must be used to determine a "Location" column displaying whether a game was home or away.

The "OPP" column does not use the same syntax to display teams' names as the "team" column. To correctly match on a team's name later, I create a dictionary (dic) that maps names from the "team" column to "OPP" column in order to replace the names in the "OPP" column to have the same syntax.

In [8]:

```
# clean OPP names

teamNames.sort()
result['OPP'] = result['OPP'].str.replace(r'\@ ', '')
result['OPP'] = result['OPP'].str.replace(r'^vs ', '')
otherTeamNames = result['OPP'].unique()
teamArray =[]

for o in otherTeamNames:
    # most values aligned but need to manually change certain values
    if o == "BYE":
        pass
    else:
        teamArray.append(o)

teamArray.sort()

dic = {teamArray[i]: teamNames[i] for i in range(len(teamArray))}
for key,value in dic.items():
    if key == 'N.Y. Giants':
        dic[key] = 'New York Giants'
    elif key == 'N.Y. Jets':
        dic[key] = 'New York Jets'
    elif key == 'New England':
        dic[key] = 'New England Patriots'
    elif key == 'New Orleans':
        dic[key] = 'New Orleans Saints'
dic
```

Out[8]:

```
{'Arizona': 'Arizona Cardinals',
 'Atlanta': 'Atlanta Falcons',
 'Baltimore': 'Baltimore Ravens',
 'Buffalo': 'Buffalo Bills',
 'Carolina': 'Carolina Panthers',
 'Chicago': 'Chicago Bears',
 'Cincinnati': 'Cincinnati Bengals',
 'Cleveland': 'Cleveland Browns',
 'Dallas': 'Dallas Cowboys',
 'Denver': 'Denver Broncos',
 'Detroit': 'Detroit Lions',
 'Green Bay': 'Green Bay Packers',
 'Houston': 'Houston Texans',
 'Indianapolis': 'Indianapolis Colts',
 'Jacksonville': 'Jacksonville Jaguars',
 'Kansas City': 'Kansas City Chiefs',
 'L.A. Chargers': 'Los Angeles Chargers',
 'L.A. Rams': 'Los Angeles Rams',
 'Miami': 'Miami Dolphins',
 'Minnesota': 'Minnesota Vikings',
 'N.Y. Giants': 'New York Giants',
 'N.Y. Jets': 'New York Jets',
 'New England': 'New England Patriots',
 'New Orleans': 'New Orleans Saints',
 'Oakland': 'Oakland Raiders',
 'Philadelphia': 'Philadelphia Eagles',
 'Pittsburgh': 'Pittsburgh Steelers',
 'San Francisco': 'San Francisco 49ers',
 'Seattle': 'Seattle Seahawks',
 'Tampa Bay': 'Tampa Bay Buccaneers',
 'Tennessee': 'Tennessee Titans',
 'Washington': 'Washington Redskins'}
```

In [9]:

```
# use dic in clean_opp function

def clean_opp(x):
    if x['OPP'] == 'BYE':
        return
    else:
        return dic[x["OPP"]]

result['OPP'] = result.apply(clean_opp, axis=1)

result.set_index('team').head(20)
```

Out[9]:

team	Wk	Date	OPP	Result	Record	Outcome	Offense	Defense	Location
Buffalo Bills	1	Sep 8, 2019	New York Jets	W 17-16	1-0	W	17	16	Away
Buffalo Bills	2	Sep 15, 2019	New York Giants	W 28-14	2-0	W	28	14	Away
Buffalo Bills	3	Sep 22, 2019	Cincinnati Bengals	W 21-17	3-0	W	21	17	Home
Buffalo Bills	4	Sep 29, 2019	New England Patriots	L 16-10	3-1	L	10	16	Home
Buffalo Bills	5	Oct 6, 2019	Tennessee Titans	W 14-7	4-1	W	14	7	Away
Buffalo Bills	6	—	None	—	—	NaN	None	None	None
Buffalo Bills	7	Oct 20, 2019	Miami Dolphins	W 31-21	5-1	W	31	21	Home
Buffalo Bills	8	Oct 27, 2019	Philadelphia Eagles	L 31-13	5-2	L	13	31	Home
Buffalo Bills	9	Nov 3, 2019	Washington Redskins	W 24-9	6-2	W	24	9	Home
Buffalo Bills	10	Nov 10, 2019	Cleveland Browns	L 19-16	6-3	L	16	19	Away
Buffalo Bills	11	Nov 17, 2019	Miami Dolphins	W 37-20	7-3	W	37	20	Away
Buffalo Bills	12	Nov 24, 2019	Denver Broncos	W 20-3	8-3	W	20	3	Home
Buffalo Bills	13	Nov 28, 2019	Dallas Cowboys	W 26-15	9-3	W	26	15	Away
Buffalo Bills	14	Dec 8, 2019	Baltimore Ravens	L 24-17	9-4	L	17	24	Home
Miami Dolphins	1	Sep 8, 2019	Baltimore Ravens	L 59-10	0-1	L	10	59	Home
Miami Dolphins	2	Sep 15, 2019	New England Patriots	L 43-0	0-2	L	0	43	Home
Miami Dolphins	3	Sep 22, 2019	Dallas Cowboys	L 31-6	0-3	L	6	31	Away
Miami Dolphins	4	Sep 29, 2019	Los Angeles Chargers	L 30-10	0-4	L	10	30	Home
Miami Dolphins	5	—	None	—	—	NaN	None	None	None

team	Wk	Date	OPP	Result	Record	Outcome	Offense	Defense	Location
Miami Dolphins	6	Oct 13, 2019	Washington Redskins	L 17-16	0-5	L	16	17	Home

upcoming_week dataframe

The upcoming_week dataframe must display "Away" and "Home" teams with consistent syntax to the "team" column in the result dataframe. I use the same dictionary (dic) to replace the team names in upcoming_week.

In [10]:

```
# clean upcoming_week "Home" and "Away"
def clean_home(x):
    return dic[x["Home"]]

def clean_away(x):
    return dic[x["Away"]]

upcoming_week['Home'] = upcoming_week.apply(clean_home, axis=1)
upcoming_week['Away'] = upcoming_week.apply(clean_away, axis=1)
upcoming_week
```

Out[10]:

	Away	Home	Time	TV	Venue	Buy Tickets
0	New York Jets	Baltimore Ravens	8:20 pm	NFLN	M&T Bank Stadium	Tickets Starting at $48.00
0	Miami Dolphins	New York Giants	1:00 pm	NaN	MetLife Stadium	Tickets Starting at $24.29
1	Houston Texans	Tennessee Titans	1:00 pm	NaN	Nissan Stadium	Tickets Starting at $55.48
2	New England Patriots	Cincinnati Bengals	1:00 pm	NaN	Paul Brown Stadium	Tickets Starting at $51.03
3	Seattle Seahawks	Carolina Panthers	1:00 pm	FOX	Bank of America Stadium	Tickets Starting at $40.00
4	Tampa Bay Buccaneers	Detroit Lions	1:00 pm	FOX	Ford Field	Tickets Starting at $32.00
5	Denver Broncos	Kansas City Chiefs	1:00 pm	NaN	Arrowhead Stadium	Tickets Starting at $48.00
6	Philadelphia Eagles	Washington Redskins	1:00 pm	FOX	FedEx Field	Tickets Starting at $40.00
7	Chicago Bears	Green Bay Packers	1:00 pm	FOX	Lambeau Field	Tickets Starting at $121.00
8	Jacksonville Jaguars	Oakland Raiders	4:05 pm	NaN	RingCentral Coliseum	Tickets Starting at $152.52
9	Cleveland Browns	Arizona Cardinals	4:05 pm	NaN	State Farm Stadium	Tickets Starting at $60.00
10	Minnesota Vikings	Los Angeles Chargers	4:05 pm	NaN	Dignity Health Sports Park	Tickets Starting at $209.95
11	Atlanta Falcons	San Francisco 49ers	4:25 pm	FOX	Levi's Stadium	Tickets Starting at $95.00
12	Los Angeles Rams	Dallas Cowboys	4:25 pm	FOX	AT&T Stadium	Tickets Starting at $39.00
13	Buffalo Bills	Pittsburgh Steelers	8:20 pm	NBC	Heinz Field	Tickets Starting at $94.00
0	Indianapolis Colts	New Orleans Saints	8:15 pm	ESPN	Mercedes-Benz Superdome	Tickets Starting at $73.99

In [11]:

```
# minimize table to only include Home and Away team
upcoming_week = upcoming_week[['Home', 'Away']]
upcoming_week
```

Out[11]:

	Home	Away
0	Baltimore Ravens	New York Jets
0	New York Giants	Miami Dolphins
1	Tennessee Titans	Houston Texans
2	Cincinnati Bengals	New England Patriots
3	Carolina Panthers	Seattle Seahawks
4	Detroit Lions	Tampa Bay Buccaneers
5	Kansas City Chiefs	Denver Broncos
6	Washington Redskins	Philadelphia Eagles
7	Green Bay Packers	Chicago Bears
8	Oakland Raiders	Jacksonville Jaguars
9	Arizona Cardinals	Cleveland Browns
10	Los Angeles Chargers	Minnesota Vikings
11	San Francisco 49ers	Atlanta Falcons
12	Dallas Cowboys	Los Angeles Rams
13	Pittsburgh Steelers	Buffalo Bills
0	New Orleans Saints	Indianapolis Colts

bets dataframe

The bets dataframe contains columns that are not very indicative of their contents. I rename the bets dataframe to have the following columns: Team Favored to Win, Spread, Spread Price, O/U, O/U Price.

I create dic2 to replace the teams in the "Team Favored Win" column with the same syntax as "teams" in the result dataframe column.

In [12]:

```
# start cleaning bets
# map keys to team names  to clean bets "Teams Favored to Win"

dic2 = {
    'AZ': 'Arizona Cardinals',
    'ATL': 'Atlanta Falcons',
    'BAL': 'Baltimore Ravens',
    'BUFF': 'Buffalo Bills',
    'CAR': 'Carolina Panthers',
    'CHI': 'Chicago Bears',
    'CIN': 'Cincinnati Bengals',
    'CLE': 'Cleveland Browns',
    'DAL': 'Dallas Cowboys',
    'DEN': 'Denver Broncos',
    'DET': 'Detroit Lions',
    'GB': 'Green Bay Packers',
    'HOU': 'Houston Texans',
    'IND': 'Indianapolis Colts',
    'JAC': 'Jacksonville Jaguars',
    'KC': 'Kansas City Chiefs',
    'LAC': 'Los Angeles Chargers',
    'LAR': 'Los Angeles Rams',
    'MIA': 'Miami Dolphins',
    'MIN': 'Minnesota Vikings',
    'NYG': 'New York Giants',
    'NYJ': 'New York Jets',
    'NE': 'New England Patriots',
    'NO': 'New Orleans Saints',
    'OAK': 'Oakland Raiders',
    'PHI': 'Philadelphia Eagles',
    'PIT': 'Pittsburgh Steelers',
    'SF': 'San Francisco 49ers',
    'SEA': 'Seattle Seahawks',
    'TB': 'Tampa Bay Buccaneers',
    'TEN': 'Tennessee Titans',
    'WAS': 'Washington Redskins'
}
```

In [13]:

```
# cleans bets "Team Favored to Win" using dic2
bets = bets[['ATS Consensus', 'Spread', 'Price', 'O/U Consensus', 'Price.1']]

bets = bets.rename(columns={"ATS Consensus": "Team Favored to Win", "Price": "Sp
read Price", "O/U Consensus" : "O/U", "Price.1" : "O/U Price"})

bets["Team Favored to Win"] = bets["Team Favored to Win"].str.extract(r'(\w+)')

def clean_team(x):
    return dic2[x["Team Favored to Win"]]

bets["Team Favored to Win"] = bets.apply(clean_team, axis=1)
bets
```

Out[13]:

	Team Favored to Win	Spread	Spread Price	O/U	O/U Price
1	Baltimore Ravens	-17.0	-110	O/U44	-110
1	Green Bay Packers	-4.0	-110	O/U40.5	-110
1	Kansas City Chiefs	-10.0	-105	O/U45	-110
1	Tennessee Titans	-3.0	-120	O/U51	-110
0	Seattle Seahawks	-6.5	-110	O/U49	-110
0	New England Patriots	-10.0	-110	O/U41.5	-115
0	Tampa Bay Buccaneers	-3.5	-110	O/U46	-110
1	New York Giants	-3.5	-105	O/U46.5	-110
0	Philadelphia Eagles	-5.0	-110	O/U39	-110
1	Oakland Raiders	-6.5	-110	O/U45.5	-110
0	Cleveland Browns	-2.5	-110	O/U49	-110
0	Minnesota Vikings	-2.5	-110	O/U45.5	-110
1	San Francisco 49ers	-10.5	-110	O/U48	-110
0	Los Angeles Rams	-1.5	-110	O/U49	-110
1	Pittsburgh Steelers	-1.5	-110	O/U35.5	-110
1	New Orleans Saints	-9.0	-110	O/U46.5	-110

I clean the bets "O/U" column by changing values like "O/U41" through extracting the numeric value-- 41.

In [14]:

```
# clean bets "O/U"
bets["O/U"] = bets["O/U"].str.extract(r'(\d+[.\d]+)')
bets
```

Out[14]:

	Team Favored to Win	Spread	Spread Price	O/U	O/U Price
1	Baltimore Ravens	-17.0	-110	44	-110
1	Green Bay Packers	-4.0	-110	40.5	-110
1	Kansas City Chiefs	-10.0	-105	45	-110
1	Tennessee Titans	-3.0	-120	51	-110
0	Seattle Seahawks	-6.5	-110	49	-110
0	New England Patriots	-10.0	-110	41.5	-115
0	Tampa Bay Buccaneers	-3.5	-110	46	-110
1	New York Giants	-3.5	-105	46.5	-110
0	Philadelphia Eagles	-5.0	-110	39	-110
1	Oakland Raiders	-6.5	-110	45.5	-110
0	Cleveland Browns	-2.5	-110	49	-110
0	Minnesota Vikings	-2.5	-110	45.5	-110
1	San Francisco 49ers	-10.5	-110	48	-110
0	Los Angeles Rams	-1.5	-110	49	-110
1	Pittsburgh Steelers	-1.5	-110	35.5	-110
1	New Orleans Saints	-9.0	-110	46.5	-110

3. Create a Predictive Model

I calculate the average amount of points each team scores on offense and average amount of points each team allows on defense.

In [15]:

```
result['Offense'] =pd.to_numeric(result['Offense'])
result['Defense'] =pd.to_numeric(result['Defense'])
averages = result.groupby("team").agg({'Offense': np.nanmean, 'Defense': np.nanm
ean})
averages
```

Out[15]:

team	Offense	Defense
Arizona Cardinals	20.923077	28.769231
Atlanta Falcons	23.076923	26.384615
Baltimore Ravens	33.076923	18.153846
Buffalo Bills	21.076923	16.307692
Carolina Panthers	23.076923	27.692308
Chicago Bears	18.692308	17.846154
Cincinnati Bengals	15.230769	25.000000
Cleveland Browns	21.000000	22.384615
Dallas Cowboys	25.692308	20.538462
Denver Broncos	18.153846	20.076923
Detroit Lions	22.076923	25.769231
Green Bay Packers	23.769231	20.769231
Houston Texans	24.384615	23.769231
Indianapolis Colts	22.769231	22.692308
Jacksonville Jaguars	17.692308	25.923077
Kansas City Chiefs	28.538462	21.615385
Los Angeles Chargers	22.230769	19.307692
Los Angeles Rams	23.923077	20.153846
Miami Dolphins	17.000000	30.692308
Minnesota Vikings	26.076923	19.153846
New England Patriots	26.000000	12.923077
New Orleans Saints	26.461538	22.769231
New York Giants	19.000000	27.846154
New York Jets	17.384615	23.153846
Oakland Raiders	19.846154	28.153846
Philadelphia Eagles	22.846154	23.153846
Pittsburgh Steelers	19.923077	18.615385
San Francisco 49ers	30.538462	17.615385
Seattle Seahawks	26.230769	24.692308
Tampa Bay Buccaneers	29.076923	29.307692
Tennessee Titans	24.461538	19.615385
Washington Redskins	14.461538	23.846154

I calculate offensive and defensive adjusted Scores (see full description of functions in Dimension 4).

In [16]:

```
# this function divides the offensive team's points scored (x['Offense'])
# by the average amount of points the opposing team gives up on defense
# this creates an "adjusted score" that proves a team played well on offense if
 the result is over 1 and badly if less than 1
def calc_off_adjusted(x):
    if x['OPP']:
        off_score = x['Offense']
        opp_avg_defense = averages.loc[x['OPP'], 'Defense']
        return (off_score / opp_avg_defense)
    else:
        return

result['Offense_Adj'] = result.apply(calc_off_adjusted, axis=1)

# this function divides the number of points the team gave up on defense
# by the average amount of points the opposing team's offense typically scores
# this creates an "adjusted score" that proves a team's defense played badly if
 greater than 1 or well if less than 1
def calc_def_adjusted(x):
    if x['OPP']:
        def_score = x['Defense']
        opp_avg_off = averages.loc[x['OPP'], 'Offense']
        return (def_score / opp_avg_off)
    else:
        return

result['Defense_Adj'] = result.apply(calc_def_adjusted, axis=1)

result.set_index('team').loc['New York Giants']
```

Out[16]:

team	Wk	Date	OPP	Result	Record	Outcome	Offense	Defense	Location	Offens
New York Giants	1	Sep 8, 2019	Dallas Cowboys	L 35-17	0-1	L	17.0	35.0	Away	0.8
New York Giants	2	Sep 15, 2019	Buffalo Bills	L 28-14	0-2	L	14.0	28.0	Home	0.8
New York Giants	3	Sep 22, 2019	Tampa Bay Buccaneers	W 32-31	1-2	W	32.0	31.0	Away	1.0
New York Giants	4	Sep 29, 2019	Washington Redskins	W 24-3	2-2	W	24.0	3.0	Home	1.0
New York Giants	5	Oct 6, 2019	Minnesota Vikings	L 28-10	2-3	L	10.0	28.0	Home	0.5
New York Giants	6	Oct 10, 2019	New England Patriots	L 35-14	2-4	L	14.0	35.0	Away	1.0
New York Giants	7	Oct 20, 2019	Arizona Cardinals	L 27-21	2-5	L	21.0	27.0	Home	0.7
New York Giants	8	Oct 27, 2019	Detroit Lions	L 31-26	2-6	L	26.0	31.0	Away	1.0
New York Giants	9	Nov 4, 2019	Dallas Cowboys	L 37-18	2-7	L	18.0	37.0	Home	0.8
New York Giants	10	Nov 10, 2019	New York Jets	L 34-27	2-8	L	27.0	34.0	Away	1.1
New York Giants	11	—	None	—	—	NaN	NaN	NaN	None	
New York Giants	12	Nov 24, 2019	Chicago Bears	L 19-14	2-9	L	14.0	19.0	Away	0.7
New York Giants	13	Dec 1, 2019	Green Bay Packers	L 31-13	2-10	L	13.0	31.0	Home	0.6
New York Giants	14	Dec 9, 2019	Philadelphia Eagles	L 23-17 / OT	2-11	L	17.0	23.0	Away	0.7

I query roof type from stadiums dataframe into result via the following method:

- Look at Location column to determine if the game is being played home or away
  - If home, get stadium for team in ‘team’ column
  - If away, get stadium for team in ‘OPP’ column

In [17]:

```
stadiums = stadiums.set_index('Team(s)')
def get_roof_type(x):
    # get opponent stadium roof type
    if x['Location'] == 'Away':
        return stadiums.loc[x['OPP'], 'Roof type']
    # get team stadium roof type
    else:
        return stadiums.loc[x['team'], 'Roof type']

result['Roof Type'] = result.apply(get_roof_type, axis=1)
result.set_index('team').loc['New York Giants']
```

Out[17]:

team	Wk	Date	OPP	Result	Record	Outcome	Offense	Defense	Location	Offens
New York Giants	1	Sep 8, 2019	Dallas Cowboys	L 35-17	0-1	L	17.0	35.0	Away	0.8
New York Giants	2	Sep 15, 2019	Buffalo Bills	L 28-14	0-2	L	14.0	28.0	Home	0.8
New York Giants	3	Sep 22, 2019	Tampa Bay Buccaneers	W 32-31	1-2	W	32.0	31.0	Away	1.0
New York Giants	4	Sep 29, 2019	Washington Redskins	W 24-3	2-2	W	24.0	3.0	Home	1.0
New York Giants	5	Oct 6, 2019	Minnesota Vikings	L 28-10	2-3	L	10.0	28.0	Home	0.5
New York Giants	6	Oct 10, 2019	New England Patriots	L 35-14	2-4	L	14.0	35.0	Away	1.0
New York Giants	7	Oct 20, 2019	Arizona Cardinals	L 27-21	2-5	L	21.0	27.0	Home	0.7
New York Giants	8	Oct 27, 2019	Detroit Lions	L 31-26	2-6	L	26.0	31.0	Away	1.0
New York Giants	9	Nov 4, 2019	Dallas Cowboys	L 37-18	2-7	L	18.0	37.0	Home	0.8
New York Giants	10	Nov 10, 2019	New York Jets	L 34-27	2-8	L	27.0	34.0	Away	1.1
New York Giants	11	—	None	—	—	NaN	NaN	NaN	None	
New York Giants	12	Nov 24, 2019	Chicago Bears	L 19-14	2-9	L	14.0	19.0	Away	0.7
New York Giants	13	Dec 1, 2019	Green Bay Packers	L 31-13	2-10	L	13.0	31.0	Home	0.6
New York Giants	14	Dec 9, 2019	Philadelphia Eagles	L 23-17 / OT	2-11	L	17.0	23.0	Away	0.7

In [18]:

```
averages = result.groupby("team").agg({'Offense': np.nanmean, 'Defense': np.nanm
ean, 'Offense_Adj': np.nanmean, 'Defense_Adj': np.nanmean})
averages
```

Out[18]:

team	Offense	Defense	Offense_Adj	Defense_Adj
Arizona Cardinals	20.923077	28.769231	0.913995	1.193411
Atlanta Falcons	23.076923	26.384615	0.938825	1.077970
Baltimore Ravens	33.076923	18.153846	1.542580	0.835868
Buffalo Bills	21.076923	16.307692	0.904926	0.803488
Carolina Panthers	23.076923	27.692308	0.964388	1.192742
Chicago Bears	18.692308	17.846154	0.815416	0.817924
Cincinnati Bengals	15.230769	25.000000	0.713382	1.056457
Cleveland Browns	21.000000	22.384615	1.012960	0.991645
Dallas Cowboys	25.692308	20.538462	1.123913	1.002929
Denver Broncos	18.153846	20.076923	0.845988	0.902960
Detroit Lions	22.076923	25.769231	0.996711	1.175986
Green Bay Packers	23.769231	20.769231	1.037239	0.916959
Houston Texans	24.384615	23.769231	1.118323	1.019582
Indianapolis Colts	22.769231	22.692308	0.977311	1.010512
Jacksonville Jaguars	17.692308	25.923077	0.785023	1.131252
Kansas City Chiefs	28.538462	21.615385	1.333809	0.928131
Los Angeles Chargers	22.230769	19.307692	0.975750	0.914380
Los Angeles Rams	23.923077	20.153846	0.977316	0.791498
Miami Dolphins	17.000000	30.692308	0.823327	1.375989
Minnesota Vikings	26.076923	19.153846	1.093235	0.860586
New England Patriots	26.000000	12.923077	1.142878	0.548545
New Orleans Saints	26.461538	22.769231	1.123784	0.925692
New York Giants	19.000000	27.846154	0.870461	1.215548
New York Jets	17.384615	23.153846	0.738654	1.164162
Oakland Raiders	19.846154	28.153846	0.931637	1.239668
Philadelphia Eagles	22.846154	23.153846	1.049712	1.095948
Pittsburgh Steelers	19.923077	18.615385	0.886544	0.795200
San Francisco 49ers	30.538462	17.615385	1.277379	0.757684
Seattle Seahawks	26.230769	24.692308	1.188475	1.025568
Tampa Bay Buccaneers	29.076923	29.307692	1.226356	1.236086
Tennessee Titans	24.461538	19.615385	1.016233	0.893078
Washington Redskins	14.461538	23.846154	0.653472	1.102553

Predictive model for upcoming week NFL games (see full description of functions in Dimension 4)

In [19]:

```
averages['Offense'] =pd.to_numeric(averages['Offense'])
averages['Defense'] =pd.to_numeric(averages['Defense'])
averages['Offense_Adj'] =pd.to_numeric(averages['Offense_Adj'])
averages['Defense_Adj'] =pd.to_numeric(averages['Defense_Adj'])
```

In [20]:

```
def predict_home(x):
    home_off = averages.loc[x['Home'],'Offense']
    away_def = averages.loc[x['Away'],'Defense']
    avg = (home_off + away_def) / 2
    home_off_adj = averages.loc[x['Home'],'Offense_Adj']
    away_def_adj = averages.loc[x['Away'],'Defense_Adj']
    avg_adj = (home_off_adj + away_def_adj) / 2
    final_prediction = avg * avg_adj
    return final_prediction

def predict_away(x):
    away_off = averages.loc[x['Away'],'Offense']
    home_def = averages.loc[x['Home'],'Defense']
    avg = (away_off + home_def) / 2
    away_off_adj = averages.loc[x['Away'],'Offense_Adj']
    home_def_adj = averages.loc[x['Home'],'Defense_Adj']
    avg_adj = (away_off_adj + home_def_adj) / 2
    final_prediction = avg * avg_adj
    return final_prediction

def predict_total_points(x):
    return x['Predicted_Home'] + x['Predicted_Away']

def predict_difference(x):
    return x['Predicted_Home'] - x['Predicted_Away']

upcoming_week['Predicted_Home'] = upcoming_week.apply(predict_home, axis=1)
upcoming_week['Predicted_Away'] = upcoming_week.apply(predict_away, axis=1)
upcoming_week['Predicted_Total'] = upcoming_week.apply(predict_total_points, axi
s=1)
upcoming_week['Predicted_Difference'] = upcoming_week.apply(predict_difference,
axis=1)
upcoming_week
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:27: Set
tingWithCopyWarning:
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: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:28: Set
tingWithCopyWarning:
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: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:29: Set
tingWithCopyWarning:
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: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```

Out[20]:

	Home	Away	Predicted_Home	Predicted_Away	Predicted_Total	Predicted_Diffe
0	Baltimore Ravens	New York Jets	38.050541	13.989025	52.039566	24.0
0	New York Giants	Miami Dolphins	27.907813	22.858926	50.766739	5.0
1	Tennessee Titans	Houston Texans	24.547235	22.125413	46.672649	2.4
2	Cincinnati Bengals	New England Patriots	8.882024	28.041511	36.923535	-19.1
3	Carolina Panthers	Seattle Seahawks	23.764670	32.100631	55.865301	-8.3
4	Detroit Lions	Tampa Bay Buccaneers	28.682854	32.939798	61.622652	-4.2
5	Kansas City Chiefs	Denver Broncos	27.185342	17.638846	44.824188	9.5
6	Washington Redskins	Philadelphia Eagles	16.451275	25.123557	41.574833	-8.6
7	Green Bay Packers	Chicago Bears	19.300837	17.090546	36.391383	2.2
8	Oakland Raiders	Jacksonville Jaguars	23.604211	23.206072	46.810283	0.3
9	Arizona Cardinals	Cleveland Browns	20.632210	27.452347	48.084556	-6.8
10	Los Angeles Chargers	Minnesota Vikings	18.999010	22.778705	41.777715	-3.7
11	San Francisco 49ers	Atlanta Falcons	33.518419	17.258718	50.777137	16.2
12	Dallas Cowboys	Los Angeles Rams	21.953554	22.011188	43.964742	-0.0
13	Pittsburgh Steelers	Buffalo Bills	15.307791	16.870479	32.178270	-1.5
0	New Orleans Saints	Indianapolis Colts	26.227212	21.664953	47.892165	4.5

4. Compare Predictions with Betting Odds

Predictions are listed next to betting odds in order to make easy decisions on the upcoming week's spreads and O/Us

In [21]:

```
# join with upcoming_week table with bets table to compare data and make a choic
e
upcoming_week = upcoming_week.set_index('Home')
bets = bets.set_index('Team Favored to Win')
df_merge1 = pd.merge(upcoming_week, bets, how='inner', left_index=True, right_in
dex=True)
df_merge1 = df_merge1.reset_index()
df_merge1 = df_merge1.rename(columns={"index": "Home"})
df_merge1['Team Favored to Win'] = df_merge1['Home']
```

In [22]:

```
upcoming_week = upcoming_week.reset_index()
bets = bets.reset_index()
upcoming_week = upcoming_week.set_index('Away')
bets = bets.set_index('Team Favored to Win')
df_merge2 = pd.merge(upcoming_week, bets, how='inner', left_index=True, right_in
dex=True)
df_merge2 = df_merge2.reset_index()
df_merge2 = df_merge2.rename(columns={"index": "Away"})
df_merge2['Team Favored to Win'] = df_merge2['Away']
```

In [23]:

```
upcoming_week_predictions = df_merge1.append(df_merge2, sort=False)
upcoming_week_predictions = upcoming_week_predictions[['Home', 'Away', 'Predicted_Home', 'Predicted_Away', 'Predicted_Difference', 'Team Favored to Win', 'Spread', 'Spread Price', 'Predicted_Total','O/U', 'O/U Price']]
upcoming_week_predictions
```

Out[23]:

	Home	Away	Predicted_Home	Predicted_Away	Predicted_Difference	Team Favored to Win
0	Baltimore Ravens	New York Jets	38.050541	13.989025	24.061515	Baltimore Ravens
1	New York Giants	Miami Dolphins	27.907813	22.858926	5.048887	New York Giants
2	Tennessee Titans	Houston Texans	24.547235	22.125413	2.421822	Tennessee Titans
3	Kansas City Chiefs	Denver Broncos	27.185342	17.638846	9.546496	Kansas City Chiefs
4	Green Bay Packers	Chicago Bears	19.300837	17.090546	2.210290	Green Bay Packers
5	Oakland Raiders	Jacksonville Jaguars	23.604211	23.206072	0.398139	Oakland Raiders
6	San Francisco 49ers	Atlanta Falcons	33.518419	17.258718	16.259701	San Francisco 49ers
7	Pittsburgh Steelers	Buffalo Bills	15.307791	16.870479	-1.562688	Pittsburgh Steelers
8	New Orleans Saints	Indianapolis Colts	26.227212	21.664953	4.562259	New Orleans Saints
0	Cincinnati Bengals	New England Patriots	8.882024	28.041511	-19.159487	New England Patriots
1	Carolina Panthers	Seattle Seahawks	23.764670	32.100631	-8.335961	Seattle Seahawks
2	Detroit Lions	Tampa Bay Buccaneers	28.682854	32.939798	-4.256944	Tampa Bay Buccaneers
3	Washington Redskins	Philadelphia Eagles	16.451275	25.123557	-8.672282	Philadelphia Eagles
4	Arizona Cardinals	Cleveland Browns	20.632210	27.452347	-6.820137	Cleveland Browns
5	Los Angeles Chargers	Minnesota Vikings	18.999010	22.778705	-3.779694	Minnesota Vikings
6	Dallas Cowboys	Los Angeles Rams	21.953554	22.011188	-0.057634	Los Angeles Rams

5. Make Data Visualizations to Determine Bets

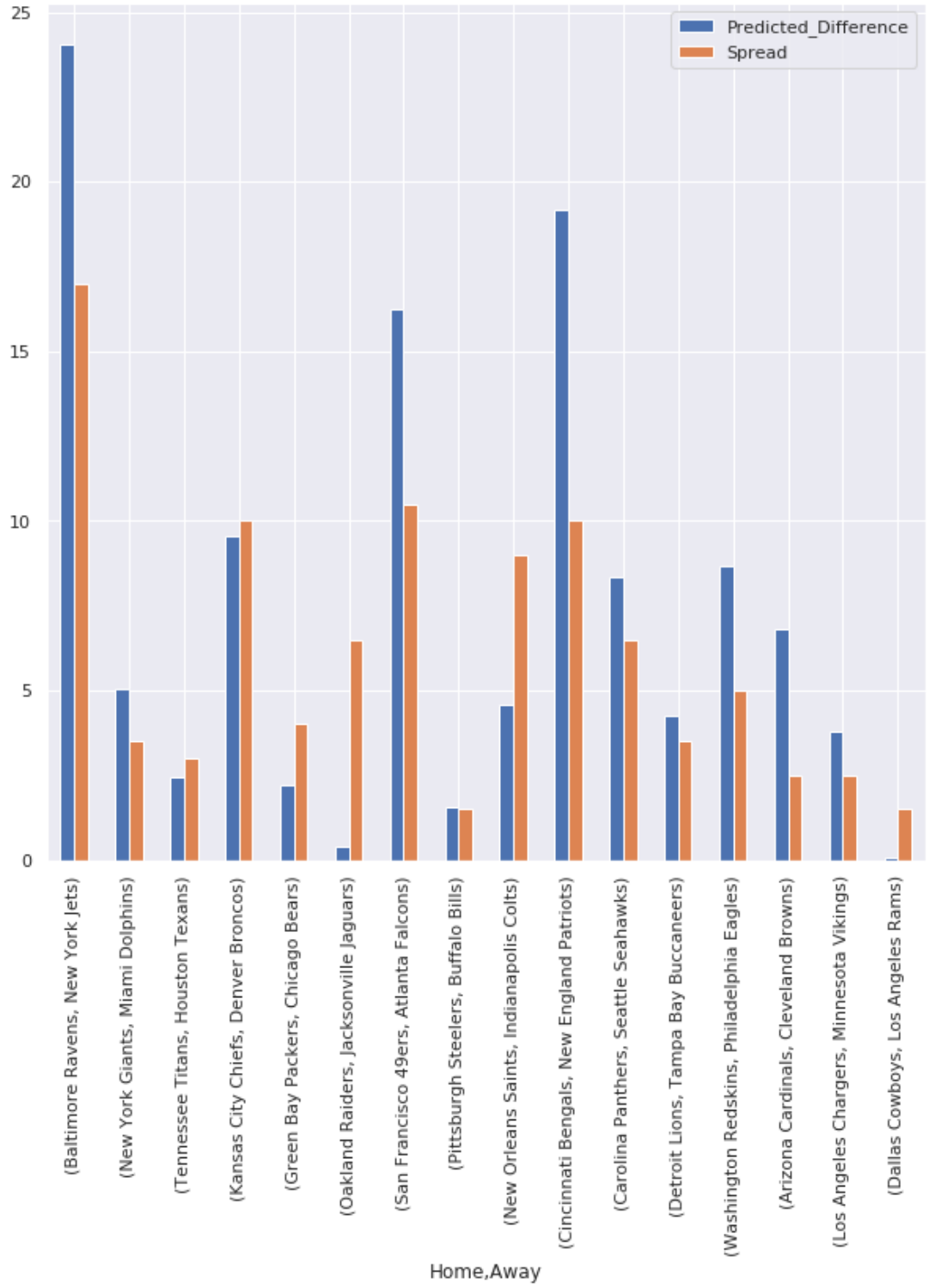
I compare predicted team differences to spreads in a bar chart - anywhere there is a large disparity between predicted difference and spread could be a good game to bet against the spread.

In [24]:

```
upcoming_week_predictions['Predicted_Difference'] = abs(upcoming_week_prediction
s['Predicted_Difference'])
upcoming_week_predictions['Spread'] = abs(upcoming_week_predictions['Spread'])
```

In [25]:

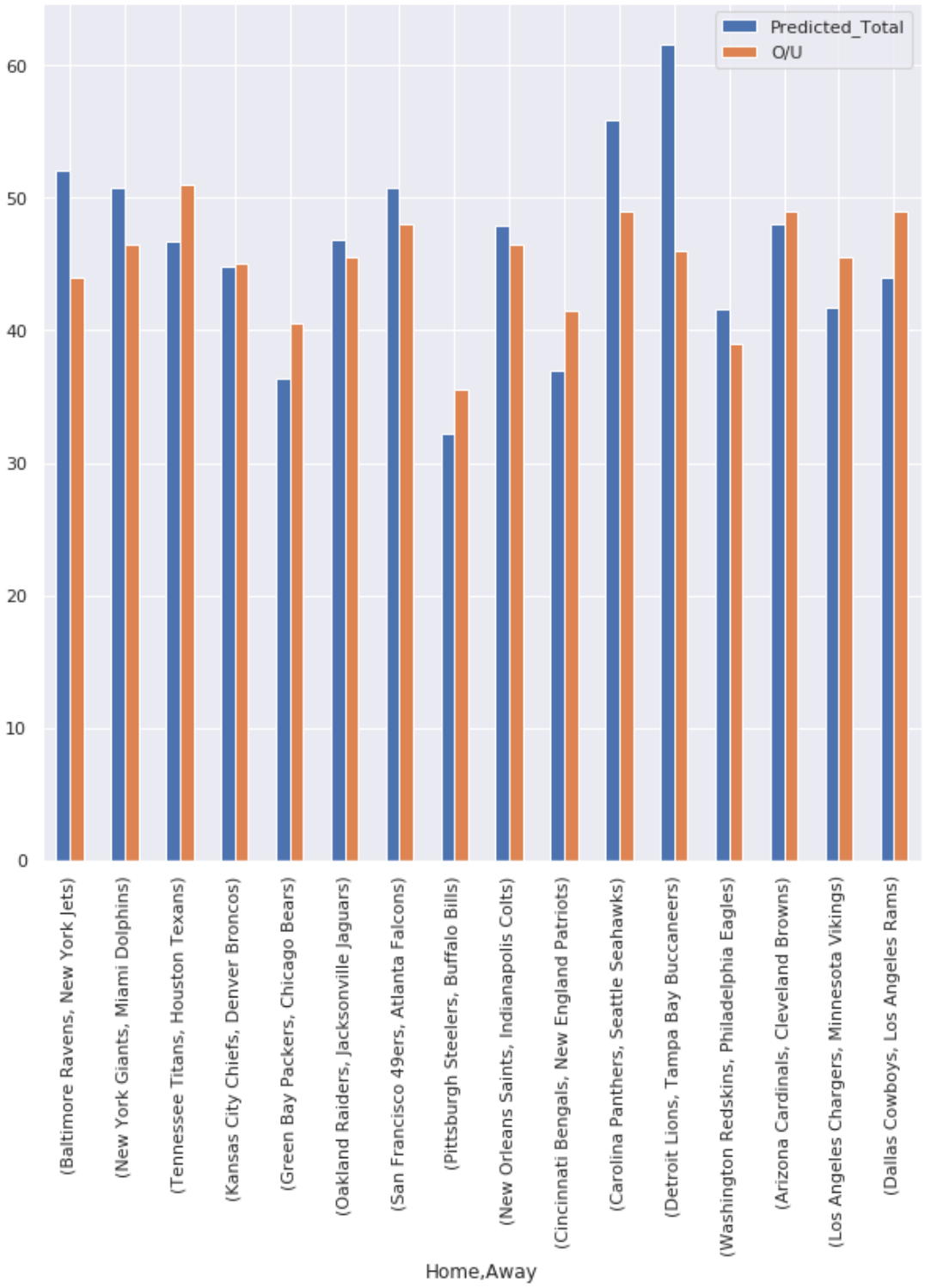
```
sns.set()  # use Seaborn styles
bar_chart = upcoming_week_predictions.set_index(['Home', 'Away'])[['Predicted_Di
fference', 'Spread']].plot.bar(figsize=(10,10))
```


Predicted_Difference
Spread
25
20
15
10
5
0
(Baltimore Ravens, New York Jets)
(New York Giants, Miami Dolphins)
(Tennessee Titans, Houston Texans)
(Kansas City Chiefs, Denver Broncos)
(Green Bay Packers, Chicago Bears)
(Oakland Raiders, Jacksonville Jaguars)
(San Francisco 49ers, Atlanta Falcons)
(Pittsburgh Steelers, Buffalo Bills)
(New Orleans Saints, Indianapolis Colts)
(Cincinnati Bengals, New England Patriots)
(Carolina Panthers, Seattle Seahawks)
(Detroit Lions, Tampa Bay Buccaneers)
(Washington Redskins, Philadelphia Eagles)
(Arizona Cardinals, Cleveland Browns)
(Los Angeles Chargers, Minnesota Vikings)
(Dallas Cowboys, Los Angeles Rams)
Home,Away

I compare predicted team totals to O/U in a bar chart - anywhere there is a large disparity between predicted total and O/U could be a good game to bet either the Over (if the predicted total blue line is largely above the O/U orange line) or Under (if the predicted total blue line is clearly below the O/U orange line).

In [26]:

```
upcoming_week_predictions['O/U'] = pd.to_numeric(upcoming_week_predictions['O/U'
])
bar_chart = upcoming_week_predictions.set_index(['Home', 'Away'])[['Predicted_To
tal', 'O/U']].plot.bar(figsize=(10,10))
```


Predicted_Total
O/U
60
50
40
30
20
10
0
(Baltimore Ravens, New York Jets)
(New York Giants, Miami Dolphins)
(Tennessee Titans, Houston Texans)
(Kansas City Chiefs, Denver Broncos)
(Green Bay Packers, Chicago Bears)
(Oakland Raiders, Jacksonville Jaguars)
(San Francisco 49ers, Atlanta Falcons)
(Pittsburgh Steelers, Buffalo Bills)
(New Orleans Saints, Indianapolis Colts)
(Cincinnati Bengals, New England Patriots)
(Carolina Panthers, Seattle Seahawks)
(Detroit Lions, Tampa Bay Buccaneers)
(Washington Redskins, Philadelphia Eagles)
(Arizona Cardinals, Cleveland Browns)
(Los Angeles Chargers, Minnesota Vikings)
(Dallas Cowboys, Los Angeles Rams)
Home,Away

FOR WEEK 15 PREDICTIVE BETTING DATA:

MIGHT BE DIFFERENT INFORMATION IF YOU ARE RUNNING THE NOTEBOOK TO PREDICT GAMES AFTER WEEK 15

Potential game to bet against the spread: BAL -15

Potential games to bet the O/U: DET O47

6. Analyze Team Data to Verify Bets

To verify certain bets, it is helpful to look at average team data to confirm the predictive model.

BELOW I WILL ANALYZE BETTING BAL -15: THIS ANALYSIS WILL NOT APPLY TO WEEKS AFTER 15 (THE CURRENT WEEK I AM ANALYZING)

To verify BAL -15 where the Baltimore Ravens are playing at home against the New York Jets, I look at how many points BAL scores at home vs away on average, their overall record compared to the NYJ, and how many points they typically score under the kind of stadium they are playing in (in this scenario, at home in a fixed roof).

In [27]:

```
result_pivot = pd.pivot_table(result, values=['Offense', 'Defense'], index=['tea
m', 'Roof Type'], aggfunc=np.nanmean)
result_pivot.head(60)
```

Out[27]:

team	Roof Type	Defense	Offense
Arizona Cardinals	Fixed	31.000000	9.000000
	Open	26.600000	24.600000
	Retractable	30.000000	20.000000
Atlanta Falcons	Fixed	18.500000	19.000000
	Open	3.000000	29.000000
	Retractable	30.300000	23.300000
Baltimore Ravens	Open	18.153846	33.076923
Buffalo Bills	Open	16.416667	20.666667
	Retractable	15.000000	26.000000
Carolina Panthers	Fixed	34.000000	31.000000
	Open	28.444444	21.666667
	Retractable	23.333333	24.666667
Chicago Bears	Fixed	20.000000	24.000000
	Open	17.666667	18.250000
Cincinnati Bengals	Open	25.000000	15.230769
Cleveland Browns	Open	22.384615	21.000000
Dallas Cowboys	Fixed	19.500000	22.500000
	Open	21.400000	24.600000
	Retractable	20.166667	27.666667
Denver Broncos	Fixed	27.000000	23.000000
	Open	19.500000	16.200000
	Retractable	19.500000	25.500000
Detroit Lions	Fixed	27.285714	22.571429
	Open	23.400000	20.400000
	Retractable	27.000000	27.000000
Green Bay Packers	Open	20.500000	22.916667
	Retractable	24.000000	34.000000
Houston Texans	Fixed	30.000000	28.000000
	Open	22.000000	22.750000
	Retractable	23.875000	24.750000
Indianapolis Colts	Open	24.800000	24.200000
	Retractable	21.375000	21.875000
Jacksonville Jaguars	Open	26.454545	18.636364
	Retractable	23.000000	12.500000
Kansas City Chiefs	Fixed	30.000000	34.000000
	Open	20.916667	28.083333

team	Roof Type	Defense	Offense
Los Angeles Chargers	Fixed	13.000000	10.000000
	Open	19.833333	23.250000
Los Angeles Rams	Open	22.272727	21.818182
	Retractable	8.500000	35.500000
Miami Dolphins	Open	32.363636	18.090909
	Retractable	21.500000	11.000000
Minnesota Vikings	Fixed	16.428571	29.714286
	Open	22.000000	20.600000
	Retractable	24.000000	28.000000
New England Patriots	Open	11.666667	26.333333
	Retractable	28.000000	22.000000
New Orleans Saints	Fixed	25.142857	27.571429
	Open	20.400000	25.000000
	Retractable	18.000000	26.000000
New York Giants	Fixed	31.000000	26.000000
	Open	26.909091	18.545455
	Retractable	35.000000	17.000000
New York Jets	Open	23.153846	17.384615
Oakland Raiders	Fixed	34.000000	14.000000
	Open	28.100000	18.900000
	Retractable	25.500000	27.500000
Philadelphia Eagles	Fixed	38.000000	20.000000
	Open	20.200000	24.700000
	Retractable	30.500000	15.000000

Baltimore scores on average 33 points in an open roof, whereas the Jets score 17. — BAL could win by 16 points so BAL -15 still looks like a good bet.

In [28]:

```
sns.set()  # use Seaborn styles
records = pd.pivot_table(result, values = 'OPP', index='team', columns='Outcome'
, aggfunc='count')
records.plot.bar(figsize=(15,15))
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f85f4292be0>
```

BAL has won 11 games and the NYJ have won only 5.

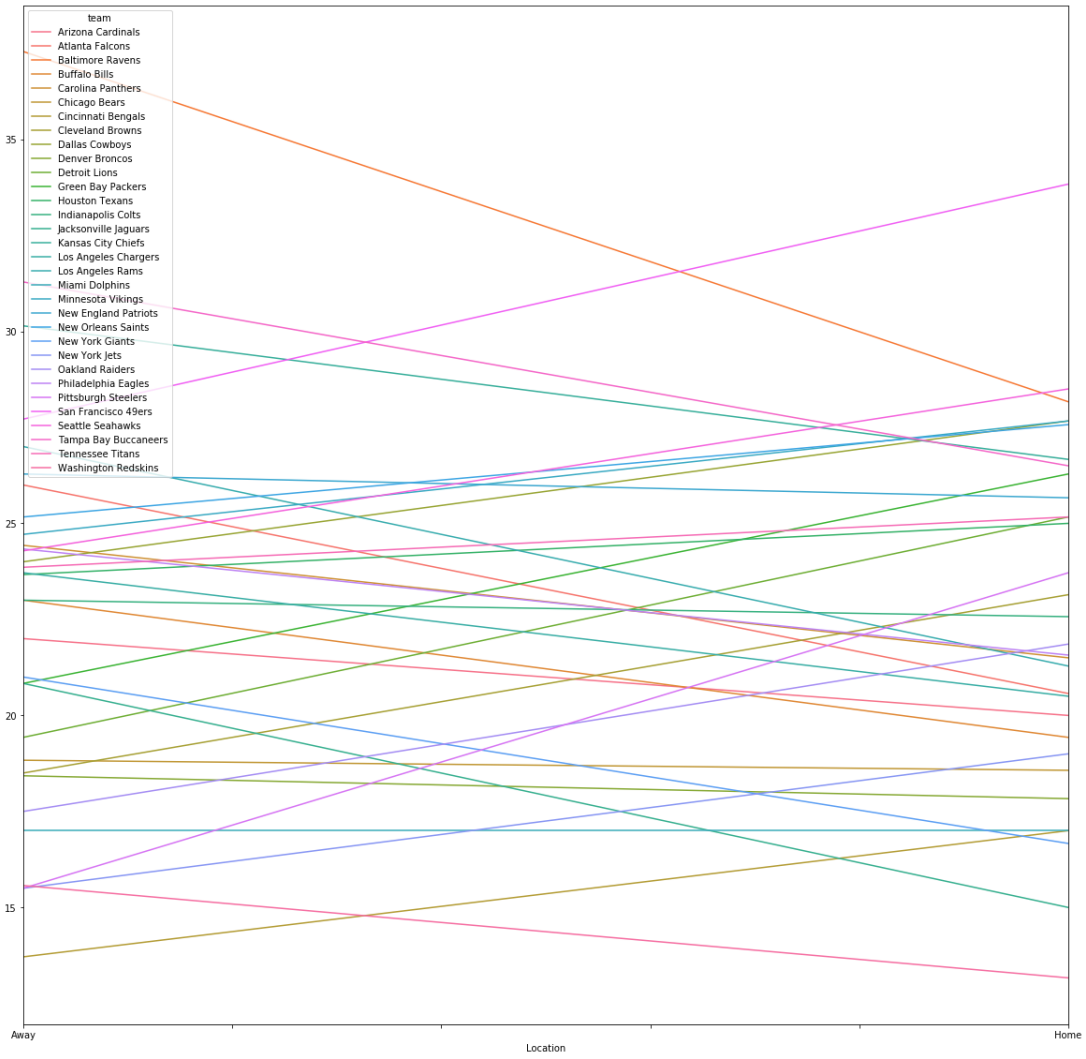
In [29]:

```
NUM_COLORS = 32

sns.reset_orig()  # get default matplotlib styles back
clrs = sns.color_palette('husl', n_colors=NUM_COLORS)  # a list of RGB tuples
scores_by_location = result.pivot_table('Offense', index='Location', columns='te
am', aggfunc='mean').plot(figsize=(20, 20), color=clrs)

plt.show()
plt. savefig('avgScoreByLocation.png')
```

```
/opt/conda/lib/python3.7/_collections_abc.py:841: MatplotlibDeprecat
ionWarning:
The examples.directory rcparam was deprecated in Matplotlib 3.0 and
will be removed in 3.2. In the future, examples will be found relati
ve to the 'datapath' directory.
  self[key] = other[key]
/opt/conda/lib/python3.7/_collections_abc.py:841: MatplotlibDeprecat
ionWarning:
The savefig.frameon rcparam was deprecated in Matplotlib 3.1 and wil
l be removed in 3.3.
  self[key] = other[key]
/opt/conda/lib/python3.7/_collections_abc.py:841: MatplotlibDeprecat
ionWarning:
The text.latex.unicode rcparam was deprecated in Matplotlib 3.0 and
will be removed in 3.2.
  self[key] = other[key]
/opt/conda/lib/python3.7/_collections_abc.py:841: MatplotlibDeprecat
ionWarning:
The verbose.fileo rcparam was deprecated in Matplotlib 3.1 and will
be removed in 3.3.
  self[key] = other[key]
/opt/conda/lib/python3.7/_collections_abc.py:841: MatplotlibDeprecat
ionWarning:
The verbose.level rcparam was deprecated in Matplotlib 3.1 and will
be removed in 3.3.
  self[key] = other[key]
```


team
Arizona Cardinals
Atlanta Falcons
Baltimore Ravens
Buffalo Bills
Carolina Panthers
Chicago Bears
Cincinnati Bengals
Cleveland Browns
Dallas Cowboys
Denver Broncos
Detroit Lions
Green Bay Packers
Houston Texans
Indianapolis Colts
Jacksonville Jaguars
Kansas City Chiefs
Los Angeles Chargers
Los Angeles Rams
Miami Dolphins
Minnesota Vikings
New England Patriots
New Orleans Saints
New York Giants
New York Jets
Oakland Raiders
Philadelphia Eagles
Pittsburgh Steelers
San Francisco 49ers
Seattle Seahawks
Tampa Bay Buccaneers
Tennessee Titans
Washington Redskins
35
30
25
20
15
Away
Home
Location

```
<Figure size 432x288 with 0 Axes>
```

BAL scores more point away than home but still scores around 29 points at home.

BAL -15 seems like a good bet based on the predictive model, and sports gambling is close to being legalized in Michigan: https://www.actionnetwork.com/news/michigan-legalizes-sports-betting-online?utm_source=Iterable&utm_medium=email&utm_campaign=campaign_925793 (https://www.actionnetwork.com/news/michigan-legalizes-sports-betting-online?utm_source=Iterable&utm_medium=email&utm_campaign=campaign_925793)

Author: Caroline Gilhool, University of Michigan, December 12, 2019

In [ ]: