Getting Under and Over Vegas NFL Lines

In [632]:

```
from IPython.display import Image
Image("/Users/matthewnykaza/Documents/Flatiron/Phase_3_Project/NFL_Betting_Data/images/vegas_image.jpeg")
```

Out[632]:

Overview

For this project I sourced NFL betting, stadiums and teams data from the data website Kaggle. This data includes data going back to the 1978 NFL seasons
creating a classification model that will assist sports bettors with selecting the over or under in an NFL game. The Over/Under line is the predicted combined
the two teams must combine for greater than that total, and to go Under they must combine for less than that total. I began this project by preprocessing the
variables, engineering relevant ones from the given data, and performing intelligent decisions as what to do with missing/incomplete data. Once the processi
using that data, and Sklearn's pipeline function. After some early attempts with base a Logistic Regression model, it was determined that using a tree-based
scores. The final model used was a Random Forest model, with some hyperparameter tuning, that allowed me to achieve roughly 53% accuracy on my test
information, and continue to tune the model to achieve optimal results.

In [603]:

```
import numpy as np
import pandas as pd
import requests
from geopy.extra.rate_limiter import RateLimiter
from geopy.geocoders import Nominatim
import geocoder
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score, RandomizedSearchCV
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, plot_confusion_matrix, roc_curve, auc
from sklearn.neighbors import KNeighborsClassifier
import category_encoders as ce
import seaborn as sns
from statsmodels.stats.outliers_influence import variance_inflation_factor
import xgboost as xgb
```

The Datasets

For this project I had three datasets

- NFL betting information which included dates, week of schedule, teams, scores, betting lines, weather, stadium names, and playoff information
- NFL stadium data which had more detailed information about the various stadiums that NFL teams have played at since 1978
- NFL team data which included information about individual teams such as nicknames, conference and division information The main data that I used wa information to dig into greater detail about individual stadiums, and I used the NFL teams data to compare conference/divisional matchups, as well as he team's last 5 games.

The main data that I used was te NFL betting, but I used the stadium information to dig into greater detail about individual stadiums, and I used the NFL team matchups, as well as help setup the average scores for each individual team's last 5 games.

Load in Datasets

In [ ]:

```

```

In [2]:

```
nfl_teams = pd.read_csv("//Users/matthewnykaza/Documents/Flatiron/Phase_3_Project/NFL_Betting_Data/Files/nfl_teams.cs
```

In [171]:

```
nfl_teams2 = pd.read_csv("//Users/matthewnykaza/Documents/Flatiron/Phase_3_Project/NFL_Betting_Data/Files/nfl_teams.c
```

In [3]:

```
nfl_teams.loc[nfl_teams['team_name']== 'Las Vegas Raiders']
```

Out[3]:

	team_name	team_name_short	team_id	team_id_pfr	team_conference	team_division	team_conference_pre2002	team_division_pre2002
31	Las Vegas Raiders	Raiders	LVR	RAI	AFC	NaN	AFC	AFC West

```
In [4]: nfl_teams.loc[nfl_teams['team_name'] == 'Las Vegas Raiders', 'team_division'] = 'AFC West'
```

```
In [5]: nfl_stadiums = pd.read_csv("//Users/matthewnykaza/Documents/Flatiron/Phase_3_Project/NFL_Betting_Data/Files/nfl_stadi
        nfl_stadiums.head()
```

Out[5]:

	stadium_name	stadium_location	stadium_open	stadium_close	stadium_type	stadium_address	stadium_weather_station_code	stadium_weather_type	stadi
0	Alamo Dome	San Antonio, TX	NaN	NaN	indoor	100 Montana St, San Antonio, TX 78203	78203	dome	
1	Allegiant Stadium	Paradise, NV	2020.0	NaN	indoor	NaN	NaN	dome	
2	Alltel Stadium	Jacksonville, FL	NaN	NaN	NaN	NaN	NaN	NaN	
3	Alumni Stadium	Chestnut Hill, MA	NaN	NaN	outdoor	Perimeter Rd, Chestnut Hill, MA 02467	2467	cold	
4	Anaheim Stadium	Anaheim, CA	1980.0	1994.0	outdoor	2000 E Gene Autry Way, Anaheim, CA 92806	92806	warm	

```
In [6]: nfl_scores = pd.read_csv("//Users/matthewnykaza/Documents/Flatiron/Phase_3_Project/NFL_Betting_Data/Files/spreadspoke
        nfl_scores.head()
```

Out[6]:

ason	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite	over_under_line	stadium	stadiun
1966	1	False	Miami Dolphins	14.0	23.0	Oakland Raiders	NaN	NaN	NaN	Orange Bowl	
1966	1	False	Houston Oilers	45.0	7.0	Denver Broncos	NaN	NaN	NaN	Rice Stadium	
1966	1	False	San Diego Chargers	27.0	7.0	Buffalo Bills	NaN	NaN	NaN	Balboa Stadium	
1966	2	False	Miami Dolphins	14.0	19.0	New York Jets	NaN	NaN	NaN	Orange Bowl	
1966	1	False	Green Bay Packers	24.0	3.0	Baltimore Colts	NaN	NaN	NaN	Lambeau Field	

Focus on NFL Stadiums Null Values

In [7]:

```
nfl_stadiums.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 106 entries, 0 to 105
Data columns (total 15 columns):
stadium_name                    106 non-null object
stadium_location                106 non-null object
stadium_open                    82 non-null float64
stadium_close                   41 non-null float64
stadium_type                    99 non-null object
stadium_address                 94 non-null object
stadium_weather_station_code    93 non-null object
stadium_weather_type            99 non-null object
stadium_capacity                45 non-null object
stadium_surface                 59 non-null object
STATION                         55 non-null object
NAME                            55 non-null object
LATITUDE                        55 non-null float64
LONGITUDE                       55 non-null float64
ELEVATION                       55 non-null float64
dtypes: float64(5), object(10)
memory usage: 12.5+ KB
```

A few quick thoughts

- We want this to help make sure we have all the data possible for a merge on nfl_scores
- Knowing indoor and outdoor stadiums will be important later on, and we can find that info on the web and do a .loc to update the info
- Once we have the address and stadium type's updated, we can import all latitudes, longitudes and elevation information
- We should also be able to get weather on a given day with this info aswell

In [8]:

```
nfl_stadiums[nfl_stadiums['stadium_address'].isnull()]
```

Out[8]:

	stadium_name	stadium_location	stadium_open	stadium_close	stadium_type	stadium_address	stadium_weather_station_code	stadium_weather_type	sta
1	Allegiant Stadium	Paradise, NV	2020.0	NaN	indoor	NaN	NaN	dome	
2	Alltel Stadium	Jacksonville, FL	NaN	NaN	NaN	NaN	NaN	NaN	
18	Dolphin Stadium	Miami, FL	NaN	NaN	NaN	NaN	NaN	NaN	
38	Jack Murphy Stadium	San Diego, CA	NaN	NaN	NaN	NaN	NaN	NaN	
39	Joe Robbie Stadium	Miami, FL	NaN	NaN	NaN	NaN	NaN	NaN	
55	Mercedes-Benz Stadium	Atlanta, GA	2017.0	NaN	indoor	NaN	NaN	dome	
69	Pro Player Stadium	Miami, FL	NaN	NaN	NaN	NaN	NaN	NaN	

In [9]:
```
nfl_stadiums[nfl_stadiums['stadium_type'].isnull()]
```

Out[9]:

	stadium_name	stadium_location	stadium_open	stadium_close	stadium_type	stadium_address	stadium_weather_station_code	stadium_weather_type	stac
2	Alltel Stadium	Jacksonville, FL	NaN	NaN	NaN	NaN	NaN	NaN	
18	Dolphin Stadium	Miami, FL	NaN	NaN	NaN	NaN	NaN	NaN	
38	Jack Murphy Stadium	San Diego, CA	NaN	NaN	NaN	NaN	NaN	NaN	
39	Joe Robbie Stadium	Miami, FL	NaN	NaN	NaN	NaN	NaN	NaN	
69	Pro Player Stadium	Miami, FL	NaN	NaN	NaN	NaN	NaN	NaN	
84	Stanford Stadium	Palo Alto, CA	NaN	NaN	NaN	NaN	NaN	NaN	
89	Tampa Stadium	Tampa, FL	NaN	NaN	NaN	NaN	NaN	NaN	

In [10]:
```
addresses = ['3333 Al Davis Way, Las Vegas, NV 89118', '1 TIAA Bank Field Dr, Jacksonville, FL 32202', '347 Don Shula
types = ['outdoor', 'outdoor', 'outdoor', 'outdoor', 'outdoor', 'outdoor', 'outdoor']
```

In [11]:
```
nfl_stadiums.loc[nfl_stadiums['stadium_address'].isnull(), 'stadium_address'] = addresses
nfl_stadiums.loc[nfl_stadiums['stadium_type'].isnull(), 'stadium_type'] = types
nfl_stadiums.head()
```

Out[11]:

	stadium_name	stadium_location	stadium_open	stadium_close	stadium_type	stadium_address	stadium_weather_station_code	stadium_weather_type	stadi
0	Alamo Dome	San Antonio, TX	NaN	NaN	indoor	100 Montana St, San Antonio, TX 78203	78203	dome	
1	Allegiant Stadium	Paradise, NV	2020.0	NaN	indoor	3333 Al Davis Way, Las Vegas, NV 89118	NaN	dome	
2	Alltel Stadium	Jacksonville, FL	NaN	NaN	outdoor	1 TIAA Bank Field Dr, Jacksonville, FL 32202	NaN	NaN	
3	Alumni Stadium	Chestnut Hill, MA	NaN	NaN	outdoor	Perimeter Rd, Chestnut Hill, MA 02467	2467	cold	
4	Anaheim Stadium	Anaheim, CA	1980.0	1994.0	outdoor	2000 E Gene Autry Way, Anaheim, CA 92806	92806	warm	

In [12]:
```
nfl_stadiums['stadium_address'].isnull().sum()
```

Out[12]:
```
0
```

In [13]:
```
nfl_stadiums['stadium_type'].isnull().sum()
```

Out[13]:
```
0
```

In [14]:
```
nfl_stadiums.head(50)
```

Out[14]:

	stadium_name	stadium_location	stadium_open	stadium_close	stadium_type	stadium_address	stadium_weather_station_code	stadium_weather_type	stac
0	Alamo Dome	San Antonio, TX	NaN	NaN	indoor	100 Montana St, San Antonio, TX 78203	78203	dome	
1	Allegiant Stadium	Paradise, NV	2020.0	NaN	indoor	3333 Al Davis Way, Las Vegas, NV 89118	NaN	dome	
2	Alltel Stadium	Jacksonville, FL	NaN	NaN	outdoor	1 TIAA Bank Field Dr, Jacksonville, FL 32202	NaN	NaN	
3	Alumni Stadium	Chestnut Hill, MA	NaN	NaN	outdoor	Perimeter Rd, Chestnut Hill, MA 02467	2467	cold	
4	Anaheim Stadium	Anaheim, CA	1980.0	1994.0	outdoor	2000 E Gene Autry Way, Anaheim, CA 92806	92806	warm	

In [15]:
```
nfl_stadiums.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 106 entries, 0 to 105
Data columns (total 15 columns):
stadium_name                    106 non-null object
stadium_location                106 non-null object
stadium_open                    82 non-null float64
stadium_close                   41 non-null float64
stadium_type                    106 non-null object
stadium_address                 106 non-null object
stadium_weather_station_code    93 non-null object
stadium_weather_type            99 non-null object
stadium_capacity                45 non-null object
stadium_surface                 59 non-null object
STATION                         55 non-null object
NAME                            55 non-null object
LATITUDE                        55 non-null float64
LONGITUDE                       55 non-null float64
ELEVATION                       55 non-null float64
dtypes: float64(5), object(10)
memory usage: 12.5+ KB
```

Now let's update all latitudes, longitudes and elevation

In [16]:
```
locator = Nominatim(user_agent='myGeocoder')
```

In [17]:

```
geocode = RateLimiter(locator.geocode, min_delay_seconds=1)
nfl_stadiums['location'] = nfl_stadiums['stadium_address'].apply(geocode)
nfl_stadiums['point'] = nfl_stadiums['location'].apply(lambda loc: tuple(loc.point) if loc else None)
nfl_stadiums[['latitude', 'longitude', 'elevation']] = pd.DataFrame(nfl_stadiums['point'].tolist(), index=nfl_stadiun
```

In [18]:

```
nfl_stadiums.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 106 entries, 0 to 105
Data columns (total 20 columns):
stadium_name                    106 non-null object
stadium_location                106 non-null object
stadium_open                    82 non-null float64
stadium_close                   41 non-null float64
stadium_type                    106 non-null object
stadium_address                 106 non-null object
stadium_weather_station_code    93 non-null object
stadium_weather_type            99 non-null object
stadium_capacity                45 non-null object
stadium_surface                 59 non-null object
STATION                         55 non-null object
NAME                            55 non-null object
LATITUDE                        55 non-null float64
LONGITUDE                       55 non-null float64
ELEVATION                       55 non-null float64
location                        82 non-null object
point                           82 non-null object
latitude                        82 non-null float64
longitude                       82 non-null float64
elevation                       82 non-null float64
dtypes: float64(8), object(12)
memory usage: 16.7+ KB
```

In [19]:

```
nfl_stadiums.head(50)
```

44	Liberty Bowl Memorial Stadium	Memphis, TN	1997.0	1997.0	outdoor	335 S Hollywood St, Memphis, TN 38104	38104	moderate
45	Lincoln Financial Field	Philadelphia, PA	2003.0	NaN	outdoor	1020 Pattison Ave, Philadelphia, PA 19148	19148	cold
46	Los Angeles Memorial Coliseum	Los Angeles, CA	1946.0	NaN	outdoor	3911 S Figueroa St, Los Angeles, CA 90037	90037	warm

- So we didn't get all of the lats and longs we wanted, but we certainly have more than before, so we'll keep them in for now and see what happens after t
- But for the lats and longs we did get, they look to be accurate to the existing data

- Elevations all 0, so that is dissapointing, but we will move on

Remove unwanted columns

```
In [20]: nfl_stadiums = nfl_stadiums.drop(columns=['stadium_location', 'stadium_open', 'stadium_close', 'stadium_weather_type'
         nfl_stadiums.head()
```

Out[20]:

	stadium_name	stadium_type	stadium_address	stadium_weather_station_code	STATION	NAME	ELEVATION	latitude	longitu
0	Alamo Dome	indoor	100 Montana St, San Antonio, TX 78203	78203	NaN	NaN	NaN	29.416892	-98.4788
1	Allegiant Stadium	indoor	3333 Al Davis Way, Las Vegas, NV 89118	NaN	NaN	NaN	NaN	NaN	Na
2	Alltel Stadium	outdoor	1 TIAA Bank Field Dr, Jacksonville, FL 32202	NaN	NaN	NaN	NaN	NaN	Na
3	Alumni Stadium	outdoor	Perimeter Rd, Chestnut Hill, MA 02467	2467	NaN	NaN	NaN	NaN	Na
4	Anaheim Stadium	outdoor	2000 E Gene Autry Way, Anaheim, CA 92806	92806	NaN	NaN	NaN	33.799711	-117.8893

```
In [ ]:
```

Reasoning for drops

- Location is worse than address
- open and close really don't matter at all for this
- Weather type is very arbitrary
- capacity doesn't really matter for what we are trying to do
- Surface has too many nulls, and there isn't enough time to look up each stadium
- LATITUDE and LONGITUDE were redunant with the ones we created
- elevation was useless, all 0

```
In [21]: nfl_stadiums.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 106 entries, 0 to 105
Data columns (total 9 columns):
stadium_name                    106 non-null object
stadium_type                    106 non-null object
stadium_address                 106 non-null object
stadium_weather_station_code    93 non-null object
STATION                         55 non-null object
NAME                            55 non-null object
ELEVATION                       55 non-null float64
latitude                        82 non-null float64
longitude                       82 non-null float64
dtypes: float64(3), object(6)
memory usage: 7.6+ KB
```

Move on to nfl_scores data

```
In [22]: nfl_scores.info()
```

```
Data columns (total 17 columns):
schedule_date           12947 non-null object
schedule_season         12947 non-null int64
schedule_week           12947 non-null object
schedule_playoff        12947 non-null bool
team_home               12947 non-null object
score_home              12946 non-null float64
score_away              12946 non-null float64
team_away               12947 non-null object
team_favorite_id        10468 non-null object
spread_favorite         10468 non-null float64
over_under_line         10458 non-null object
stadium                 12947 non-null object
stadium_neutral         12947 non-null bool
weather_temperature     12008 non-null float64
weather_wind_mph        12008 non-null float64
weather_humidity        8388 non-null object
weather_detail          2711 non-null object
dtypes: bool(2), float64(5), int64(1), object(9)
memory usage: 1.5+ MB
```

```
In [23]: #Make schedule a datetime
         nfl_scores['schedule_date'] = pd.to_datetime(nfl_scores['schedule_date'])
```

- Considering this whole project is about trying to get a classifier that can pick overs/unders we need all that data we can, any without that data will be rer

```
In [24]: nfl_scores = nfl_scores[nfl_scores['spread_favorite'].notna()]
         nfl_scores = nfl_scores[nfl_scores['over_under_line'].notna()]
```

```
In [25]: nfl_scores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10458 entries, 350 to 12946
Data columns (total 17 columns):
schedule_date           10458 non-null datetime64[ns]
schedule_season         10458 non-null int64
schedule_week           10458 non-null object
schedule_playoff        10458 non-null bool
team_home               10458 non-null object
score_home              10457 non-null float64
score_away              10457 non-null float64
team_away               10458 non-null object
team_favorite_id        10458 non-null object
spread_favorite         10458 non-null float64
over_under_line         10458 non-null object
stadium                 10458 non-null object
stadium_neutral         10458 non-null bool
weather_temperature     9749 non-null float64
weather_wind_mph        9749 non-null float64
weather_humidity        6289 non-null object
weather_detail          2553 non-null object
dtypes: bool(2), datetime64[ns](1), float64(5), int64(1), object(8)
memory usage: 1.3+ MB
```

In [26]:

```
nfl_scores.head(100)
```

Out[26]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite
350	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.5
538	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.0
727	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.0
916	1971-01-17	1970	Superbowl	True	Baltimore Colts	16.0	13.0	Dallas Cowboys	IND	-2.5
1105	1972-01-16	1971	Superbowl	True	Dallas Cowboys	24.0	3.0	Miami Dolphins	DAL	-6.0
...	...	...	...	...	...	...	...	...	...	...
2584	1979-10-08	1979	6	False	Oakland Raiders	13.0	3.0	Miami Dolphins	MIA	-1.0
2585	1979-10-14	1979	7	False	Baltimore Colts	16.0	28.0	Houston Oilers	TEN	-6.0
2586	1979-10-14	1979	7	False	Chicago Bears	7.0	27.0	New England Patriots	NE	-4.0
2587	1979-10-14	1979	7	False	Cincinnati Bengals	34.0	10.0	Pittsburgh Steelers	PIT	-10.0
2588	1979-10-14	1979	7	False	Cleveland Browns	9.0	13.0	Washington Redskins	CLE	-4.0

100 rows × 17 columns

- Only the superbowls before 1979 have the betting data, so we will remove those

In [27]:

```
nfl_scores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10458 entries, 350 to 12946
Data columns (total 17 columns):
schedule_date          10458 non-null datetime64[ns]
schedule_season        10458 non-null int64
schedule_week          10458 non-null object
schedule_playoff       10458 non-null bool
team_home              10458 non-null object
score_home             10457 non-null float64
score_away             10457 non-null float64
team_away              10458 non-null object
team_favorite_id       10458 non-null object
spread_favorite        10458 non-null float64
over_under_line        10458 non-null object
stadium                10458 non-null object
stadium_neutral        10458 non-null bool
weather_temperature    9749 non-null float64
weather_wind_mph       9749 non-null float64
weather_humidity       6289 non-null object
```

In [28]:

```
nfl_scores[nfl_scores['score_away'].isnull()]
```

Out[28]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorit
12946	2021-02-07	2020	Superbowl	True	Tampa Bay Buccaneers	NaN	NaN	Kansas City Chiefs	KC	-3.

- The only data without a score_home and score_away is the last superbowl, since this game happened this will be easy enough to input

In [29]:

```
nfl_scores.loc[nfl_scores['score_home'].isnull(), 'score_home'] = 31.0
nfl_scores.loc[nfl_scores['score_away'].isnull(), 'score_away'] = 9.0
```

In [30]:

```
nfl_scores[nfl_scores['score_away'].isnull()]
```

Out[30]:

schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite	ov

In [31]:
```
nfl_scores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10458 entries, 350 to 12946
Data columns (total 17 columns):
schedule_date          10458 non-null datetime64[ns]
schedule_season        10458 non-null int64
schedule_week          10458 non-null object
schedule_playoff       10458 non-null bool
team_home              10458 non-null object
score_home             10458 non-null float64
score_away             10458 non-null float64
team_away              10458 non-null object
team_favorite_id       10458 non-null object
spread_favorite        10458 non-null float64
over_under_line        10458 non-null object
stadium                10458 non-null object
stadium_neutral        10458 non-null bool
weather_temperature    9749 non-null float64
weather_wind_mph       9749 non-null float64
weather_humidity       6289 non-null object
weather_detail         2553 non-null object
dtypes: bool(2), datetime64[ns](1), float64(5), int64(1), object(8)
memory usage: 1.3+ MB
```

- Weather detail is mostly unfilled, and mostly arbitrary, we're just gonna drop that
- I may have a different idea for adding it back in

In [32]:
```
nfl_scores = nfl_scores.drop(columns='weather_detail')
```

In [33]:
```
nfl_scores.info()
```

```
Int64Index: 10458 entries, 350 to 12946
Data columns (total 16 columns):
schedule_date          10458 non-null datetime64[ns]
schedule_season        10458 non-null int64
schedule_week          10458 non-null object
schedule_playoff       10458 non-null bool
team_home              10458 non-null object
score_home             10458 non-null float64
score_away             10458 non-null float64
team_away              10458 non-null object
team_favorite_id       10458 non-null object
spread_favorite        10458 non-null float64
over_under_line        10458 non-null object
stadium                10458 non-null object
stadium_neutral        10458 non-null bool
weather_temperature    9749 non-null float64
weather_wind_mph       9749 non-null float64
weather_humidity       6289 non-null object
dtypes: bool(2), datetime64[ns](1), float64(5), int64(1), object(7)
memory usage: 1.2+ MB
```

- As far as the rest of the nulls, I want to wait until we have merged with the stadium data, I have an idea that a lot of that has to due with being in a indoor
- I have concerns about over_under_line being an object

In [34]:
```python
nfl_scores[nfl_scores['over_under_line'].str.contains(' ')]
```

Out[34]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite
2725	1979-12-23	1979	Wildcard	True	Houston Oilers	13.0	7.0	Denver Broncos	TEN	-7.0
2726	1979-12-23	1979	Wildcard	True	Philadelphia Eagles	27.0	17.0	Chicago Bears	PHI	-6.5
2727	1979-12-29	1979	Division	True	San Diego Chargers	14.0	17.0	Houston Oilers	LAC	-8.0
2728	1979-12-29	1979	Division	True	Tampa Bay Buccaneers	24.0	17.0	Philadelphia Eagles	PHI	-4.5
2729	1979-12-30	1979	Division	True	Dallas Cowboys	19.0	21.0	Los Angeles Rams	DAL	-8.5
...	...	...	...	...	...	...	...	...	...	...
4034	1986-01-04	1985	Division	True	Miami Dolphins	24.0	21.0	Cleveland Browns	MIA	-10.5

- Looks like we have 62 rows of data in the over_under_lines that are strings, for simplicity I am simply going to remove them

In [35]:
```python
nfl_scores = nfl_scores[~nfl_scores['over_under_line'].str.contains(' ')]
nfl_scores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 350 to 12946
Data columns (total 16 columns):
schedule_date          10396 non-null datetime64[ns]
schedule_season        10396 non-null int64
schedule_week          10396 non-null object
schedule_playoff       10396 non-null bool
team_home              10396 non-null object
score_home             10396 non-null float64
score_away             10396 non-null float64
team_away              10396 non-null object
team_favorite_id       10396 non-null object
spread_favorite        10396 non-null float64
over_under_line        10396 non-null object
stadium                10396 non-null object
stadium_neutral        10396 non-null bool
weather_temperature    9745 non-null float64
weather_wind_mph       9745 non-null float64
weather_humidity       6289 non-null object
dtypes: bool(2), datetime64[ns](1), float64(5), int64(1), object(7)
```

In [36]:
```python
convert = {'over_under_line': 'float'}
nfl_scores = nfl_scores.astype(convert)
```

In [37]:
```
nfl_scores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 350 to 12946
Data columns (total 16 columns):
schedule_date          10396 non-null datetime64[ns]
schedule_season        10396 non-null int64
schedule_week          10396 non-null object
schedule_playoff       10396 non-null bool
team_home              10396 non-null object
score_home             10396 non-null float64
score_away             10396 non-null float64
team_away              10396 non-null object
team_favorite_id       10396 non-null object
spread_favorite        10396 non-null float64
over_under_line        10396 non-null float64
stadium                10396 non-null object
stadium_neutral        10396 non-null bool
weather_temperature    9745 non-null float64
weather_wind_mph       9745 non-null float64
weather_humidity       6289 non-null object
dtypes: bool(2), datetime64[ns](1), float64(6), int64(1), object(6)
memory usage: 1.2+ MB
```

In [38]:
```
nfl_scores.describe()
```

Out[38]:

	schedule_season	score_home	score_away	spread_favorite	over_under_line	weather_temperature	weather_wind_mph
count	10396.000000	10396.000000	10396.000000	10396.000000	10396.000000	9745.000000	9745.000000
mean	2000.489708	22.707291	20.034340	-5.377982	42.093565	59.894202	7.268958
std	11.961715	10.374221	10.082921	3.431925	4.777090	15.411954	5.719802
min	1967.000000	0.000000	0.000000	-26.500000	28.000000	-6.000000	0.000000
25%	1990.000000	16.000000	13.000000	-7.000000	38.500000	50.000000	1.000000
50%	2001.000000	23.000000	20.000000	-4.500000	42.000000	64.000000	7.000000
75%	2011.000000	30.000000	27.000000	-3.000000	45.000000	72.000000	11.000000
max	2020.000000	62.000000	59.000000	0.000000	63.500000	97.000000	40.000000

Begin Merge of nfl_stadiums and nfl_scores

In [39]:
```python
# Need to make stadium_name equivalent to what it is (stadium) for the nfl_scores df
nfl_stadiums = nfl_stadiums.rename(columns={'stadium_name': 'stadium'})
nfl_stadiums.head()
```

Out[39]:

	stadium	stadium_type	stadium_address	stadium_weather_station_code	STATION	NAME	ELEVATION	latitude	longitu
0	Alamo Dome	indoor	100 Montana St, San Antonio, TX 78203	78203	NaN	NaN	NaN	29.416892	-98.4788
1	Allegiant Stadium	indoor	3333 Al Davis Way, Las Vegas, NV 89118	NaN	NaN	NaN	NaN	NaN	Na
2	Alltel Stadium	outdoor	1 TIAA Bank Field Dr, Jacksonville, FL 32202	NaN	NaN	NaN	NaN	NaN	Na
3	Alumni Stadium	outdoor	Perimeter Rd, Chestnut Hill, MA 02467	2467	NaN	NaN	NaN	NaN	Na
4	Anaheim Stadium	outdoor	2000 E Gene Autry Way, Anaheim, CA 92806	92806	NaN	NaN	NaN	33.799711	-117.8893

In [40]:
```python
nfl = pd.merge(nfl_scores, nfl_stadiums, on='stadium', how='left')
nfl.head(50)
```

6	1974-01-13	1973	Superbowl	True	Miami Dolphins	24.0	7.0	Minnesota Vikings	MIA	-6.5
7	1975-01-12	1974	Superbowl	True	Minnesota Vikings	6.0	16.0	Pittsburgh Steelers	PIT	-3.0
8	1976-01-18	1975	Superbowl	True	Dallas Cowboys	17.0	21.0	Pittsburgh Steelers	PIT	-7.0
9	1977-01-09	1976	Superbowl	True	Minnesota Vikings	14.0	32.0	Oakland Raiders	LVR	-4.0
10	1978-01-15	1977	Superbowl	True	Dallas Cowboys	27.0	10.0	Denver Broncos	DAL	-6.0

In [41]:
```python
nfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 0 to 10395
Data columns (total 24 columns):
schedule_date                   10396 non-null datetime64[ns]
schedule_season                 10396 non-null int64
schedule_week                   10396 non-null object
schedule_playoff                10396 non-null bool
team_home                       10396 non-null object
score_home                      10396 non-null float64
score_away                      10396 non-null float64
team_away                       10396 non-null object
team_favorite_id                10396 non-null object
spread_favorite                 10396 non-null float64
over_under_line                 10396 non-null float64
stadium                         10396 non-null object
stadium_neutral                 10396 non-null bool
weather_temperature             9745 non-null float64
weather_wind_mph                9745 non-null float64
weather_humidity                6289 non-null object
```

Evaluate those pesky stadiums without addresses/types

In [42]:
```
#Seperate the areas of data without info se we can get a easier look at them
null_stad = nfl[nfl['stadium_address'].isnull()]
```

In [43]:
```
null_stad['stadium'].value_counts()
```

Out[43]:
```
FedEx Field                  196
TIAA Bank Field               21
Tottenham Hotspur Stadium      2
Tottenham Stadium              1
Name: stadium, dtype: int64
```

Mostly looks like it effects a few stadiums, we should be able to add in the addresses and types pretty easily

In [44]:
```
null_stad.loc[null_stad['stadium'] == 'FedEx Field', 'stadium_address'] = '1600 Fedex Way, Landover, MD 20785'
null_stad.loc[null_stad['stadium'] == 'Tottenham Hotspur Stadium', 'stadium'] = 'Tottenham Stadium'
null_stad.loc[null_stad['stadium'] == 'Tottenham Stadium', 'stadium_address'] = '782 High Rd, Tottenham, London N17 0
null_stad.loc[null_stad['stadium'] == 'TIAA Bank Field', 'stadium_address'] = '410 Franklin St, Jacksonville, FL 3220
null_stad.head(50)
```

```
/Users/matthewnykaza/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/pandas/core/indexing.py:494: SettingWit
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/pandas-docs/stable/user_guide/indexing.html#returning-
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.obj[item] = s
```

```
In [45]: null_stad.loc[null_stad['stadium'] == 'Tottenham Stadium']
```

Out[45]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite
9682	2018-10-14	2018	6	False	Oakland Raiders	3.0	27.0	Seattle Seahawks	SEA	-3.0
9932	2019-10-06	2019	5	False	Oakland Raiders	24.0	21.0	Chicago Bears	CHI	-6.5
9950	2019-10-13	2019	6	False	Tampa Bay Buccaneers	26.0	37.0	Carolina Panthers	CAR	-2.0

3 rows × 24 columns

Let's try that geopy lat and long software again to see if we can get this thing right!

```
In [46]: null_stad['location'] = null_stad['stadium_address'].apply(geocode)
         null_stad['point'] = null_stad['location'].apply(lambda loc: tuple(loc.point) if loc else None)
         null_stad[['latitude', 'longitude', 'elevation']] = pd.DataFrame(null_stad['point'].tolist(), index=null_stad.index)
```

```
/Users/matthewnykaza/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWithCop
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/pandas-docs/stable/user_guide/indexing.html#returning-
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  """Entry point for launching an IPython kernel.
/Users/matthewnykaza/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWithCop
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/pandas-docs/stable/user_guide/indexing.html#returning-
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

/Users/matthewnykaza/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/pandas/core/frame.py:3509: SettingWith(
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/pandas-docs/stable/user_guide/indexing.html#returning-
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self[k1] = value[k2]
```

In [47]:
```
null_stad.drop(columns=['elevation', 'point', 'location'], inplace=True)
null_stad.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 220 entries, 4083 to 10385
Data columns (total 24 columns):
schedule_date                   220 non-null datetime64[ns]
schedule_season                 220 non-null int64
schedule_week                   220 non-null object
schedule_playoff                220 non-null bool
team_home                       220 non-null object
score_home                      220 non-null float64
score_away                      220 non-null float64
team_away                       220 non-null object
team_favorite_id                220 non-null object
spread_favorite                 220 non-null float64
over_under_line                 220 non-null float64
stadium                         220 non-null object
stadium_neutral                 220 non-null bool
weather_temperature             184 non-null float64
weather_wind_mph                184 non-null float64
weather_humidity                136 non-null object
stadium_type                    0 non-null object
stadium_address                 220 non-null object
stadium_weather_station_code    0 non-null object
STATION                         0 non-null object
NAME                            0 non-null object
ELEVATION                       0 non-null float64
latitude                        220 non-null float64
longitude                       220 non-null float64
dtypes: bool(2), datetime64[ns](1), float64(9), int64(1), object(11)
memory usage: 40.0+ KB

/Users/matthewnykaza/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/pandas/core/frame.py:4117: SettingWith(
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  errors=errors,
```

Now join the dataframes back together

In [48]:
```
len(nfl.columns)
```

Out[48]:
```
24
```

In [49]:
```
len(null_stad.columns)
```

Out[49]:
```
24
```

In [50]:
```
new_nfl = nfl.combine_first(null_stad)
```

In [51]:

```
new_nfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 0 to 10395
Data columns (total 24 columns):
schedule_date                   10396 non-null datetime64[ns]
schedule_season                 10396 non-null int64
schedule_week                   10396 non-null object
schedule_playoff                10396 non-null bool
team_home                       10396 non-null object
score_home                      10396 non-null float64
score_away                      10396 non-null float64
team_away                       10396 non-null object
team_favorite_id                10396 non-null object
spread_favorite                 10396 non-null float64
over_under_line                 10396 non-null float64
stadium                         10396 non-null object
stadium_neutral                 10396 non-null bool
weather_temperature             9745 non-null float64
weather_wind_mph                9745 non-null float64
weather_humidity                6289 non-null object
stadium_type                    10176 non-null object
stadium_address                 10396 non-null object
stadium_weather_station_code    10110 non-null object
STATION                         7954 non-null object
NAME                            7954 non-null object
ELEVATION                       7954 non-null float64
latitude                        9033 non-null float64
longitude                       9033 non-null float64
dtypes: bool(2), datetime64[ns](1), float64(9), int64(1), object(11)
memory usage: 1.8+ MB
```

```
In [52]: nfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 0 to 10395
Data columns (total 24 columns):
schedule_date                   10396 non-null datetime64[ns]
schedule_season                 10396 non-null int64
schedule_week                   10396 non-null object
schedule_playoff                10396 non-null bool
team_home                       10396 non-null object
score_home                      10396 non-null float64
score_away                      10396 non-null float64
team_away                       10396 non-null object
team_favorite_id                10396 non-null object
spread_favorite                 10396 non-null float64
over_under_line                 10396 non-null float64
stadium                         10396 non-null object
stadium_neutral                 10396 non-null bool
weather_temperature             9745 non-null float64
weather_wind_mph                9745 non-null float64
weather_humidity                6289 non-null object
stadium_type                    10176 non-null object
stadium_address                 10176 non-null object
stadium_weather_station_code    10110 non-null object
STATION                         7954 non-null object
NAME                            7954 non-null object
ELEVATION                       7954 non-null float64
latitude                        8813 non-null float64
longitude                       8813 non-null float64
dtypes: bool(2), datetime64[ns](1), float64(9), int64(1), object(11)
memory usage: 2.2+ MB
```

```
In [53]: pd.set_option('display.max_columns', None)
         pd.set_option('display.max_rows', 10)
```

```
In [54]: new_nfl[new_nfl['latitude'].isnull()]
```

...	...	...	...	...	...	...	...	...	...	.
10354	2020-12-26	2020	16	False	Las Vegas Raiders	25.0	26.0	Miami Dolphins	MIA	-2.
10361	2020-12-27	2020	16	False	Los Angeles Chargers	19.0	16.0	Denver Broncos	LAC	-2.
10369	2021-01-03	2020	17	False	Chicago Bears	16.0	35.0	Green Bay Packers	GB	-4.
10375	2021-01-03	2020	17	False	Indianapolis Colts	28.0	14.0	Jacksonville Jaguars	IND	-15.
10377	2021-01-03	2020	17	False	Los Angeles Rams	18.0	7.0	Arizona Cardinals	LAR	-1.

In [55]:

```
new_nfl.loc[new_nfl['stadium'] == 'Soldier Field', 'latitude'] = 41.8623
new_nfl.loc[new_nfl['stadium'] == 'Soldier Field', 'longitude'] = -87.67167
new_nfl.loc[new_nfl['stadium_address'] == '100 S Capitol Ave, Indianapolis, IN 46225', 'longitude'] = -86.164062
new_nfl.loc[new_nfl['stadium_address'] == '100 S Capitol Ave, Indianapolis, IN 46225', 'latitude'] = 39.764705
new_nfl.loc[new_nfl['stadium_address'] == '347 Don Shula Dr, Miami Gardens, FL 33056', 'latitude'] = 25.957564
new_nfl.loc[new_nfl['stadium_address'] == '347 Don Shula Dr, Miami Gardens, FL 33056', 'longitude'] = -80.238302
new_nfl.loc[new_nfl['stadium_address'] == '500 S Capitol Ave, Indianapolis, IN 46225', 'longitude'] = -86.164062
new_nfl.loc[new_nfl['stadium_address'] == '500 S Capitol Ave, Indianapolis, IN 46225', 'latitude'] = 39.764705
new_nfl.loc[new_nfl['stadium_address'] == '1 Georgia Dome Dr, Atlanta, GA 30313', 'latitude'] = 33.757577
new_nfl.loc[new_nfl['stadium_address'] == '1 Georgia Dome Dr, Atlanta, GA 30313', 'longitude'] = -84.400952
new_nfl.loc[new_nfl['stadium_address'] == '1 Everbank Field Dr, Jacksonville, FL 32202', 'longitude'] = -81.637963
new_nfl.loc[new_nfl['stadium_address'] == '1 Everbank Field Dr, Jacksonville, FL 32202', 'latitude'] = 30.322143
new_nfl.loc[new_nfl['stadium_address'] == '1 Avenue of Champions, Clemson, SC 29634', 'latitude'] = 34.679326
new_nfl.loc[new_nfl['stadium_address'] == '1 Avenue of Champions, Clemson, SC 29634', 'longitude'] = -82.844591
new_nfl.loc[new_nfl['stadium_address'] == '1 TIAA Bank Field Dr, Jacksonville, FL 32202', 'latitude'] = 30.328008652
new_nfl.loc[new_nfl['stadium_address'] == '1 TIAA Bank Field Dr, Jacksonville, FL 32202', 'longitude'] = -81.65515899
new_nfl.loc[new_nfl['stadium_weather_station_code'] == 'Mexico City, MX', 'latitude'] = 19.303062439
new_nfl.loc[new_nfl['stadium_weather_station_code'] == 'Mexico City, MX', 'longitude'] = -99.150215149
new_nfl.loc[new_nfl['stadium_address'] == '3333 Al Davis Way, Las Vegas, NV 89118', 'latitude'] = 36.089813
new_nfl.loc[new_nfl['stadium_address'] == '3333 Al Davis Way, Las Vegas, NV 89118', 'longitude'] = -115.183925000
new_nfl.loc[new_nfl['stadium'] == 'SoFi Stadium', 'latitude'] = 33.949903
new_nfl.loc[new_nfl['stadium'] == 'SoFi Stadium', 'longitude'] = -118.343304
new_nfl.loc[new_nfl['stadium'] == 'Tulane Stadium', 'latitude'] = 29.9429822
new_nfl.loc[new_nfl['stadium'] == 'Tulane Stadium', 'longitude'] = -90.1175732
new_nfl.loc[new_nfl['stadium'] == 'Rice Stadium', 'latitude'] = 29.7163407
new_nfl.loc[new_nfl['stadium'] == 'Rice Stadium', 'longitude'] = -95.4096618
```

In [56]:

```
new_nfl.info()
```

```
score_away                      10396 non-null float64
team_away                       10396 non-null object
team_favorite_id                10396 non-null object
spread_favorite                 10396 non-null float64
over_under_line                 10396 non-null float64
stadium                         10396 non-null object
stadium_neutral                 10396 non-null bool
weather_temperature             9745 non-null float64
weather_wind_mph                9745 non-null float64
weather_humidity                6289 non-null object
stadium_type                    10176 non-null object
stadium_address                 10396 non-null object
stadium_weather_station_code    10110 non-null object
STATION                         7954 non-null object
NAME                            7954 non-null object
ELEVATION                       7954 non-null float64
latitude                        10396 non-null float64
longitude                       10396 non-null float64
dtypes: bool(2), datetime64[ns](1), float64(9), int64(1), object(11)
memory usage: 1.8+ MB
```

In [57]:

```
new_nfl[new_nfl['stadium_type'].isnull()]
```

Out[57]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorit
4083	1997-09-14	1997	3	False	Washington Redskins	19.0	13.0	Arizona Cardinals	WAS	-8.
4109	1997-09-28	1997	5	False	Washington Redskins	24.0	12.0	Jacksonville Jaguars	WAS	-1.
4135	1997-10-13	1997	7	False	Washington Redskins	21.0	16.0	Dallas Cowboys	DAL	-3.
4160	1997-10-26	1997	9	False	Washington Redskins	17.0	20.0	Baltimore Ravens	WAS	-6.
4190	1997-11-09	1997	11	False	Washington Redskins	30.0	7.0	Detroit Lions	WAS	-4.

In [58]:

```
new_nfl.loc[new_nfl['stadium'] == 'FedEx Field', 'stadium_type'] = 'outdoor'
new_nfl.loc[new_nfl['stadium'] == 'TIAA Bank Field', 'stadium_type'] = 'outdoor'
new_nfl.loc[new_nfl['stadium'] == 'Tottenham Stadium', 'stadium_type'] = 'outdoor'
new_nfl.loc[new_nfl['stadium'] == 'Tottenham Hotspur Stadium', 'stadium_type'] = 'outdoor'
```

In [59]:

```
new_nfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 0 to 10395
Data columns (total 24 columns):
schedule_date                   10396 non-null datetime64[ns]
schedule_season                 10396 non-null int64
schedule_week                   10396 non-null object
schedule_playoff                10396 non-null bool
team_home                       10396 non-null object
score_home                      10396 non-null float64
score_away                      10396 non-null float64
team_away                       10396 non-null object
team_favorite_id                10396 non-null object
spread_favorite                 10396 non-null float64
over_under_line                 10396 non-null float64
stadium                         10396 non-null object
stadium_neutral                 10396 non-null bool
weather_temperature             9745 non-null float64
weather_wind_mph                9745 non-null float64
weather_humidity                6289 non-null object
stadium_type                    10396 non-null object
stadium_address                 10396 non-null object
stadium_weather_station_code    10110 non-null object
STATION                         7954 non-null object
NAME                            7954 non-null object
ELEVATION                       7954 non-null float64
latitude                        10396 non-null float64
longitude                       10396 non-null float64
dtypes: bool(2), datetime64[ns](1), float64(9), int64(1), object(11)
memory usage: 1.8+ MB
```

NaNs left

- weather_temperature and weather_humidity we will allow the SimpleImputer we employ later to handle that as the mean values of both are within norma found on the internet
-
  - We do need to make sure that weather_humidity is in a float to make sure it can be appropriately handled by our metrics
- weather_wind we will make sure is 0 for indoor stadiums, as the mean would not be valid here. For the rest the mean will be fine
- Not worried about stadium_weather_station_code, STATION, NAME, and ELEVATION as we will drop these. We were hoping to get real weather data, bu result these are not needed.

```
In [60]: new_nfl.loc[(new_nfl.stadium_type == 'indoor'), 'weather_wind_mph'] = 0
```

```
In [61]: new_nfl[new_nfl['weather_wind_mph'].isnull()]
```

Out[61]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorit
236	1980-01-20	1979	Superbowl	True	Los Angeles Rams	19.0	31.0	Pittsburgh Steelers	PIT	-10.
813	1983-01-30	1982	Superbowl	True	Miami Dolphins	17.0	27.0	Washington Redskins	MIA	-3.
1038	1984-01-22	1983	Superbowl	True	Washington Redskins	9.0	38.0	Los Angeles Raiders	WAS	-3.
1263	1985-01-20	1984	Superbowl	True	San Francisco 49ers	38.0	16.0	Miami Dolphins	SF	-3.
1713	1986-12-28	1986	Wildcard	True	New York Jets	35.0	15.0	Kansas City Chiefs	NYJ	-3.
...	...	...	...	...	...	...	...	...	...	.
10390	2021-01-16	2020	Division	True	Green Bay Packers	32.0	18.0	Los Angeles Rams	GB	-7.
10391	2021-01-17	2020	Division	True	Kansas City Chiefs	22.0	17.0	Cleveland Browns	KC	-8.
10393	2021-01-24	2020	Conference	True	Green Bay Packers	26.0	31.0	Tampa Bay Buccaneers	GB	-3.
10394	2021-01-24	2020	Conference	True	Kansas City Chiefs	38.0	24.0	Buffalo Bills	KC	-3.
10395	2021-02-07	2020	Superbowl	True	Tampa Bay Buccaneers	31.0	9.0	Kansas City Chiefs	KC	-3.

641 rows × 24 columns

In [62]:
```
convert2 = {'weather_humidity': 'float'}
new_nfl = new_nfl.astype(convert2)
```

In [63]:
```
new_nfl[new_nfl.weather_humidity.isnull()]
```

10391	2021-01-17	2020	Division	True	Kansas City Chiefs	22.0	17.0	Cleveland Browns	KC	-8.
10392	2021-01-17	2020	Division	True	New Orleans Saints	20.0	30.0	Tampa Bay Buccaneers	NO	-2.
10393	2021-01-24	2020	Conference	True	Green Bay Packers	26.0	31.0	Tampa Bay Buccaneers	GB	-3.
10394	2021-01-24	2020	Conference	True	Kansas City Chiefs	38.0	24.0	Buffalo Bills	KC	-3.
10395	2021-02-07	2020	Superbowl	True	Tampa Bay Buccaneers	31.0	9.0	Kansas City Chiefs	KC	-3.

4107 rows × 24 columns

In [64]:
```
new_nfl.loc[(new_nfl.stadium_type == 'indoor'), 'weather_humidity'] = 50
```

In [65]:
```
new_nfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 0 to 10395
Data columns (total 24 columns):
schedule_date                   10396 non-null datetime64[ns]
schedule_season                 10396 non-null int64
schedule_week                   10396 non-null object
schedule_playoff                10396 non-null bool
team_home                       10396 non-null object
score_home                      10396 non-null float64
score_away                      10396 non-null float64
team_away                       10396 non-null object
team_favorite_id                10396 non-null object
spread_favorite                 10396 non-null float64
over_under_line                 10396 non-null float64
stadium                         10396 non-null object
stadium_neutral                 10396 non-null bool
weather_temperature             9745 non-null float64
weather_wind_mph                9755 non-null float64
weather_humidity                8197 non-null float64
stadium_type                    10396 non-null object
```

In [66]:
```
new_nfl.drop(columns=['STATION', 'stadium_weather_station_code', 'NAME', 'ELEVATION'], inplace=True)
```

In [67]:

```
new_nfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 0 to 10395
Data columns (total 20 columns):
schedule_date          10396 non-null datetime64[ns]
schedule_season        10396 non-null int64
schedule_week          10396 non-null object
schedule_playoff       10396 non-null bool
team_home              10396 non-null object
score_home             10396 non-null float64
score_away             10396 non-null float64
team_away              10396 non-null object
team_favorite_id       10396 non-null object
spread_favorite        10396 non-null float64
over_under_line        10396 non-null float64
stadium                10396 non-null object
stadium_neutral        10396 non-null bool
weather_temperature    9745 non-null float64
weather_wind_mph       9755 non-null float64
weather_humidity       8197 non-null float64
stadium_type           10396 non-null object
stadium_address        10396 non-null object
latitude               10396 non-null float64
longitude              10396 non-null float64
dtypes: bool(2), datetime64[ns](1), float64(9), int64(1), object(7)
memory usage: 1.5+ MB
```

Code our target

Our target variable is weather or not the over was reached, an over will be coded as a 1 and a under/push will be coded as a 0

We are also going to work towards adding division and conference information to our dataset, as these may be quite useless for our model later on

In [68]:

```
# rename team_home so we can get a merge goign with the nfl_teams Dataframe
new_nfl.rename(columns={'team_home': 'team_name'}, inplace=True)
new_nfl.head()
```

Out[68]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_name	score_home	score_away	team_away	team_favorite_id	spread_favorite
0	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.5
1	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.0
2	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.0
3	1971-01-17	1970	Superbowl	True	Baltimore Colts	16.0	13.0	Dallas Cowboys	IND	-2.5
4	1972-01-16	1971	Superbowl	True	Dallas Cowboys	24.0	3.0	Miami Dolphins	DAL	-6.0

```
In [69]: # view and delete unneccesary columns in nfl_teams
         nfl_teams.head()
         nfl_teams.drop(columns=['team_name_short', 'team_id', 'team_id_pfr'], inplace=True)
```

```
In [70]: nfl_teams.head()
```

```
Out[70]:
             team_name  team_conference  team_division  team_conference_pre2002  team_division_pre2002
   0  Arizona Cardinals             NFC       NFC West                      NFC               NFC West
   1  Phoenix Cardinals             NFC            NaN                      NFC               NFC East
   2  St. Louis Cardinals           NFC            NaN                      NFC               NFC East
   3     Atlanta Falcons            NFC      NFC South                      NFC               NFC West
   4   Baltimore Ravens             AFC      AFC North                      AFC            AFC Central
```

```
In [71]: # Merge so we can get the team_division and conference information into this dataframe
         nfl_2 = pd.merge(new_nfl, nfl_teams, how='left', on=['team_name'])
```

In [72]:

```
nfl_2.head(20)
```

Out[72]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_name	score_home	score_away	team_away	team_favorite_id	spread_favorite
0	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.5
1	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.0
2	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.0
3	1971-01-17	1970	Superbowl	True	Baltimore Colts	16.0	13.0	Dallas Cowboys	IND	-2.5
4	1972-01-16	1971	Superbowl	True	Dallas Cowboys	24.0	3.0	Miami Dolphins	DAL	-6.0
...	...	...	...	...	...	...	...	...	...	...
15	1979-09-02	1979	1	False	Denver Broncos	10.0	0.0	Cincinnati Bengals	DEN	-3.0
16	1979-09-02	1979	1	False	Kansas City Chiefs	14.0	0.0	Baltimore Colts	KC	-1.0
17	1979-09-02	1979	1	False	Los Angeles Rams	17.0	24.0	Oakland Raiders	LAR	-4.0
18	1979-09-02	1979	1	False	Minnesota Vikings	28.0	22.0	San Francisco 49ers	MIN	-7.0
19	1979-09-02	1979	1	False	New Orleans Saints	34.0	40.0	Atlanta Falcons	NO	-5.0

20 rows × 24 columns

In [73]:

```
#Now rename appropriate home team columns in this new dataframe
nfl_2.rename(columns={'team_name': 'team_home', 'team_conference': 'team_home_conference', 'team_division':'team_home
nfl_2.head()
```

Out[73]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite	o
0	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.5	
1	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.0	
2	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.0	
3	1971-01-17	1970	Superbowl	True	Baltimore Colts	16.0	13.0	Dallas Cowboys	IND	-2.5	
4	1972-01-16	1971	Superbowl	True	Dallas Cowboys	24.0	3.0	Miami Dolphins	DAL	-6.0	

In [74]:

```
#Do same with away teams
nfl_2.rename(columns={'team_away': 'team_name'}, inplace=True)
nfl_3 = pd.merge(nfl_2, nfl_teams, how='left', on=['team_name'])
nfl_3.rename(columns={'team_name': 'team_away', 'team_conference': 'team_away_conference', 'team_division': 'team_awa
nfl_3.head()
```

Out[74]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite	o
0	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.5	
1	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.0	
2	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.0	
3	1971-01-17	1970	Superbowl	True	Baltimore Colts	16.0	13.0	Dallas Cowboys	IND	-2.5	
4	1972-01-16	1971	Superbowl	True	Dallas Cowboys	24.0	3.0	Miami Dolphins	DAL	-6.0	

In [75]:

```
# Split DF into pre-2002 and post-2002 for the sake of correctly adding team division and conference
pre_2002 = nfl_3[(nfl_3['schedule_season'] < 2002)]
post_2002 = nfl_3[(nfl_3['schedule_season'] >= 2002)]

post_2002.head()
```

Out[75]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite
5321	2002-09-05	2002	1	False	New York Giants	13.0	16.0	San Francisco 49ers	SF	-4.0
5322	2002-09-08	2002	1	False	Buffalo Bills	31.0	37.0	New York Jets	NYJ	-3.0
5323	2002-09-08	2002	1	False	Carolina Panthers	10.0	7.0	Baltimore Ravens	PICK	0.0
5324	2002-09-08	2002	1	False	Chicago Bears	27.0	23.0	Minnesota Vikings	CHI	-4.5
5325	2002-09-08	2002	1	False	Cincinnati Bengals	6.0	34.0	San Diego Chargers	CIN	-3.0

In [76]:

```
pre_2002.drop(columns=['team_home_conference', 'team_home_division', 'team_away_conference', 'team_away_division'], i
pre_2002.head()
```

```
/Users/matthewnykaza/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/pandas/core/frame.py:4117: SettingWith(
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  errors=errors,
```

Out[76]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite
0	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.5
1	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.0
2	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.0
					Baltimore			Dallas		

In [77]:

```
pre_2002.rename(columns={'team_home_conference_pre2002': 'team_home_conference', 'team_home_division_pre2002': 'team_
pre_2002.head()
```

```
/Users/matthewnykaza/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/pandas/core/frame.py:4238: SettingWithC
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  return super().rename(**kwargs)
```

Out[77]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite	o
0	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.5	
1	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.0	
2	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.0	
3	1971-01-17	1970	Superbowl	True	Baltimore Colts	16.0	13.0	Dallas Cowboys	IND	-2.5	
4	1972-01-16	1971	Superbowl	True	Dallas Cowboys	24.0	3.0	Miami Dolphins	DAL	-6.0	

In [78]:

```
#Do the same for post 2001
post_2002.head()
```

Out[78]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite
5321	2002-09-05	2002	1	False	New York Giants	13.0	16.0	San Francisco 49ers	SF	-4.0
5322	2002-09-08	2002	1	False	Buffalo Bills	31.0	37.0	New York Jets	NYJ	-3.0
5323	2002-09-08	2002	1	False	Carolina Panthers	10.0	7.0	Baltimore Ravens	PICK	0.0
5324	2002-09-08	2002	1	False	Chicago Bears	27.0	23.0	Minnesota Vikings	CHI	-4.5
5325	2002-09-08	2002	1	False	Cincinnati Bengals	6.0	34.0	San Diego Chargers	CIN	-3.0

In [79]:

```
post_2002.drop(columns=['team_home_conference_pre2002', 'team_home_division_pre2002', 'team_away_conference_pre2002',
post_2002.head()
```

```
/Users/matthewnykaza/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/pandas/core/frame.py:4117: SettingWithC
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  errors=errors,
```

Out[79]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite
5321	2002-09-05	2002	1	False	New York Giants	13.0	16.0	San Francisco 49ers	SF	-4.0
5322	2002-09-08	2002	1	False	Buffalo Bills	31.0	37.0	New York Jets	NYJ	-3.0
5323	2002-09-08	2002	1	False	Carolina Panthers	10.0	7.0	Baltimore Ravens	PICK	0.0
5324	2002-09-08	2002	1	False	Chicago Bears	27.0	23.0	Minnesota Vikings	CHI	-4.5
5325	2002-09-08	2002	1	False	Cincinnati Bengals	6.0	34.0	San Diego Chargers	CIN	-3.0

In [80]:

```
#Put pre and post back together
all_nfl = pre_2002.append(post_2002)
all_nfl.head()
```

Out[80]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite	o
0	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.5	
1	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.0	
2	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.0	
3	1971-01-17	1970	Superbowl	True	Baltimore Colts	16.0	13.0	Dallas Cowboys	IND	-2.5	
4	1972-01-16	1971	Superbowl	True	Dallas Cowboys	24.0	3.0	Miami Dolphins	DAL	-6.0	

In [81]:

```
#Look to have a few missing divisions, but overall I'm not too concerned about that, just gonna make them 'unknown' a
all_nfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 0 to 10395
Data columns (total 24 columns):
schedule_date           10396 non-null datetime64[ns]
schedule_season         10396 non-null int64
schedule_week           10396 non-null object
schedule_playoff        10396 non-null bool
team_home               10396 non-null object
score_home              10396 non-null float64
score_away              10396 non-null float64
team_away               10396 non-null object
team_favorite_id        10396 non-null object
spread_favorite         10396 non-null float64
over_under_line         10396 non-null float64
stadium                 10396 non-null object
stadium_neutral         10396 non-null bool
weather_temperature     9745 non-null float64
weather_wind_mph        9755 non-null float64
weather_humidity        8197 non-null float64
stadium_type            10396 non-null object
stadium_address         10396 non-null object
latitude                10396 non-null float64
longitude               10396 non-null float64
team_home_conference    10396 non-null object
team_home_division      10283 non-null object
team_away_conference    10396 non-null object
team_away_division      10282 non-null object
dtypes: bool(2), datetime64[ns](1), float64(9), int64(1), object(11)
memory usage: 1.8+ MB
```

In [82]:

```
all_nfl.loc[all_nfl['team_home_division'].isnull(), 'team_home_division'] = 'unknown'
all_nfl.loc[all_nfl['team_away_division'].isnull(), 'team_away_division'] = 'unknown'
all_nfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 0 to 10395
Data columns (total 24 columns):
schedule_date           10396 non-null datetime64[ns]
schedule_season         10396 non-null int64
schedule_week           10396 non-null object
schedule_playoff        10396 non-null bool
team_home               10396 non-null object
score_home              10396 non-null float64
score_away              10396 non-null float64
team_away               10396 non-null object
team_favorite_id        10396 non-null object
spread_favorite         10396 non-null float64
over_under_line         10396 non-null float64
stadium                 10396 non-null object
stadium_neutral         10396 non-null bool
weather_temperature     9745 non-null float64
weather_wind_mph        9755 non-null float64
weather_humidity        8197 non-null float64
stadium_type            10396 non-null object
stadium_address         10396 non-null object
latitude                10396 non-null float64
longitude               10396 non-null float64
team_home_conference    10396 non-null object
team_home_division      10396 non-null object
team_away_conference    10396 non-null object
team_away_division      10396 non-null object
dtypes: bool(2), datetime64[ns](1), float64(9), int64(1), object(11)
memory usage: 1.8+ MB
```

In [83]:

```
all_nfl.head()
```

Out[83]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite	o
0	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.5	
1	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.0	
2	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.0	
3	1971-01-17	1970	Superbowl	True	Baltimore Colts	16.0	13.0	Dallas Cowboys	IND	-2.5	
4	1972-01-16	1971	Superbowl	True	Dallas Cowboys	24.0	3.0	Miami Dolphins	DAL	-6.0	

```
In [84]: def conf(row):
             if row['team_home_conference'] == row['team_away_conference']:
                 val = 1
             else:
                 val = 0
             return val
```

```
In [85]: def divi(row):
             if row['team_home_division'] == row['team_away_division']:
                 val = 1
             else:
                 val = 0
             return val
```

```
In [86]: all_nfl['intra_conference'] = all_nfl.apply(conf, axis=1)
```

```
In [87]: all_nfl['intra_division'] = all_nfl.apply(divi, axis=1)
```

```
In [88]: all_nfl.head(-5)
```

Out[88]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorit
0	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.
1	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.
2	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.
3	1971-01-17	1970	Superbowl	True	Baltimore Colts	16.0	13.0	Dallas Cowboys	IND	-2.
4	1972-01-16	1971	Superbowl	True	Dallas Cowboys	24.0	3.0	Miami Dolphins	DAL	-6.
...	...	...	...	...	...	...	...	...	...	.
10386	2021-01-10	2020	Wildcard	True	New Orleans Saints	21.0	9.0	Chicago Bears	NO	-11.
10387	2021-01-10	2020	Wildcard	True	Pittsburgh Steelers	37.0	48.0	Cleveland Browns	PIT	-5.
10388	2021-01-10	2020	Wildcard	True	Tennessee Titans	13.0	20.0	Baltimore Ravens	BAL	-3.
10389	2021-01-16	2020	Division	True	Buffalo Bills	17.0	3.0	Baltimore Ravens	BUF	-2.
10390	2021-01-16	2020	Division	True	Green Bay Packers	32.0	18.0	Los Angeles Rams	GB	-7.

10391 rows × 26 columns

In [ ]:

```

```

In [89]:

```
#create total, which will be the start of creating our target
all_nfl['total'] = all_nfl['score_home'] + all_nfl['score_away']
all_nfl.head()
```

Out[89]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorite
0	1968-01-14	1967	Superbowl	True	Green Bay Packers	33.0	14.0	Oakland Raiders	GB	-13.5
1	1969-01-12	1968	Superbowl	True	Baltimore Colts	7.0	16.0	New York Jets	IND	-18.0
2	1970-01-11	1969	Superbowl	True	Kansas City Chiefs	23.0	7.0	Minnesota Vikings	MIN	-12.0
3	1971-01-17	1970	Superbowl	True	Baltimore Colts	16.0	13.0	Dallas Cowboys	IND	-2.5
4	1972-01-16	1971	Superbowl	True	Dallas Cowboys	24.0	3.0	Miami Dolphins	DAL	-6.0

In [90]:

```
# Create Target Variable for real I promise
conditional = [
    (all_nfl['total'] > all_nfl['over_under_line']),
    (all_nfl['total'] <= all_nfl['total'])]
valuez = [1, 0]
all_nfl['hit_over'] = np.select(conditional, valuez)
all_nfl.head()
```

Out[90]:

weather_humidity	stadium_type	stadium_address	latitude	longitude	team_home_conference	team_home_division	team_away_conference	team_away_divisio
74.0	outdoor	1501 NW 3rd St, Miami, FL 33125	25.776346	-80.219909	NFC	NFC Central	AFC	AFC We
80.0	outdoor	1501 NW 3rd St, Miami, FL 33125	25.776346	-80.219909	AFC	AFC East	NFC	AFC Ea
84.0	outdoor	Willow St. & Audubon Blvd., New Orleans, LA 70118	29.942982	-90.117573	AFC	AFC West	NFC	NFC Centr
60.0	outdoor	1501 NW 3rd St, Miami, FL 33125	25.776346	-80.219909	AFC	AFC East	NFC	NFC Ea
40.0	outdoor	Willow St. & Audubon Blvd., New Orleans, LA 70118	29.942982	-90.117573	NFC	NFC East	AFC	AFC Ea

In [91]:
```python
#Looks like we have a really good mix here, not to biased on one way or the other
all_nfl['hit_over'].value_counts()
```

Out[91]:
```
0    5363
1    5033
Name: hit_over, dtype: int64
```

In [92]:
```python
#Copy dataframe so I don't ruin any data unneccesarily
roll_df = all_nfl
```

In [93]:
```python
#Getting a rolling average of points scored for each team in the league
roll = pd.concat(
    [
    roll_df[['schedule_date', 'team_home', 'score_home']].rename(
    columns={'team_home': 'team', 'score_home':'score'},
    ),
    roll_df[['schedule_date', 'team_away', 'score_away']].rename(
    columns={'team_away': 'team', 'score_away': 'score'},
    ),
    ],ignore_index = True,).sort_values('schedule_date')
```

In [94]:
```python
team_dfs = [
    roll[
        roll['team']==team
    ].set_index('schedule_date') for team in roll['team'].unique()
]
```

In [95]:
```python
for team_df in team_dfs:
    team_df['last_5'] = team_df['score'].shift(1).rolling(window=5, min_periods=1).mean()
    print(team_df, '\n')
```

```
                            team  score     last_5
schedule_date
1968-01-14     Green Bay Packers   33.0        NaN
1979-09-02     Green Bay Packers    3.0  33.000000
1979-09-09     Green Bay Packers   28.0  18.000000
1979-09-16     Green Bay Packers   10.0  21.333333
1979-09-23     Green Bay Packers   21.0  18.500000
...                          ...    ...        ...
2020-12-19     Green Bay Packers   24.0  31.400000
2020-12-27     Green Bay Packers   40.0  31.400000
2021-01-03     Green Bay Packers   35.0  33.200000
2021-01-16     Green Bay Packers   32.0  32.000000
2021-01-24     Green Bay Packers   26.0  32.400000

[704 rows x 3 columns]

                          team  score     last_5
schedule_date
1968-01-14     Oakland Raiders   14.0        NaN
```

In [96]:
```python
last5_df = pd.concat(team_dfs)
```

In [97]:
```python
last5_df.reset_index(inplace=True)
```

```
In [98]: last5_df.tail(16)
```

```
Out[98]:
       schedule_date               team  score     last_5
20776     2020-09-13  Las Vegas Raiders   34.0        NaN
20777     2020-09-21  Las Vegas Raiders   34.0  34.000000
20778     2020-09-27  Las Vegas Raiders   20.0  34.000000
20779     2020-10-04  Las Vegas Raiders   23.0  29.333333
20780     2020-10-11  Las Vegas Raiders   40.0  27.750000
  ...            ...                ...    ...        ...
20787     2020-12-06  Las Vegas Raiders   31.0  24.200000
20788     2020-12-13  Las Vegas Raiders   27.0  27.200000
20789     2020-12-17  Las Vegas Raiders   27.0  26.400000
20790     2020-12-26  Las Vegas Raiders   25.0  24.400000
20791     2021-01-03  Las Vegas Raiders   32.0  23.200000

16 rows × 4 columns
```

```
In [99]: roll_df.rename(columns={'team_home': 'team'}, inplace=True)
         roll_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 0 to 10395
Data columns (total 28 columns):
schedule_date           10396 non-null datetime64[ns]
schedule_season         10396 non-null int64
schedule_week           10396 non-null object
schedule_playoff        10396 non-null bool
team                    10396 non-null object
score_home              10396 non-null float64
score_away              10396 non-null float64
team_away               10396 non-null object
team_favorite_id        10396 non-null object
spread_favorite         10396 non-null float64
over_under_line         10396 non-null float64
stadium                 10396 non-null object
stadium_neutral         10396 non-null bool
weather_temperature     9745 non-null float64
weather_wind_mph        9755 non-null float64
weather_humidity        8197 non-null float64
stadium_type            10396 non-null object
stadium_address         10396 non-null object
latitude                10396 non-null float64
longitude               10396 non-null float64
team_home_conference    10396 non-null object
team_home_division      10396 non-null object
team_away_conference    10396 non-null object
team_away_division      10396 non-null object
intra_conference        10396 non-null int64
intra_division          10396 non-null int64
total                   10396 non-null float64
hit_over                10396 non-null int64
dtypes: bool(2), datetime64[ns](1), float64(10), int64(4), object(11)
memory usage: 2.2+ MB
```

```
In [100]: new_nfl = roll_df.merge(last5_df, how='inner', left_on=['schedule_date', 'team'], right_on=['schedule_date', 'team'])
In [101]: last5_df.loc[last5_df['team']=='Seattle Seahawks']
```

```
Out[101]:
       schedule_date              team  score      last_5
13860     1979-09-02  Seattle Seahawks   16.0         NaN
13861     1979-09-09  Seattle Seahawks   10.0   16.000000
13862     1979-09-16  Seattle Seahawks   27.0   13.000000
13863     1979-09-23  Seattle Seahawks   34.0   17.666667
13864     1979-09-30  Seattle Seahawks    6.0   21.750000
  ...            ...               ...    ...         ...
14546     2020-12-13  Seattle Seahawks   40.0   22.600000
14547     2020-12-20  Seattle Seahawks   20.0   23.800000
14548     2020-12-27  Seattle Seahawks   20.0   24.600000
14549     2021-01-03  Seattle Seahawks   26.0   23.000000
14550     2021-01-09  Seattle Seahawks   20.0   23.600000

691 rows × 4 columns
```

```
In [102]: new_nfl.loc[new_nfl['team']=='Seattle Seahawks']
```

```
                                                 Seattle                  Houston
  120  1979-10-21  1979   8  False  Seahawks  34.0  14.0    Oilers  TEN   -2.0

                                                                             Los
                                                 Seattle                 Angeles
  149  1979-11-04  1979  10  False  Seahawks   0.0  24.0      Rams  SEA   -3.0

  ...         ...   ...  ...   ...       ...   ...   ...       ...  ...    ...

                                                 Seattle                 Arizona
10274  2020-11-19  2020  11  False  Seahawks  28.0  21.0 Cardinals  SEA   -3.0

                                                 Seattle                New York
10314  2020-12-06  2020  13  False  Seahawks  12.0  17.0    Giants  SEA  -11.0

                                                 Seattle                New York
10332  2020-12-13  2020  14  False  Seahawks  40.0   3.0      Jets  SEA  -16.5

                                                                             Los
```

In [103]:

```
#Checking amount of NAN data, makes sense considering anytime a team has a their first 5 games, the first 4 of whice
#will not have data
new_nfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10396 entries, 0 to 10395
Data columns (total 30 columns):
schedule_date           10396 non-null datetime64[ns]
schedule_season         10396 non-null int64
schedule_week           10396 non-null object
schedule_playoff        10396 non-null bool
team                    10396 non-null object
score_home              10396 non-null float64
score_away              10396 non-null float64
team_away               10396 non-null object
team_favorite_id        10396 non-null object
spread_favorite         10396 non-null float64
over_under_line         10396 non-null float64
stadium                 10396 non-null object
stadium_neutral         10396 non-null bool
weather_temperature     9745 non-null float64
weather_wind_mph        9755 non-null float64
weather_humidity        8197 non-null float64
stadium_type            10396 non-null object
stadium_address         10396 non-null object
latitude                10396 non-null float64
longitude               10396 non-null float64
team_home_conference    10396 non-null object
team_home_division      10396 non-null object
team_away_conference    10396 non-null object
team_away_division      10396 non-null object
intra_conference        10396 non-null int64
intra_division          10396 non-null int64
total                   10396 non-null float64
hit_over                10396 non-null int64
score                   10396 non-null float64
last_5                  10377 non-null float64
dtypes: bool(2), datetime64[ns](1), float64(12), int64(4), object(11)
memory usage: 2.3+ MB
```

In [104]:
```
new_nfl.rename(columns={'team':'team_home', 'last_5':'last_5_home'},inplace = True)
new_nfl.drop(columns=['score'], inplace=True)
new_nfl.tail()
```

Out[104]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	team_away	team_favorite_id	spread_favorit
10391	2021-01-17	2020	Division	True	Kansas City Chiefs	22.0	17.0	Cleveland Browns	KC	-8.
10392	2021-01-17	2020	Division	True	New Orleans Saints	20.0	30.0	Tampa Bay Buccaneers	NO	-2.
10393	2021-01-24	2020	Conference	True	Green Bay Packers	26.0	31.0	Tampa Bay Buccaneers	GB	-3.
10394	2021-01-24	2020	Conference	True	Kansas City Chiefs	38.0	24.0	Buffalo Bills	KC	-3.
10395	2021-02-07	2020	Superbowl	True	Tampa Bay Buccaneers	31.0	9.0	Kansas City Chiefs	KC	-3.

In [105]:
```
#Now do the same thing for away teams
new_nfl.rename(columns={'team_away': 'team'}, inplace=True)
```

In [106]:
```
new_nfl = new_nfl.merge(last5_df, how='inner', left_on=['schedule_date', 'team'], right_on=['schedule_date', 'team'])
```

In [107]:
```
#Do not want the model to know the scores of the game before hand, that would be cheating
#Any data that includes information about the score in that game will be dropped
#Not worried about stadium address, pretty arbitrary and we have other data that represents it i.e stadium name
#We'll keep the conference info for now, but I see a world where that could be dropped
new_nfl.drop(columns=['score_home','score_away', 'stadium_address', 'total', 'score'], inplace=True)
```

In [108]:
```
new_nfl.rename(columns={'team':'team_away', 'last_5': 'last_5_away'},inplace = True)
```

```
In [109]: new_nfl.tail()
```

Out[109]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	team_away	team_favorite_id	spread_favorite	over_under_line	stad
10391	2021-01-17	2020	Division	True	Kansas City Chiefs	Cleveland Browns	KC	-8.0	56.0	Arrowh Stad
10392	2021-01-17	2020	Division	True	New Orleans Saints	Tampa Bay Buccaneers	NO	-2.5	53.0	Merced B Superdo
10393	2021-01-24	2020	Conference	True	Green Bay Packers	Tampa Bay Buccaneers	GB	-3.0	53.0	Lamb F
10394	2021-01-24	2020	Conference	True	Kansas City Chiefs	Buffalo Bills	KC	-3.0	55.0	Arrowh Stad
10395	2021-02-07	2020	Superbowl	True	Tampa Bay Buccaneers	Kansas City Chiefs	KC	-3.0	56.0	Raym Jai Stad

```
In [110]: #Check work
          new_nfl[(new_nfl == 'Seattle Seahawks').any(axis=1)]
```

```
/Users/matthewnykaza/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/pandas/core/ops/__init__.py:1115: Futur
iled; returning scalar instead, but in the future will perform elementwise comparison
  result = method(y)
```

Out[110]:

le_season	schedule_week	schedule_playoff	team_home	team_away	team_favorite_id	spread_favorite	over_under_line	stadium	stadium_neutral	weather_t
1979	1	False	Seattle Seahawks	San Diego Chargers	SEA	-2.0	42.5	Seattle Kingdome	False	
1979	2	False	Miami Dolphins	Seattle Seahawks	MIA	-7.0	40.5	Orange Bowl	False	
1979	3	False	Seattle Seahawks	Oakland Raiders	SEA	-3.0	44.0	Seattle Kingdome	False	
1979	4	False	Denver Broncos	Seattle Seahawks	DEN	-6.0	37.0	Mile High Stadium	False	
1979	5	False	Seattle Seahawks	Kansas City Chiefs	SEA	-6.0	43.0	Seattle Kingdome	False	
...	...	...	...	...	...	...	...	...	...	
2020	14	False	Seattle Seahawks	New York Jets	SEA	-16.5	49.0	CenturyLink Field	False	
2020	15	False	Washington Football Team	Seattle Seahawks	SEA	-6.0	44.0	FedEx Field	False	
2020	16	False	Seattle Seahawks	Los Angeles Rams	SEA	-1.5	48.0	CenturyLink Field	False	
2020	17	False	San Francisco 49ers	Seattle Seahawks	SEA	-7.0	45.0	Levi's Stadium	False	
2020	Wildcard	True	Seattle Seahawks	Los Angeles Rams	SEA	-3.0	42.5	CenturyLink Field	False	

```
In [142]: #Make sure schedule_week is a numerical (will take multiple steps)
          new_nfl['schedule_week'] = new_nfl['schedule_week'].replace({'18':'17'})
```

```
In [148]: new_nfl['schedule_week'] = new_nfl['schedule_week'].replace({'Wildcard':'18', 'WildCard': '18', 'Division': '19',
                                                                      'Conference':'20', 'Superbowl': '21', 'SuperBowl': '21'}
```

```
In [151]: new_nfl['schedule_week'] = new_nfl['schedule_week'].astype('int32')
```

```
In [ ]:
```

```
In [192]: new_nfl.info()
```

```
team_home                 10361 non-null object
team_away                 10361 non-null object
team_favorite_id          10361 non-null object
spread_favorite           10361 non-null float64
over_under_line           10361 non-null float64
stadium_neutral           10361 non-null int64
weather_temperature       9712 non-null float64
weather_wind_mph          9722 non-null float64
weather_humidity          8166 non-null float64
stadium_type              10361 non-null object
latitude                  10361 non-null float64
longitude                 10361 non-null float64
intra_conference          10361 non-null int64
intra_division            10361 non-null int64
hit_over                  10361 non-null int64
last_5_home               10361 non-null float64
last_5_away               10361 non-null float64
estimated_total           10361 non-null float64
dtypes: float64(10), int32(1), int64(6), object(4)
memory usage: 1.7+ MB
```

```
In [161]: #Create new column based on expected point total (based on last5 scores added)
          new_nfl['estimated_total'] = new_nfl['last_5_home'] + new_nfl['last_5_away']
          new_nfl.tail()
```

Out[161]:

under_line	stadium_neutral	weather_temperature	weather_wind_mph	weather_humidity	stadium_type	latitude	longitude	intra_conference	intra_division	hi
56.0	0	NaN	NaN	NaN	outdoor	39.048939	-94.483984	1	0	
53.0	0	NaN	0.0	50.0	indoor	29.951049	-90.082308	1	1	
53.0	0	NaN	NaN	NaN	outdoor	44.500958	-88.061034	1	0	
55.0	0	NaN	NaN	NaN	outdoor	39.048939	-94.483984	1	0	
56.0	0	NaN	NaN	NaN	outdoor	27.977901	-82.505322	0	0	

In [152]:
```
new_nfl.describe()
```

Out[152]:

ead_favorite	over_under_line	stadium_neutral	weather_temperature	weather_wind_mph	weather_humidity	latitude	longitude	intra_conference	intra_d
361.000000	10361.000000	10361.000000	9712.000000	9722.000000	8166.000000	10361.000000	10361.000000	10361.000000	10361.0
-5.378294	42.109420	0.008590	59.860379	7.257972	62.686383	37.854069	-90.307461	0.726667	0.4
3.431416	4.770111	0.092287	15.417128	5.723770	15.693731	5.260936	15.843484	0.445692	0.4
-26.500000	28.000000	0.000000	-6.000000	0.000000	4.000000	19.303062	-122.389227	0.000000	0.0
-7.000000	38.500000	0.000000	50.000000	1.000000	50.000000	33.757577	-95.407756	0.000000	0.0
-4.500000	42.000000	0.000000	63.000000	7.000000	62.000000	39.098319	-86.164062	1.000000	0.0
-3.000000	45.000000	0.000000	72.000000	11.000000	75.000000	41.506056	-80.014015	1.000000	1.0
0.000000	63.500000	1.000000	97.000000	40.000000	100.000000	51.590914	-0.069968	1.000000	1.0

In [219]:
```
new_nfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10361 entries, 6 to 10395
Data columns (total 20 columns):
schedule_season        10361 non-null int64
schedule_week          10361 non-null int32
schedule_playoff       10361 non-null int64
team_home              10361 non-null object
team_away              10361 non-null object
spread_favorite        10361 non-null float64
over_under_line        10361 non-null float64
stadium_neutral        10361 non-null int64
weather_temperature    9712 non-null float64
weather_wind_mph       9722 non-null float64
weather_humidity       8166 non-null float64
stadium_type           10361 non-null object
latitude               10361 non-null float64
longitude              10361 non-null float64
intra_conference       10361 non-null int64
intra_division         10361 non-null int64
hit_over               10361 non-null int64
last_5_home            10361 non-null float64
last_5_away            10361 non-null float64
estimated_total        10361 non-null float64
dtypes: float64(10), int32(1), int64(6), object(3)
memory usage: 1.6+ MB
```

In [157]:
```
#Remove categorical data that already has information associated in data (stadium address, stadium name, etc.)
new_nfl.drop(columns=['team_home_conference', 'team_home_division', 'team_away_conference', 'team_away_division',
                     'stadium', 'schedule_date', 'team_favorite_id'], inplace=True)
```

In [158]:
```
#Going to drop NaN data in the last_5 categories, represents a tiny % of the data, should be fine moving forwar
new_nfl.dropna(subset=['last_5_home', 'last_5_away'], inplace=True)
```

In [114]:
```
hist_data = ['schedule_season', 'schedule_week', 'spread_favorite', 'over_under_line'
             , 'weather_temperature', 'weather_wind_mph', 'weather_humidity', 'stadium_type',
            'latitude', 'longitude', 'intra_conference', 'intra_division', 'hit_over', 'last_5_home', 'last_5_away']
```

Time for Modeling

In [115]:

```
def histogram_view(data_set):
    """This will produce 3 columns of histograms and a corresponding number
    of rows depending on the len(data_set.columns). The result will be a
    side by side view of all of the histograms of each column in a data set"""
    ncols = 3
    nrows = int(np.ceil(len(data_set.columns) / (1.0*ncols)))
    fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(20, 15))
    # Lazy counter so we can remove unwated axes
    counter = 0
    for i in range(nrows):
        for j in range(ncols):

            ax = axes[i][j]

            # Plot when we have data
            if counter < len(data_set.columns):

                ax.hist(data_set[data_set.columns[counter]], bins=20, color='blue', alpha=0.5, label='{}'.format(data
                ax.set_xlabel('{}'.format(data_set.columns[counter]))
                ax.set_ylabel('Density')
                leg = ax.legend(loc='best')
                leg.draw_frame(True)
                ax.grid(which='both', axis='both', linestyle='-')
            # Remove axis when we no longer have data
            else:
                ax.set_axis_off()

            counter += 1

    plt.show()
```

In [116]:

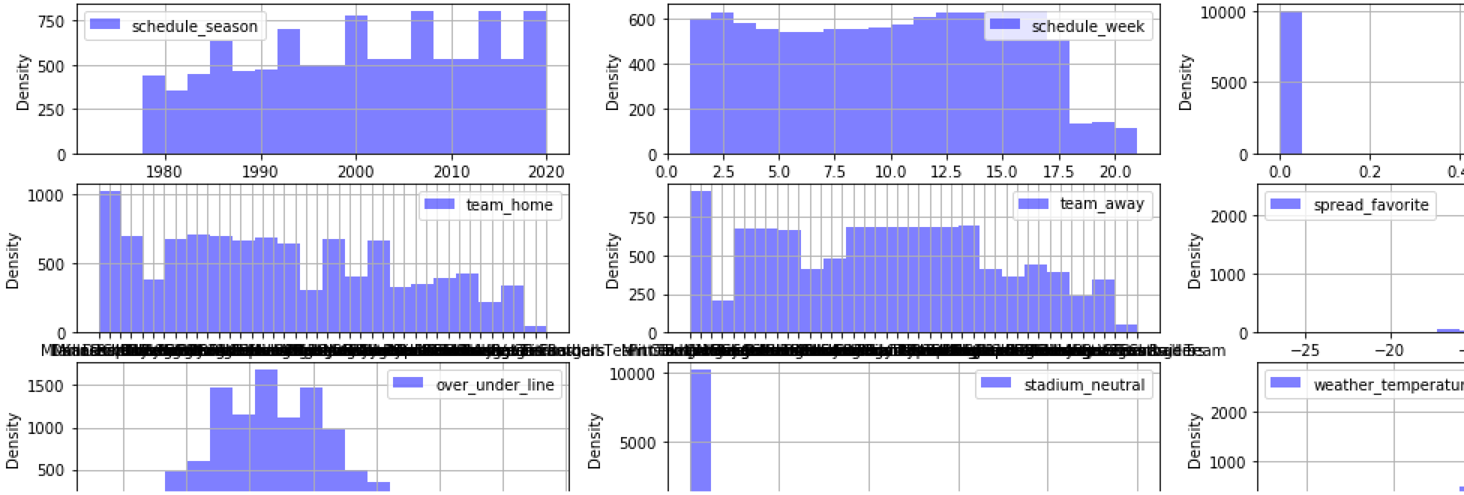
```
new_nfl.head()
```

Out[116]:

	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	team_away	team_favorite_id	spread_favorite	over_under_line	stadium
6	1974-01-13	1973	Superbowl	True	Miami Dolphins	Minnesota Vikings	MIA	-6.5	33.0	Rice Stadium
8	1976-01-18	1975	Superbowl	True	Dallas Cowboys	Pittsburgh Steelers	PIT	-7.0	36.0	Orange Bowl
9	1977-01-09	1976	Superbowl	True	Minnesota Vikings	Oakland Raiders	LVR	-4.0	38.0	Rose Bowl
11	1979-01-21	1978	Superbowl	True	Dallas Cowboys	Pittsburgh Steelers	PIT	-3.5	37.0	Orange Bowl
16	1979-09-02	1979	1	False	Kansas City Chiefs	Baltimore Colts	KC	-1.0	37.0	Arrowhead Stadium

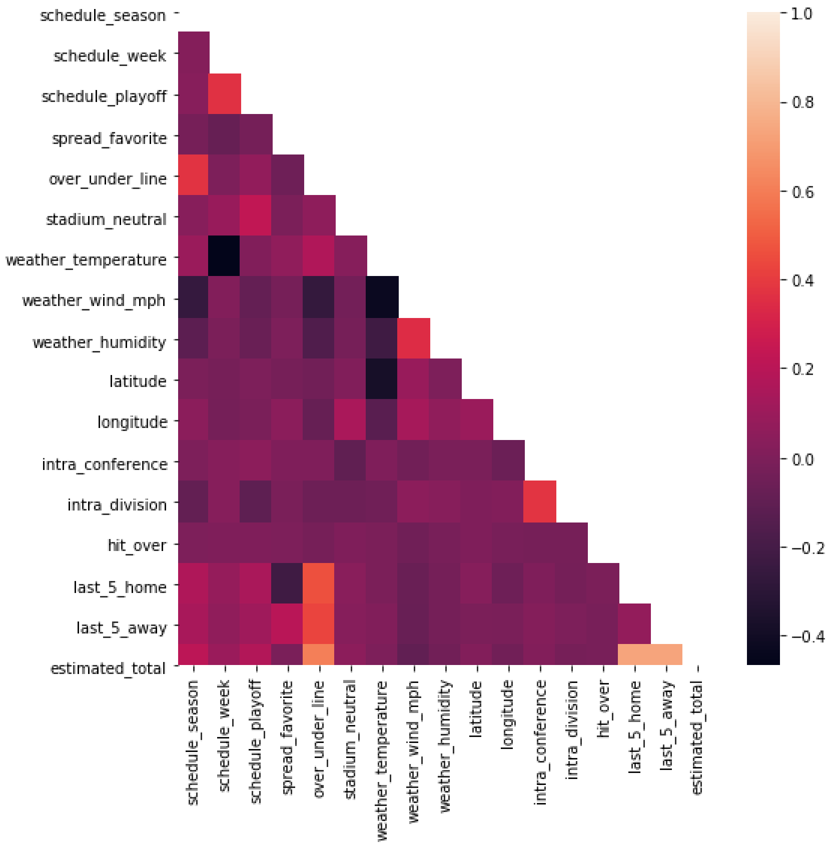
In [155]:

```
# Numpy doesn't like dtypes 'bool', so we'll just make them true = 1 and false = 0 for simplicty
new_nfl[['schedule_playoff', 'stadium_neutral']] = new_nfl[['schedule_playoff', 'stadium_neutral']].replace({True: 1,
In [195]: #Check for normalcy
          histogram_view(new_nfl)
```



- Overall the data looks very normal and should be a great starting point for regression

```
In [348]: # Now how about multicolinearity
          plt.figure(figsize=(8,8))
          matrix = np.triu(new_nfl.corr())
          sns.heatmap(new_nfl.corr(), mask=matrix)
          plt.show()
```



There are a lot of variables that appear to have a similar level of correlation in the middle of this correlation mat
by pulling up a Variance Inflation Factor

In [359]:

```
#Calculating VIF to review multicolinearity
VIF_cols = []
for c in new_nfl.columns:
    if new_nfl[c].dtype in ['float64', 'int64', 'int32']:
        VIF_cols.append(c)
vif = pd.DataFrame()
X = new_nfl[VIF_cols].dropna()
vif["variables"] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]

print(vif)
```

```
           variables         VIF
0    schedule_season  335.896910
1      schedule_week    6.841824
2   schedule_playoff    1.241090
3    spread_favorite    3.910372
4    over_under_line  146.000099
..               ...         ...
12    intra_division    2.086455
13          hit_over    1.955315
14       last_5_home         inf
15       last_5_away         inf
16   estimated_total         inf

[17 rows x 2 columns]

/Users/matthewnykaza/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/statsmodels/stats/outliers_influence.py
o encountered in double_scalars
  vif = 1. / (1. - r_squared_i)
```

- In general anything above 10 is considered to be high correlation, and just on a brief view it is clear that we have some extraordinaily high correlations go
- This gives some credence to the idea we will need to use a model (i.e. Tree Based) that does not care about multicolinearity

In [306]:

```
#Look into object cols
new_nfl[[c for c in new_nfl.columns if new_nfl[c].dtype == 'object']].describe()
```

Out[306]:

	team_home	team_away	stadium_type
count	10361	10361	10361
unique	42	42	3
top	New England Patriots	Green Bay Packers	outdoor
freq	362	350	7866

In [307]:

```
#going to to test train splits a little differently, as we want to test on the most recent data as this is likely
# what we will be doing in the future
#We are going to used the last 5 seasons of data 2016-2020 as based ont schedule season for our test, this is a tad
#small, but we can try and change this later if we find it is too small
# and we'll define X and y in this step
train = new_nfl[(new_nfl['schedule_season'] < 2016)]
test = new_nfl[(new_nfl['schedule_season'] >= 2016)]
y_train = train['hit_over']
y_test = test['hit_over']
X_train = train.drop(columns=['hit_over'])
X_test = test.drop(columns=['hit_over'])
```

In [308]:

```
#Check to make sure indicies look right for train and test
y_train.head()
```

Out[308]:

```
6     0
8     1
9     1
11    1
16    0
Name: hit_over, dtype: int64
```

In [309]:

```
X_train.head()
```

Out[309]:

	schedule_season	schedule_week	schedule_playoff	team_home	team_away	spread_favorite	over_under_line	stadium_neutral	weather_temperature	weat
6	1973	21	1	Miami Dolphins	Minnesota Vikings	-6.5	33.0	1	47.0	
8	1975	21	1	Dallas Cowboys	Pittsburgh Steelers	-7.0	36.0	1	49.0	
9	1976	21	1	Minnesota Vikings	Oakland Raiders	-4.0	38.0	1	52.0	
11	1978	21	1	Dallas Cowboys	Pittsburgh Steelers	-3.5	37.0	1	71.0	
16	1979	1	0	Kansas City Chiefs	Baltimore Colts	-1.0	37.0	0	76.0	

In [310]:

```
y_test.head()
```

Out[310]:

```
9059    1
9060    0
9061    1
9062    0
9063    0
Name: hit_over, dtype: int64
```

```
In [311]: X_test.head()
```

Out[311]:

	schedule_season	schedule_week	schedule_playoff	team_home	team_away	spread_favorite	over_under_line	stadium_neutral	weather_temperature	we
9059	2016	1	0	Denver Broncos	Carolina Panthers	-3.0	40.5	0	82.0	
9060	2016	1	0	Arizona Cardinals	New England Patriots	-8.5	44.0	0	72.0	
9061	2016	1	0	Atlanta Falcons	Tampa Bay Buccaneers	-2.5	47.0	0	72.0	
9062	2016	1	0	Baltimore Ravens	Buffalo Bills	-3.0	44.5	0	82.0	
9063	2016	1	0	Dallas Cowboys	New York Giants	-1.0	47.5	0	72.0	

All looks solid for this point, time to get to modeling.

Modeling

```
In [312]: # Look at some basic information about how we might want to set up our pipeline for the following steps

num_cols = []
ohe_cols = []
freq_cols = []

for c in X_train.columns:
    if new_nfl[c].dtype in ['float64', 'int64', 'int32']:
        num_cols.append(c)
    elif len(X_train[c].unique()) <= 15:
        ohe_cols.append(c)
    else:
        freq_cols.append(c)
```

```
In [313]: # Check our work
print(f"Numeric: {num_cols}")
print(f"To OHE: {ohe_cols}")
print(f"To Frequency Encode: {freq_cols}")
```

```
Numeric: ['schedule_season', 'schedule_week', 'schedule_playoff', 'spread_favorite', 'over_under_line', 'stadium_neut
er_wind_mph', 'weather_humidity', 'latitude', 'longitude', 'intra_conference', 'intra_division', 'last_5_home', 'last
To OHE: ['stadium_type']
To Frequency Encode: ['team_home', 'team_away']
```

- I even might just want to OHE the home and away team names, but I also might just get rid of these all together as I have some concerns about how tea relying on how a team performed in 1979, just to learn trends in the data
- I think it makes sense to OHE the stadium type, but if this gives me issues later I can also make them 0=outdoors, 1=indoors and 3=retractable, but I am other two

In [314]:

```
#Check vs. test to make sure there are not any issues
num_cols_test = []
ohe_cols_test = []
freq_cols_test = []

for c in X_test.columns:
    if new_nfl[c].dtype in ['float64', 'int64', 'int32']:
        num_cols_test.append(c)
    elif len(X_test[c].unique()) <= 15:
        ohe_cols_test.append(c)
    else:
        freq_cols_test.append(c)
```

In [315]:

```
# Check our work
print(f"Numeric: {num_cols_test}")
print(f"To OHE: {ohe_cols_test}")
print(f"To Frequency Encode: {freq_cols_test}")
```

```
Numeric: ['schedule_season', 'schedule_week', 'schedule_playoff', 'spread_favorite', 'over_under_line', 'stadium_neut
er_wind_mph', 'weather_humidity', 'latitude', 'longitude', 'intra_conference', 'intra_division', 'last_5_home', 'last
To OHE: ['stadium_type']
To Frequency Encode: ['team_home', 'team_away']
```

Setting up preprocessing

- Numerical Data
-   - Since (as seen in the histograms) there do not seem to be many major outliers present in the data I am simply going to use a Min-Max scaler to get a
-   - For the imputer I am going to start with the median, there really isn't that much a difference than the mean here so I doubt that it will have much of a
- Categorical Data
-   - Nothing too crazy here, only have the one column of stadium_type, so this should be straighforward encoding
-   - May later on try the home and away teams here, but the more I think about it, the more likely it is that I will drop them.
- Frequency Data
-   - As stated before I very well may drop the teams, just going to encode for now with the Count Encoder

In [316]:

```
X_train.describe()
```

Out[316]:

	schedule_season	schedule_week	schedule_playoff	spread_favorite	over_under_line	stadium_neutral	weather_temperature	weather_wind_mph	weathe
count	9027.000000	9027.000000	9027.000000	9027.000000	9027.000000	9027.000000	8782.000000	8782.000000	7
mean	1997.961560	9.433920	0.036889	-5.389110	41.537244	0.007422	59.496242	7.578684	
std	10.541066	5.179595	0.188501	3.426065	4.541411	0.085837	15.322526	5.731902	
min	1973.000000	1.000000	0.000000	-26.500000	28.000000	0.000000	-6.000000	0.000000	
25%	1989.000000	5.000000	0.000000	-7.000000	38.000000	0.000000	49.000000	2.000000	
50%	1998.000000	10.000000	0.000000	-4.500000	41.000000	0.000000	63.000000	8.000000	
75%	2007.000000	14.000000	0.000000	-3.000000	44.500000	0.000000	72.000000	11.000000	
max	2015.000000	21.000000	1.000000	0.000000	63.000000	1.000000	95.000000	40.000000	

In [317]:

```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9027 entries, 6 to 9058
Data columns (total 19 columns):
schedule_season        9027 non-null int64
schedule_week          9027 non-null int32
schedule_playoff       9027 non-null int64
team_home              9027 non-null object
team_away              9027 non-null object
spread_favorite        9027 non-null float64
over_under_line        9027 non-null float64
stadium_neutral        9027 non-null int64
weather_temperature    8782 non-null float64
weather_wind_mph       8782 non-null float64
weather_humidity       7987 non-null float64
stadium_type           9027 non-null object
latitude               9027 non-null float64
longitude              9027 non-null float64
intra_conference       9027 non-null int64
intra_division         9027 non-null int64
last_5_home            9027 non-null float64
last_5_away            9027 non-null float64
estimated_total        9027 non-null float64
dtypes: float64(10), int32(1), int64(5), object(3)
memory usage: 1.3+ MB
```

In [318]:

```
# Now, set up the preprocessing steps for each type of col
num_transformer = Pipeline(steps=[
    ('num_imputer', SimpleImputer(strategy='median')),
    ('scaler', MinMaxScaler())])

ohe_transformer = Pipeline(steps=[
    ('cat_imputer', SimpleImputer(strategy='constant', fill_value='Unknown')),
    ('ohe', OneHotEncoder(handle_unknown='ignore'))])

freq_transformer = Pipeline(steps=[
    ('freq_imputer', SimpleImputer(strategy='constant', fill_value='Unknown')),
    ('freq_enc', ce.CountEncoder(normalize=True,
                                 handle_unknown=0,
                                 min_group_size=0.001,
                                 min_group_name='Other'))])
```

In [319]:

```
# Put together our preprocessor using a Column Transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_transformer, num_cols),
        ('cat_ohe', ohe_transformer, ohe_cols),
        ('cat_freq', freq_transformer, freq_cols)])
```

In [320]:
```
#Check work
X_train.info()
```

```
schedule_week          9027 non-null int32
schedule_playoff       9027 non-null int64
team_home              9027 non-null object
team_away              9027 non-null object
spread_favorite        9027 non-null float64
over_under_line        9027 non-null float64
stadium_neutral        9027 non-null int64
weather_temperature    8782 non-null float64
weather_wind_mph       8782 non-null float64
weather_humidity       7987 non-null float64
stadium_type           9027 non-null object
latitude               9027 non-null float64
longitude              9027 non-null float64
intra_conference       9027 non-null int64
intra_division         9027 non-null int64
last_5_home            9027 non-null float64
last_5_away            9027 non-null float64
estimated_total        9027 non-null float64
dtypes: float64(10), int32(1), int64(5), object(3)
memory usage: 1.3+ MB
```

In [538]:
```
# Append classifier to preprocessing pipeline.
# Now we have a full prediction pipeline.
clf_logreg = Pipeline(steps=[('preprocessor', preprocessor),
                             ('classifier', LogisticRegression(class_weight='balanced', random_state=42))])

clf_logreg.fit(X_train, y_train)
```

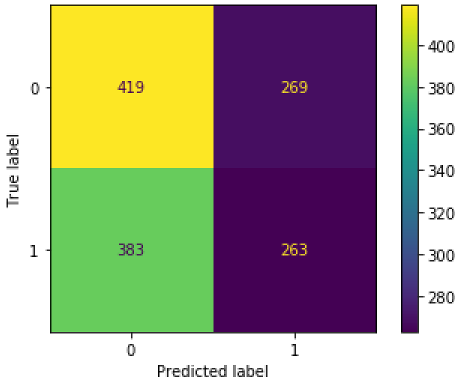
Out[538]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('num_imputer',
                                                                   SimpleImputer(strategy='median')),
                                                                  ('scaler',
                                                                   MinMaxScaler())]),
                                                  ['schedule_season',
                                                   'schedule_week',
                                                   'schedule_playoff',
                                                   'spread_favorite',
                                                   'over_under_line',
                                                   'stadium_neutral',
                                                   'weather_temperature',
                                                   'weather_wind_mph',
                                                   'weather_humidity',
                                                   'latitude', 'longitude',
                                                   'intra_conference',
                                                   'intra_division',
                                                   'last_5_home', 'last_5_away',
                                                   'estimated_total']),
                                                 ('cat_ohe',
                                                  Pipeline(steps=[('cat_imputer',
                                                                   SimpleImputer(fill_value='Unknown',
                                                                                 strategy='constant')),
                                                                  ('ohe',
                                                                   OneHotEncoder(handle_unknown='ignore'))]),
                                                  ['stadium_type'])])),
                ('classifier',
                 LogisticRegression(class_weight='balanced', random_state=42))])
```

In [539]:

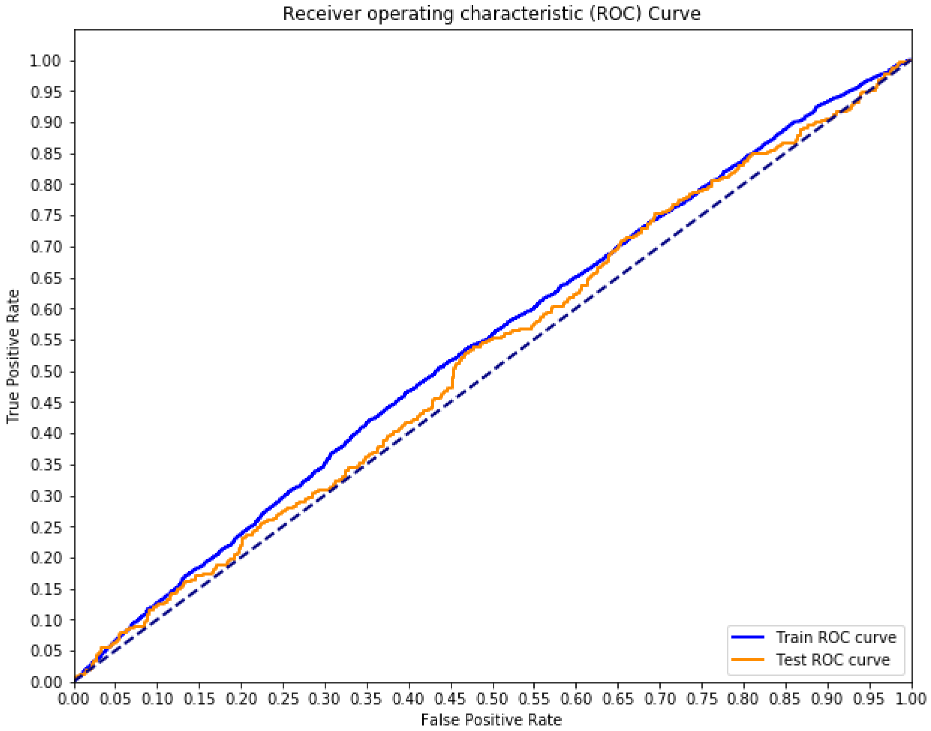
```python
# This is just a nice little bit of code that will give us relevant test results of our model!
def evaluate(estimator, X_train, X_test, y_train, y_test, use_decision_function='yes'):
    '''
    Evaluation function to show a few scores for both the train and test set
    Also shows a confusion matrix for the test set

    use_decision_function allows you to toggle whether you use decision_function or
    predict_proba in order to get the output needed for roc_auc_score
    If use_decision_function == 'skip', then it ignores calculating the roc_auc_score

    Additionally for models that have a decision function this model will show a ROC Curve
    '''
    # grab predictions
    train_preds = estimator.predict(X_train)
    test_preds = estimator.predict(X_test)

    # output needed for roc_auc_score
    if use_decision_function == 'skip': # skips calculating the roc_auc_score
        train_out = False
        test_out = False
    elif use_decision_function == 'yes': # not all classifiers have decision_function
        train_out = estimator.decision_function(X_train)
        test_out = estimator.decision_function(X_test)
    elif use_decision_function == 'no':
        train_out = estimator.predict_proba(X_train)[:, 1] # proba for the 1 class
        test_out = estimator.predict_proba(X_test)[:, 1]
    else:
        raise Exception ("The value for use_decision_function should be 'skip', 'yes' or 'no'.")

    print(type(test_out))

    # print scores
    print("Train Scores")
    print("------------")
    print(f"Accuracy: {accuracy_score(y_train, train_preds)}")
    print(f"F1 Score: {f1_score(y_train, train_preds)}")
    if type(train_out) == np.ndarray:
        print(f"ROC-AUC: {roc_auc_score(y_train, train_out)}")
    print("----" * 5)
    print("Test Scores")
    print("-----------")
    print(f"Accuracy: {accuracy_score(y_test, test_preds)}")
    print(f"F1 Score: {f1_score(y_test, test_preds)}")
    if type(test_out) == np.ndarray:
        print(f"ROC-AUC: {roc_auc_score(y_test, test_out)}")

    # plot test confusion matrix
    plot_confusion_matrix(estimator, X_test, y_test)
    plt.show()

    #Plot ROC Curve
    if use_decision_function == 'yes':
        y_train_score = estimator.decision_function(X_train)
        y_test_score = estimator.decision_function(X_test)

        train_fpr, train_tpr, train_thresholds = roc_curve(y_train, y_train_score)
        test_fpr, test_tpr, test_thresholds = roc_curve(y_test, y_test_score)

        print('Train AUC: {}'.format(auc(train_fpr, train_tpr)))
        print('Test AUC: {}'.format(auc(test_fpr, test_tpr)))
        plt.figure(figsize=(10, 8))
        lw = 2

        plt.plot(train_fpr, train_tpr, color='blue',
                 lw=lw, label='Train ROC curve')
        plt.plot(test_fpr, test_tpr, color='darkorange',
                 lw=lw, label='Test ROC curve')

        plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05])
        plt.yticks([i/20.0 for i in range(21)])
        plt.xticks([i/20.0 for i in range(21)])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('Receiver operating characteristic (ROC) Curve')
        plt.legend(loc='lower right')
        plt.show()
    else:
        None
```

```
In [541]: evaluate(clf_logreg, X_train, X_test, y_train, y_test)
```

```
<class 'numpy.ndarray'>
Train Scores
------------
Accuracy: 0.5324027916251246
F1 Score: 0.5266345183357631
ROC-AUC: 0.5416435185207935
--------------------
Test Scores
-----------
Accuracy: 0.5112443778110944
F1 Score: 0.44651952461799665
ROC-AUC: 0.523093815249478
```



```
Train AUC: 0.5416435185207935
Test AUC: 0.523093815249478
```



Some Takeaways

- Slightly better than a guess
- The confusion matrix shows that we are best at telling when a game is going to be 'under'
- ROC curve indicates that we are essentially guessing with this model

Let's take teams out and see what happens???

```
In [542]: #Remove team names from the data
          no_team_names = new_nfl.copy().drop(columns=['team_home', 'team_away'])
          train = no_team_names[(no_team_names['schedule_season'] < 2016)]
          test = no_team_names[(no_team_names['schedule_season'] >= 2016)]
          y_train = train['hit_over']
          y_test = test['hit_over']
          X_train = train.drop(columns=['hit_over'])
          X_test = test.drop(columns=['hit_over'])
```

In [543]:

```
#Need to run these again
num_cols = []
ohe_cols = []
freq_cols = []

for c in X_train.columns:
    if no_team_names[c].dtype in ['float64', 'int64', 'int32']:
        num_cols.append(c)
    elif len(X_train[c].unique()) <= 15:
        ohe_cols.append(c)
    else:
        freq_cols.append(c)
```

In [544]:

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_transformer, num_cols),
        ('cat_ohe', ohe_transformer, ohe_cols)])
```

In [545]:

```

print(f"Numeric: {num_cols}")
print(f"To OHE: {ohe_cols}")
print(f"To Frequency Encode: {freq_cols}")
```

```
Numeric: ['schedule_season', 'schedule_week', 'schedule_playoff', 'spread_favorite', 'over_under_line', 'stadium_neut
er_wind_mph', 'weather_humidity', 'latitude', 'longitude', 'intra_conference', 'intra_division', 'last_5_home', 'last
To OHE: ['stadium_type']
To Frequency Encode: []
```

In [546]:

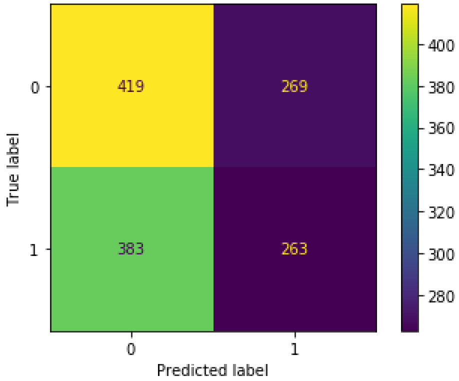
```
clf_logreg = Pipeline(steps=[('preprocessor', preprocessor),
                             ('classifier', LogisticRegression(class_weight='balanced', random_state=42))])
clf_logreg.fit(X_train, y_train)
```

Out[546]:

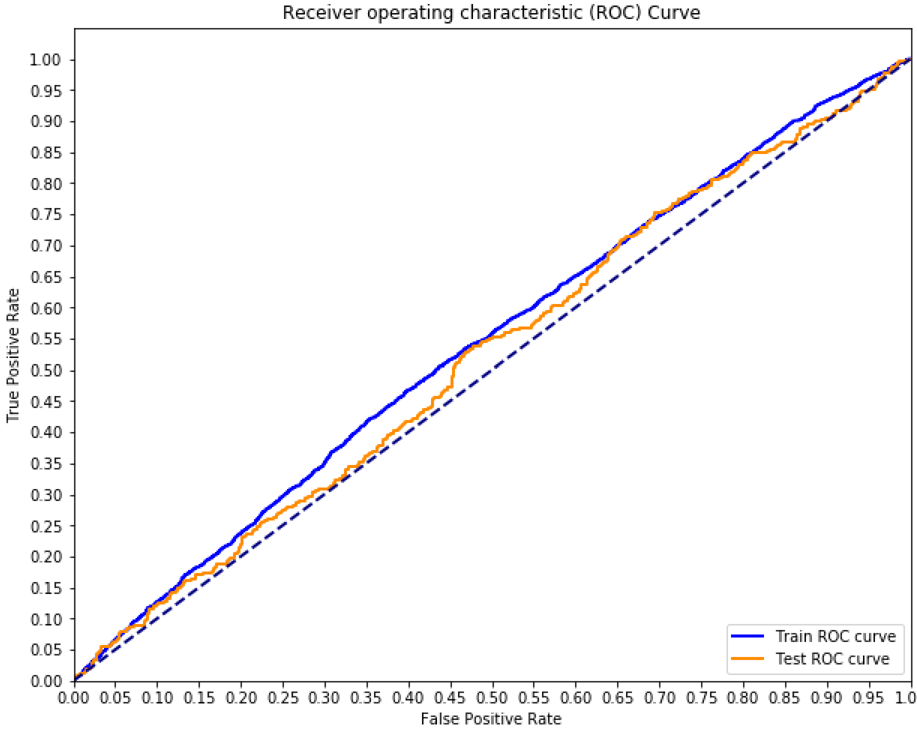
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('num_imputer',
                                                                   SimpleImputer(strategy='median')),
                                                                  ('scaler',
                                                                   MinMaxScaler())]),
                                                  ['schedule_season',
                                                   'schedule_week',
                                                   'schedule_playoff',
                                                   'spread_favorite',
                                                   'over_under_line',
                                                   'stadium_neutral',
                                                   'weather_temperature',
                                                   'weather_wind_mph',
                                                   'weather_humidity',
                                                   'latitude', 'longitude',
                                                   'intra_conference',
                                                   'intra_division',
                                                   'last_5_home', 'last_5_away',
                                                   'estimated_total']),
                                                 ('cat_ohe',
                                                  Pipeline(steps=[('cat_imputer',
                                                                   SimpleImputer(fill_value='Unknown',
                                                                                 strategy='constant')),
                                                                  ('ohe',
                                                                   OneHotEncoder(handle_unknown='ignore'))]),
                                                  ['stadium_type'])])),
                ('classifier',
                 LogisticRegression(class_weight='balanced', random_state=42))])
```

```
In [547]: evaluate(clf_logreg, X_train, X_test, y_train, y_test)
```

```
<class 'numpy.ndarray'>
Train Scores
------------
Accuracy: 0.5324027916251246
F1 Score: 0.5266345183357631
ROC-AUC: 0.5416435185207935
--------------------
Test Scores
-----------
Accuracy: 0.5112443778110944
F1 Score: 0.44651952461799665
ROC-AUC: 0.523093815249478
```



```
Train AUC: 0.5416435185207935
Test AUC: 0.523093815249478
```



Some Takeaways

- Overall it does not appear to have made much of a difference
- We may be ever so slightly overfit, but not any more so than what we were before
- It is clear that a basic Logistic Regression is not the answer
-
    - This is likely because there is a whole lot of multicolinearity in the data, so I believe it will be best to use a Tree based model as this will negate any n

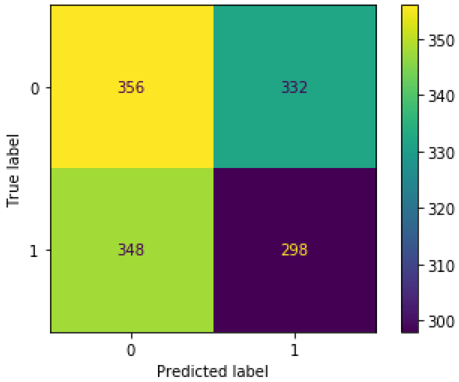
Tree Based Models

In [596]:
```
tree_logreg = Pipeline(steps=[('preprocessor', preprocessor),
                              ('classifier', DecisionTreeClassifier(class_weight='balanced', random_state=42))])
tree_logreg.fit(X_train, y_train)
```

Out[596]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('num_imputer',
                                                                   SimpleImputer(strategy='median')),
                                                                  ('scaler',
                                                                   MinMaxScaler())]),
                                                  ['schedule_season',
                                                   'schedule_week',
                                                   'schedule_playoff',
                                                   'spread_favorite',
                                                   'over_under_line',
                                                   'stadium_neutral',
                                                   'weather_temperature',
                                                   'weather_wind_mph',
                                                   'weather_humidity',
                                                   'latitude', 'longitude',
                                                   'intra_conference',
                                                   'intra_division',
                                                   'last_5_home', 'last_5_away',
                                                   'estimated_total']),
                                                 ('cat_ohe',
                                                  Pipeline(steps=[('cat_imputer',
                                                                   SimpleImputer(fill_value='Unknown',
                                                                                 strategy='constant')),
                                                                  ('ohe',
                                                                   OneHotEncoder(handle_unknown='ignore'))]),
                                                  ['stadium_type'])])),
                ('classifier',
                 DecisionTreeClassifier(class_weight='balanced',
                                        random_state=42))])
```

In [597]:
```
evaluate(tree_logreg, X_train, X_test, y_train, y_test, use_decision_function='no')
```

```
<class 'numpy.ndarray'>
Train Scores
------------
Accuracy: 1.0
F1 Score: 1.0
ROC-AUC: 1.0
--------------------
Test Scores
-----------
Accuracy: 0.49025487256371814
F1 Score: 0.4670846394984326
ROC-AUC: 0.48937108503131976
```



Some Takeaways

- Scores are largely unchanged in the grand scheme of things here, but with some hyperparameter tuning I believe that we can get a solid score
- Current goal is to get in the 60 range for all scores

In [615]:
```
#Let's use GridsearchCV
param_grid = {
    'classifier__max_depth': [x for x in range(4,12,2)],
    'classifier__min_samples_split': [x for x in range(4, 12, 2)],
    'classifier__max_features': [ 'log2'],
    'classifier__min_samples_leaf': [x for x in range(10, 20, 2)],

}
```

In [616]:
```
griddy = GridSearchCV(tree_logreg, param_grid, n_jobs=1, cv=3)
```

In [617]:

```
griddy.fit(X_train, y_train)
```

Out[617]:

```
GridSearchCV(cv=3,
             estimator=Pipeline(steps=[('preprocessor',
                                        ColumnTransformer(transformers=[('num',
                                                                         Pipeline(steps=[('num_imputer',
                                                                                          SimpleImputer(strategy='med
                                                                                         ('scaler',
                                                                                          MinMaxScaler())]),
                                                                         ['schedule_season',
                                                                          'schedule_week',
                                                                          'schedule_playoff',
                                                                          'spread_favorite',
                                                                          'over_under_line',
                                                                          'stadium_neutral',
                                                                          'weather_temperature',
                                                                          'weather_wind_mph',
                                                                          'weather_...
                                                                                                         strategy='con
                                                                                         ('ohe',
                                                                                          OneHotEncoder(handle_unknow
                                                                         ['stadium_type'])])),
                                       ('classifier',
                                        DecisionTreeClassifier(class_weight='balanced',
                                                               random_state=42))]),
             n_jobs=1,
             param_grid={'classifier__max_depth': [4, 6, 8, 10],
                         'classifier__max_features': ['log2'],
                         'classifier__min_samples_leaf': [10, 12, 14, 16, 18],
                         'classifier__min_samples_split': [4, 6, 8, 10]})
```

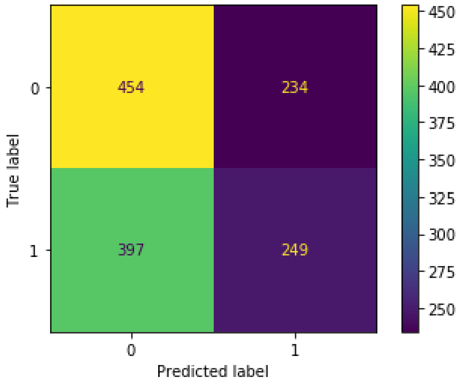
In [618]:

```
print(f"Best parameter's score: {griddy.best_score_:0.3f}):")
print(griddy.best_params_)
```

```
Best parameter's score: 0.521):
{'classifier__max_depth': 8, 'classifier__max_features': 'log2', 'classifier__min_samples_leaf': 14, 'classifier__min
```

```
In [619]: evaluate(griddy.best_estimator_, X_train, X_test, y_train, y_test, use_decision_function='no')
```

```
<class 'numpy.ndarray'>
Train Scores
------------
Accuracy: 0.5624238395923341
F1 Score: 0.47529224229543043
ROC-AUC: 0.5956151261262331
--------------------
Test Scores
-----------
Accuracy: 0.5269865067466267
F1 Score: 0.44109831709477415
ROC-AUC: 0.5279222766217871
```



On the Hyperparameters

- criterion - from what I could find, either way the model performed about the same, so I am not concerned with this
- splitter - similar takeaway as criterion
- max_depth - with nothing set the tree will be perfect on the training set, but not generalize well on test set. Through some testing I found that a depth of this seemed a bit small to me
- min_samples_split - looked like 4 was very consistently the best number here, higher numbers tend to introduce under-fitting, so I think 4 was a good nu
- min_samples_leaf- is similar to min_samples_split, but when used togther they can help us make sure that the tree isn't using 1 sample to make a decisi
- max_features - mostly helps with computational time, so I really went with allowing the model to determine what is best, as I am not too concerned with
- max_leaf_nodes - ran this a few times and it appeared that haveing a number around 80 was bst for this metric, although none of this improved the mod seemed best when I left it alone.

Some Takeaways

- The results largely improved, with the exception of F1 score on the test set, this is not a great development as the lower f1 score (combined with greater
- Seems like this was better at selecting overs (1), but may have just erred that way too often
- It seems that through pruning I am mostly getting trees that err one way or the other, but not really improving greatly on their overall scores

One last model, Random Forrests

This will be an ensemble model that ideally will be the most powerful

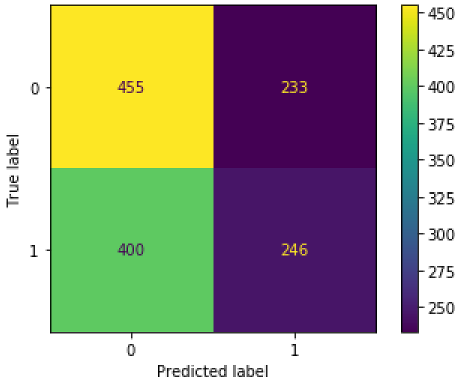
```
In [578]: forest = Pipeline(steps=[('preprocessor', preprocessor),
                                       ('classifier', RandomForestClassifier(class_weight='balanced', random_state=42))])
          forest.fit(X_train, y_train)
```

```
Out[578]: Pipeline(steps=[('preprocessor',
                           ColumnTransformer(transformers=[('num',
                                                            Pipeline(steps=[('num_imputer',
                                                                             SimpleImputer(strategy='median')),
                                                                            ('scaler',
                                                                             MinMaxScaler())]),
                                                            ['schedule_season',
                                                             'schedule_week',
                                                             'schedule_playoff',
                                                             'spread_favorite',
                                                             'over_under_line',
                                                             'stadium_neutral',
                                                             'weather_temperature',
                                                             'weather_wind_mph',
                                                             'weather_humidity',
                                                             'latitude', 'longitude',
                                                             'intra_conference',
                                                             'intra_division',
                                                             'last_5_home', 'last_5_away',
                                                             'estimated_total']),
                                                           ('cat_ohe',
                                                            Pipeline(steps=[('cat_imputer',
                                                                             SimpleImputer(fill_value='Unknown',
                                                                                           strategy='constant')),
                                                                            ('ohe',
                                                                             OneHotEncoder(handle_unknown='ignore'))]),
                                                            ['stadium_type'])])),
                          ('classifier',
                           RandomForestClassifier(class_weight='balanced',
                                                  random_state=42))])
```

In [579]:

```
evaluate(forest, X_train, X_test, y_train, y_test, use_decision_function='no')
```

```
<class 'numpy.ndarray'>
Train Scores
------------
Accuracy: 1.0
F1 Score: 1.0
ROC-AUC: 1.0
--------------------
Test Scores
-----------
Accuracy: 0.525487256371814
F1 Score: 0.43733333333333333
ROC-AUC: 0.5209315825473395
```



Some Takeaways

- The model is picking the under (0) far to often, seemingly the opposite of our last model
- On a whole the scores did not improve much, over the DecisionTree with hyperparameter selection

In [599]:

```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
```

```
In [606]: random_grid = {'classifier__n_estimators': n_estimators,
                         'classifier__max_features': max_features,
                         'classifier__max_depth': max_depth,
                         'classifier__min_samples_split': min_samples_split,
                         'classifier__min_samples_leaf': min_samples_leaf,
                         'classifier__bootstrap': bootstrap}
          print(random_grid)
```

```
{'classifier__n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000], 'classifier__max_features': ['
pth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None], 'classifier__min_samples_split': [2, 5, 10], 'classifier_
ssifier__bootstrap': [True, False]}
```

```
In [607]: forest = Pipeline(steps=[('preprocessor', preprocessor),
                                       ('classifier', RandomForestClassifier(class_weight='balanced', random_state=42))])
          forest.fit(X_train, y_train)
```

```
Out[607]: Pipeline(steps=[('preprocessor',
                           ColumnTransformer(transformers=[('num',
                                                            Pipeline(steps=[('num_imputer',
                                                                             SimpleImputer(strategy='median')),
                                                                            ('scaler',
                                                                             MinMaxScaler())]),
                                                            ['schedule_season',
                                                             'schedule_week',
                                                             'schedule_playoff',
                                                             'spread_favorite',
                                                             'over_under_line',
                                                             'stadium_neutral',
                                                             'weather_temperature',
                                                             'weather_wind_mph',
                                                             'weather_humidity',
                                                             'latitude', 'longitude',
                                                             'intra_conference',
                                                             'intra_division',
                                                             'last_5_home', 'last_5_away',
                                                             'estimated_total']),
                                                           ('cat_ohe',
                                                            Pipeline(steps=[('cat_imputer',
                                                                             SimpleImputer(fill_value='Unknown',
                                                                                           strategy='constant')),
                                                                            ('ohe',
                                                                             OneHotEncoder(handle_unknown='ignore'))]),
                                                            ['stadium_type'])])),
                          ('classifier',
                           RandomForestClassifier(class_weight='balanced',
                                                  random_state=42))])
```

In [609]:
```
# Use the random grid to search for best hyperparameters
# First create the base model to tune


# Random search of parameters, using 3 fold cross validation,
# search across 20 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = forest, param_distributions = random_grid, n_iter = 20, cv = 3, verbose=2,
# Fit the random search model
rf_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
```

Out[609]:
```
RandomizedSearchCV(cv=3,
                   estimator=Pipeline(steps=[('preprocessor',
                                              ColumnTransformer(transformers=[('num',
                                                                               Pipeline(steps=[('num_imputer',
                                                                                                SimpleImputer(strateg
                                                                                               ('scaler',
                                                                                                MinMaxScaler())]),
                                                                               ['schedule_season',
                                                                                'schedule_week',
                                                                                'schedule_playoff',
                                                                                'spread_favorite',
                                                                                'over_under_line',
                                                                                'stadium_neutral',
                                                                                'weather_temperature',
                                                                                'weather_wind_mph',
                                                                                'we...
                   n_iter=20, n_jobs=-1,
                   param_distributions={'classifier__bootstrap': [True, False],
                                        'classifier__max_depth': [10, 20, 30,
                                                                  40, 50, 60,
                                                                  70, 80, 90,
                                                                  100, 110,
                                                                  None],
                                        'classifier__max_features': ['auto',
                                                                     'sqrt'],
                                        'classifier__min_samples_leaf': [1, 2,
                                                                         4],
                                        'classifier__min_samples_split': [2, 5,
                                                                          10],
                                        'classifier__n_estimators': [200, 400,
                                                                     600, 800,
                                                                     1000, 1200,
                                                                     1400, 1600,
                                                                     1800,
                                                                     2000]},
                   verbose=2)
```

In [622]:
```
#This will give us a good baseline as to where to start our GridsearchCV process
rf_random.best_params_
```

Out[622]:
```
{'classifier__n_estimators': 200,
 'classifier__min_samples_split': 5,
 'classifier__min_samples_leaf': 4,
 'classifier__max_features': 'auto',
 'classifier__max_depth': 110,
 'classifier__bootstrap': True}
```

In [626]:
```
#Utilized those paramaters to narrow down on GridSearchCV
param_grid = {'classifier__n_estimators': [150, 200, 250],
               'classifier__max_features': ['auto'],
               'classifier__max_depth': [110, 120, 140],
               'classifier__min_samples_split': [4, 5, 6],
               'classifier__min_samples_leaf': [3, 4, 5],
               'classifier__bootstrap': [True]}
```

In [627]:
```
griddy_2 = GridSearchCV(estimator=forest, param_grid=param_grid, n_jobs=1, cv=3)
```

In [628]:
```
griddy_2.fit(X_train, y_train)
```

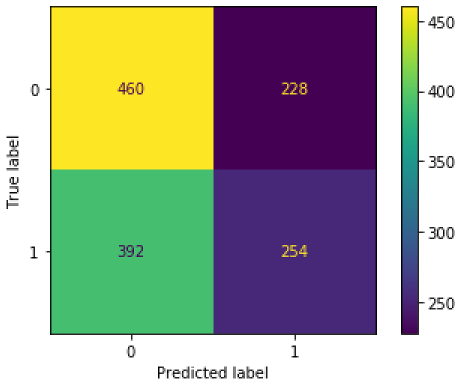
Out[628]:
```
GridSearchCV(cv=3,
             estimator=Pipeline(steps=[('preprocessor',
                                        ColumnTransformer(transformers=[('num',
                                                                         Pipeline(steps=[('num_imputer',
                                                                                          SimpleImputer(strategy='med
                                                                                         ('scaler',
                                                                                          MinMaxScaler())]),
                                                                         ['schedule_season',
                                                                          'schedule_week',
                                                                          'schedule_playoff',
                                                                          'spread_favorite',
                                                                          'over_under_line',
                                                                          'stadium_neutral',
                                                                          'weather_temperature',
                                                                          'weather_wind_mph',
                                                                          'weather_...
                                                                                          OneHotEncoder(handle_unknow
                                                                         ['stadium_type'])])),
                                       ('classifier',
                                        RandomForestClassifier(class_weight='balanced',
                                                               random_state=42))]),
             n_jobs=1,
             param_grid={'classifier__bootstrap': [True],
                         'classifier__max_depth': [110, 120, 140],
                         'classifier__max_features': ['auto'],
                         'classifier__min_samples_leaf': [3, 4, 5],
                         'classifier__min_samples_split': [4, 5, 6],
                         'classifier__n_estimators': [150, 200, 250]})
```

In [629]:
```
print(f"Best parameter's score: {griddy_2.best_score_:0.3f}):")
print(griddy_2.best_params_)
```

```
Best parameter's score: 0.521):
{'classifier__bootstrap': True, 'classifier__max_depth': 110, 'classifier__max_features': 'auto', 'classifier__min_sa
mples_split': 4, 'classifier__n_estimators': 150}
```

```
In [630]: evaluate(griddy_2.best_estimator_, X_train, X_test, y_train, y_test, use_decision_function='no')
```

```
<class 'numpy.ndarray'>
Train Scores
------------
Accuracy: 0.9996676636756398
F1 Score: 0.9996567112941984
ROC-AUC: 0.9999885999578788
--------------------
Test Scores
-----------
Accuracy: 0.5352323838080959
F1 Score: 0.45035460992907805
ROC-AUC: 0.5317247462020304
```



Conclusion

- This model was able to perform the best out of all previous models, and I really think that finding RancomizerCV as a method of getting a good starting p
- All metrics rose by around .01 - .015 points, which may not seem like a lot, but this was the greatest increase in the data seen to date.
- Overall what this shows is the need to complete more data cleaning, and get more data.
- I believe that one major issue is that I am trying to beat Las Vegas, which creates these point spreads using models much more advanced and practiced could be a viable product
- It may be that I have too much past data, when they game was very different, this could be a hinderence as well

Further Work

- Need to get more information about each individual game, this could include more data mining and more feature engineering of feature variables
- More tests of Hyperparameters

    - Utilize graphing techniques to help determine some of these features
- Try boost models
    - They tend to be more powerful, and may be able to achieve better scores

```
In [ ]:
```