2018

Analysis of Crime Neighboring Sporting Events

SPRINGBOARD DATA SCIENCE CAPSTONE PROJECT

FRANCISCO SALAS

Table of Contents

Introduction

Houston Texas with its 2.3 million residents is the fourth most populous city in the United States, just behind New York, Los Angeles, and Chicago. As with any large city, Houston has a rich sporting culture with four professional major league teams and two NCAA Division I-A athletic programs. With so many sporting events through the year, what is the likelihood of crime around a sports stadium given event?

Crime happens, given the density of a population, there is some increase in crime.

However, how often does it happen around specific areas like a sports arena? It would be helpful to sports fans if they know the chance that crime around them given the arena.

The local police department could increase/decrease staff given the right information; also, city planners could use the information to determine the best way to use a city's land and resources. The goal of this project is to develop such a predictive model for only crime around stadium arenas in the city of Houston, Texas from the years 2010 to 2017.

Data Acquisition

Crime data

Datasets were acquired from several different sources. The first dataset contains HPD Beat Crime Statistics crime data from the Houston police department and is part of the Uniform Crime Report program or UCR. It complies official data collected by law enforcement agencies across the United States. UCR criminal offenses are divided into two major groups: part I and part II.

Part I offenses are considered to be serious and are broken into two categories: violent and property crimes; they include murder, rape, robbery, aggravated assault, burglary, theft, and auto theft.

Part II offenses are all crime classifications other than those defined as Part I. some of those include: forgery, fraud, vandalism, prostitution, disorderly conduct.

The information contains in the reports are a monthly breakdown of Part I crimes for which HPD wrote police reports. The data shows the number of reports for the following crimes: murder, rape, robbery, aggravated assault, burglary, theft, and auto theft.

The ICR data is provided monthly in Microsoft Access format along with Microsoft Excel spreadsheet format.

Total of 96 files was downloaded, (12 months x 8 years), here is the crime dataset breakdown

Variable	Description
date	Date of offense, include month/date/year
Hour	Approximate time when an event occurs, value form 0-24
Offense Type	Type I offense
Beat	The geographic area of the city broken down for patrol and statistical purpose
Premise	Identify the type of location where crime occurs (apartment complex, parking lot, etc.)
Block Range	The value range of street
Street Name	Name of the street where the offense occurred
Type	Street type, rd, Blvd
Suffix	N, S, E, W
Offenses	Times offense happen within the time frame

Sports data

Houston has four major sports teams and two Division I schools. Data from each sport was acquired from separate locations

Houston Texans

The Houston Texas is a professional American football team based in you guessed it, Houston, Texas. They compete in the National Football League (NFL). Every home game is played at the NRG Stadium (formerly Reliant Stadium). To obtain dates and scores sportradar.us free trial was used.

Variable	Description
schedule	Game date and time
home.alias	Home team
scoring.home_points	Home team score
away.alias	Away team
scoring.away_points	Score from away team
WIN	Team that won

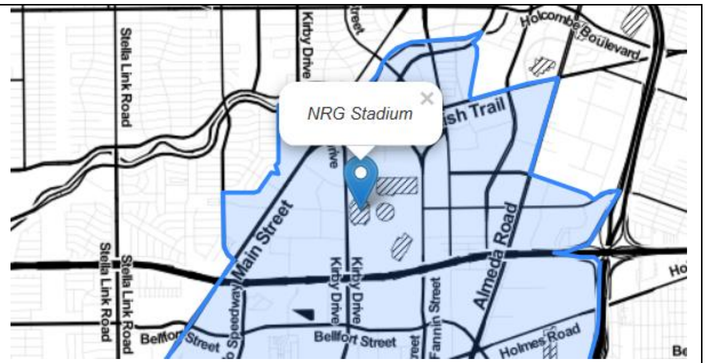
Stadium

Name: NRG Stadium

Address: NRG Pkwy, Houston, TX 77054

Coordinates: 29.684722, -95.410833

Police Beat: 15E40



Houston Astros

Houston Astros are an American professional baseball and current champions in Major League Baseball (MLB). Every home game is played at Minute Maid Park, (formerly Enron field). Dataset was acquired from baseball-reference.com

Variable	Description
Gm#	Game number
Year	Season year
date	Date of game
'blank'	Boxscore, link to more data from this game
Tm	Current team
"blank"	Has two values, "none" or "@
Opp	Opponent team
W/L	Win or lost
R	Runs scored
RA	Runs allowed
INN	More than nine innings?
W-L	Win/loss record
Rank	Current rank
GB	Games back of division/league leader
Time	Time of game
D/N	Day or night game
Attendance	Sum of people attendance of the game

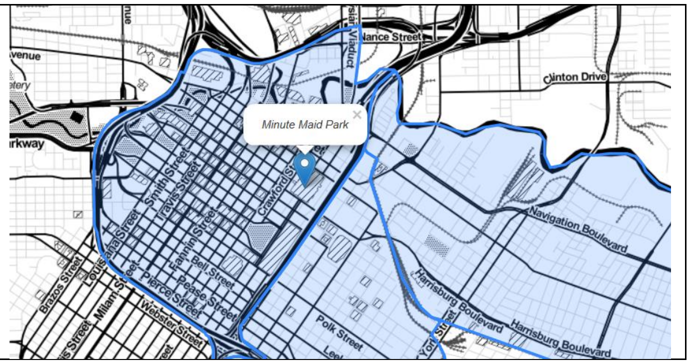
Stadium

Name: Minute Maid Park

Address: 501 Crawford St, Houston, TX 77054

Coordinates: 29.756944, -95.355556

Police Beats: 1A10, 10H30,10H10



Houston Rockets

Houston Rockets are an American basketball team and compete in the Nationa Basketball Association (NBA). Since 2001, every home game is played at the Toyota Center. Game data was acquired from basketball-reference.com

Variable	Description
G	Games
date	Date of game
time	Time value when the game happens
'blank'	Boxscore, link to more data from this game
"blank"	Has two values, "none" or "@
opponent	Opponent team
"blank"	Contains two values 'W" & "L."
'blank	OT?
Tm	Points scored
Opp	Points scored by the opponent team
W	Wins
L	Losses
Streak	Games won or lost in a row.

Stadium

Name : Toyota Center

Address: 1510 Polk St, Houston, TX 77054

Coordinates: 29.750833, -95.362222

Police Beats: 1A10, 10H30,10H40, 10H50

Houston Dynamo

Houston Dynamo is an American professional soccer club that competes in the Major League Soccer (MLS). Every home game is played at BBVA Compass Stadium. Game data was acquired from github repo FootballData.

Variable	Description
full_date	Date of games
home_team	Local team
home_score	Local team score
away_team	Away team
away_score	Away team score
winner	Winner of match

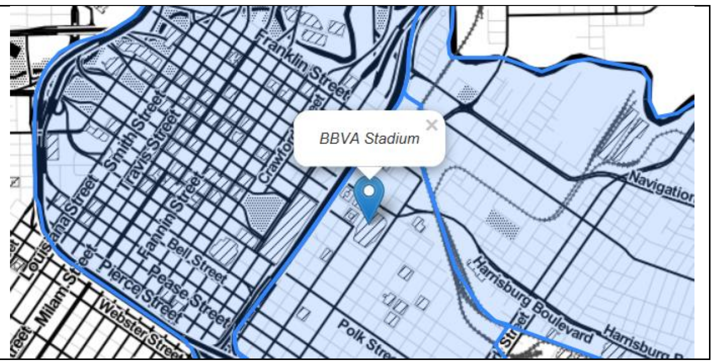
Stadium

Name: BBVA Compass Stadium

Address: 2200 Texas Ave Houston, TX

Coordinates: 29.7522, -95.3524

Police Beats: 1A10, 10H30, 10H10



University of Houston Football

University of Houston football program is an NCAA Division I college football. Every home game is played at TDECU Stadium, which was built on the site formerly occupied by Robertson Stadium, where they played before. Game data was acquired from sports-reference.com

Variable	Description
G	Games
date	Date of game
time	Time value when the game happens
day	weekday
school	Home team
“blank”	Has two values, “none” or “@
opponent	Opponent school
conf	Conference
“blank”	Contains two values ‘W” & “L.”
Pts	Points scored by the “School” team
Opp	Points scored by the opponent team
W	Wins
L	Losses
TV	Channel this game will be on

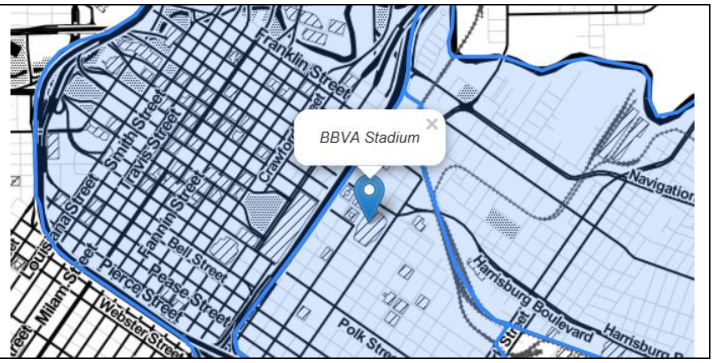
Stadium

Name: BBVA Compass Stadium

Address: 2200 Texas Ave Houston, TX

Coordinates: 29.7522, -95.3524

Police Beats: 1A10, 10H30, 10H10



Rice University Football

Rice Owls football program is an NCAA Division 1 college football. Every home game is played at Rice Stadium. Game data was acquired from sports-reference.com.

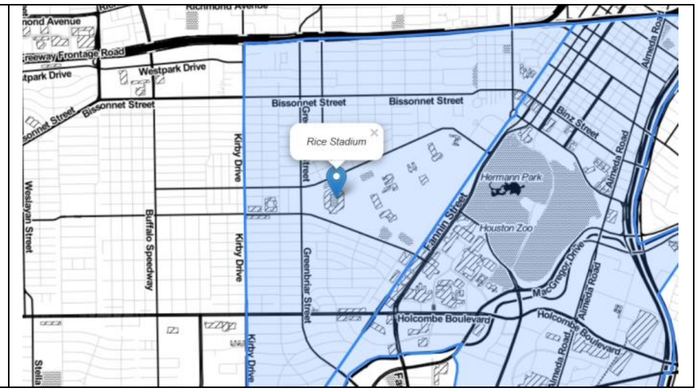
Variable	Description
G	Games
date	Date of game
time	Time value when the game happens
day	weekday
school	Home team
"blank"	Has two values, "none" or "@
opponent	Opponent school
conf	Conference
"blank"	Contains two values 'W" & "L."
Pts	Points scored by the "School" team
Opp	Points scored by the opponent team
W	Wins
L	Losses
TV	Channel this game will be on

Stadium

<table>
<tr>
<td>Name: Rice Stadium

Address: 610 South Main St Houston, TX

Coordinates: 29.721944, -95.349167

Police Beats: 10H70, 10H80</td>
<td>
</td>
</tr>
</table>

Data Cleaning

Tools and Libraries

- *Pandas* used to analyze the data
- *glob*: a python module that implements globbing of directory contents
- *os*: a python module that allows python to 'talk' to the operating system.
- *NumPy*: a powerful scientific computing library in python.

Crime data

Excel files were combine into one data frame

```
# combine all files into one df
all_files = glob.glob(os.path.join(path, "*.xls"))
df_from_each_file = (pd.read_excel(f) for f in all_files)
df   = pd.concat(df_from_each_file, ignore_index=True)
```

Several columns were named differently between months

```
# combine similar columns
df['BlockRange'] = pd.concat([df['Block Range'].dropna(),
                              df['BlockRange'].dropna()]).reindex_like(df)
```

Method used to check for null values

```
df.apply(lambda x: sum(x.isnull()))
```

Some values had extra characters or empty space, pandas' methods were used to clean up some columns.

```
# replace extra ' with empty space
crimes['Beat'] = crimes.Beat.str.replace("'", " ")
# strip empty spaces
crimes.Beat = crimes.Beat.str.strip()
```

Setting date column as index

```
# set date as datetime, index & sort
crimes.Date = pd.to_datetime(crimes.Date)
crimes = crimes.set_index('Date').sort_index(ascending=True)
```

Extracting day data from index

```
# get day, weekday,month ,year
crimes['day'] = crimes.index.strftime('%d')
crimes['weekday'] = crimes.index.strftime('%A')
crimes['month'] = crimes.index.strftime('%b')
crimes['year'] = crimes.index.strftime('%Y')
```

After cleaning the crime dataset from 2010 to 2017, they were combined into one data frame

```
df.info()
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1075199 entries, 1914-09-08 to 2033-04-21
Data columns (total 11 columns):
Beat           1075199 non-null object
BlockRange     1075199 non-null object
StreetName     1075192 non-null object
OffenseType    1075199 non-null object
Premise        1075199 non-null object
NumOffenses    1075199 non-null float64
Hour           1075199 non-null float64
day            1075199 non-null object
weekday        1075199 non-null object
month          1075199 non-null object
year           1075199 non-null object
dtypes: float64(2), object(9)
memory usage: 98.4+ MB
```

As we can see from the datetimeIndex range, some values entered wrong. I created a filter to select only events from 2010 to 2017

```
df = df['1/1/2010':'12/31/2017']
df.info()
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1072618 entries, 2010-01-01 to 2017-12-31
Data columns (total 11 columns):
Beat           1072618 non-null object
BlockRange     1072618 non-null object
StreetName     1072611 non-null object
OffenseType    1072618 non-null object
Premise        1072618 non-null object
NumOffenses    1072618 non-null float64
Hour           1072618 non-null float64
day            1072618 non-null object
weekday        1072618 non-null object
month          1072618 non-null object
year           1072618 non-null object
dtypes: float64(2), object(9)
memory usage: 98.2+ MB
```

We now have a semi-clean dataset

```
df.head()
             Beat   BlockRange   StreetName         OffenseType  \
Date
2010-01-01    5F30  13200-13299    northwest              Theft
2010-01-01   20G10    9900-9999     richmond              Theft
2010-01-01   14D20    8500-8599        rubin  Aggravated Assault
2010-01-01   14D40    4200-4299  friar point           Burglary
2010-01-01   10H70    4800-4899       austin           Burglary

                              Premise  NumOffenses  Hour day weekday month  \
```

```
Date
2010-01-01  department/discount store          1.0  22.0   1  Friday   Jan
2010-01-01      apartment parking lot          1.0  16.0   1  Friday   Jan
2010-01-01       road/street/sidewalk          1.0   7.0   1  Friday   Jan
2010-01-01            residence/house          1.0  20.0   1  Friday   Jan
2010-01-01            residence/house          1.0  21.0   1  Friday   Jan

            year
Date
2010-01-01  2010
2010-01-01  2010
2010-01-01  2010
2010-01-01  2010
2010-01-01  2010
```

Based on the location of the stadiums, we will select specific police beats that are within 1 mile radius of each stadium

```
# create a list of Beat names that we want
beats = ['10H10','10H30', '10H40', '10H50', '10H60','10H70', '10H80', '15E40', '1A10']

# filter column based on our list
selected_beats = df.Beat.isin(beats)

# create a new dataframe for each selected beat and save
beat_10H30 = df_sb[df_sb.Beat == '10H30']
beat_10H10 = df_sb[df_sb.Beat == '10H10']
beat_1A10 = df_sb[df_sb.Beat == '1A10']
beat_10H40 = df_sb[df_sb.Beat == '10H40']
beat_15E40 = df_sb[df_sb.Beat == '15H40']
beat_10H50 = df_sb[df_sb.Beat == '10H50']
beat_10H60 = df_sb[df_sb.Beat == '10H60']
beat_10H70 = df_sb[df_sb.Beat == '10H70']
beat_10H80 =  df_sb[df_sb.Beat == '10H80']
```

Changed some values in the Hour column showed 24 instead of 0

```
## change 24 to 0 value
df_sb.Hour.replace(24,0,inplace=True)
```

Sports data

For the sports data

For the Dynamo dataset a function was created to combine several files

```
def cleanup(df,year):
    '''function that cleans up dataframe'''
    df['year'] = year  # create col with var year
    df['full_date'] = df['date'] + ' ' + df['year']  # append date and year cols
    df['full_date'] =  pd.to_datetime(df['full_date'])  # convert full_date to datetime
    df['home_score']= df['result'].str.split('-').apply(lambda x: x[0])  # split score vals
    df['away_score']= df['result'].str.split('-').apply(lambda x: x[1])  # split score vals
    df = df[['full_date','home_team','home_score','away_team','away_score']]  # org df
    # winner cols given value scores
    df['winner'] = np.where(df['home_score'] > df['away_score'], df['home_team'],
df['away_team'])
    df = df.set_index('full_date').sort_index(ascending=True)  # set full_date as index
    return df
```

Astros dataset had parenthesis within the date column, they were removed using regular expression

```
mlb['full_date'] = mlb['full_date'].str.replace(r"\(.*\)"," ")
```

Change the name of columns

```
hou_rockets_plo.rename(columns={
    'Unnamed: 2':'Time',
    'team': 'Team',
    'Tm': 'Team_score',
    'Unnamed: 5': 'Location',
    'Opp': 'Opponent_score',
    'Unnamed: 7': 'Result'
},inplace=True)
```

Create a function to get the median value of block range

```
def block_split(df):
    '''
    split blockrange col values
    then give median value as a string
    '''
    first = df.BlockRange.str.split(pat='-',expand=True)[0].astype('int')
    second = df.BlockRange.str.split(pat='-',expand=True)[1].astype('int')
    med = np.ceil((second + first)/2).astype('int')
    med = med.astype('str')
    street = df.StreetName
    return med
```

Create a new column with a correct address

```
df['address'] = df[['block', 'StreetName']].apply(lambda x: ' '.join(x), axis=1)
```

create a new column that calls google maps API that returns various data given a full address.

```python
def gm_geocode(address,API_KEY):
    loc = '{}, Houston, TX'.format(address)
    gmaps = googlemaps.Client(key=API_KEY)
    r = gmaps.geocode(loc)
    #lat_lng = tuple(r[0]['geometry']['location'].values())
    #full_add = r[0]['formatted_address']
    #return lat_lng, full_add
    return r
```

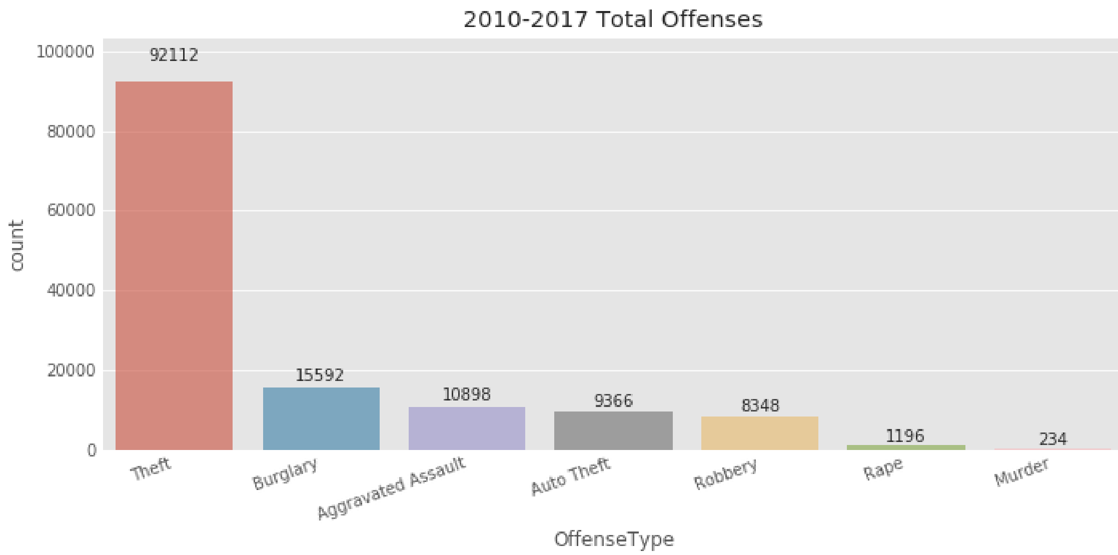
```python
df['tup_add'] = df['address'].apply(gm_geocode,args=(API_KEY,))
```

Data Exploration

Only selected Beats

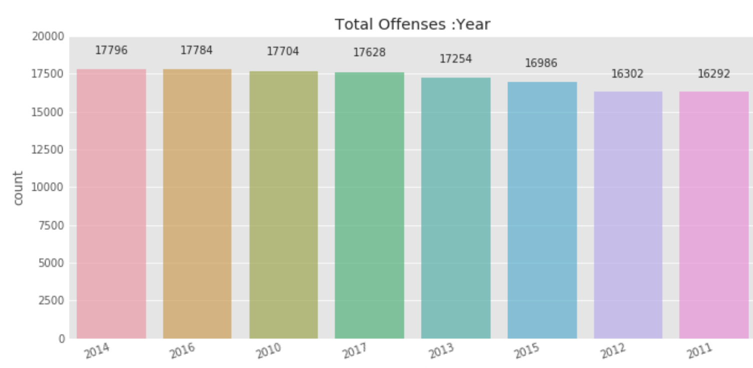
```
df.Beat.value_counts(dropna=False)
1A10     30650
10H70    22846
10H40    20920
10H50    16034
10H60    15394
10H80    14962
10H30     8796
10H10     8144
Name: Beat, dtype: int64
```

Sum of Total Offenses



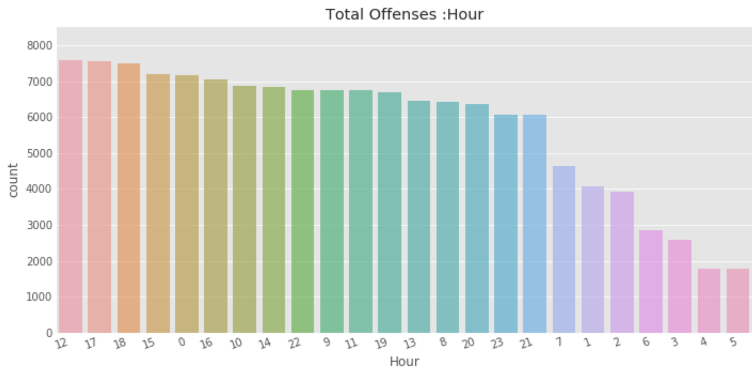
Out of 137746 crimes committed in the 8 Police beats from 2010 to 2017 Theft was the most common by a long shot. It surpasses all other offense types combined.

Sum of Offenses by Year



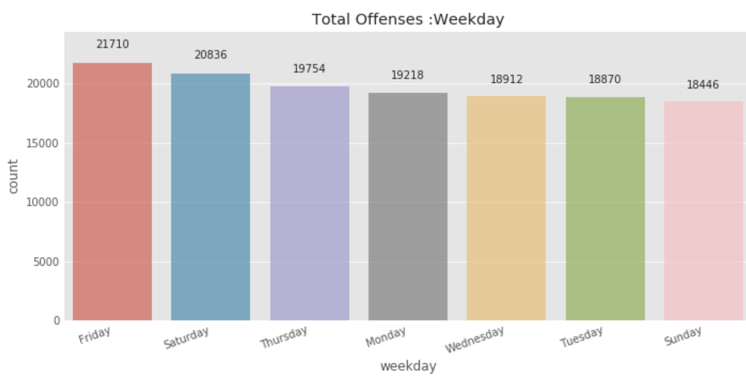
Out of the eight years, 2014 had the most offenses by just a few. 2011 and 2012 had the fewest in the 16000 range

Sum of Offenses by Hour



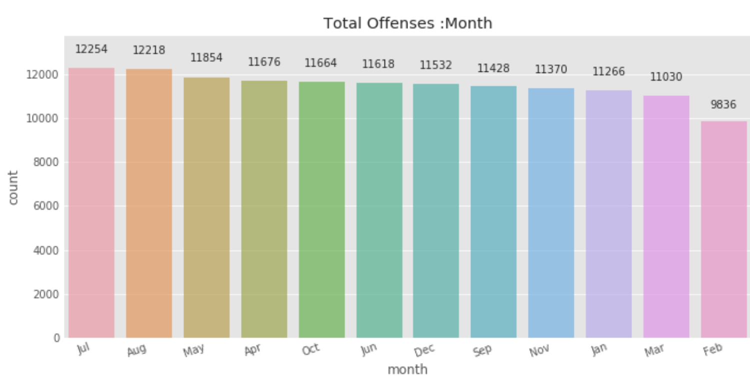
Midday is the most popular time when a crime is committed followed by 7 pm. It looks like 4-5 am are the lowest, but it could be that crimes are not reported during those time because most people are asleep.

Sum of Offenses by Weekday



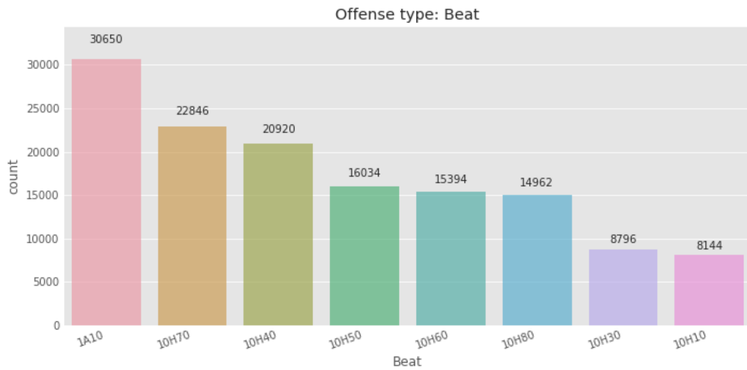
It seems that Friday is the most popular with 21710 offenses followed by Saturday with 20836. Sunday has the lowest crimes reported with 18446 offenses, almost 2000 less than Saturday

Sum of Offenses by Month



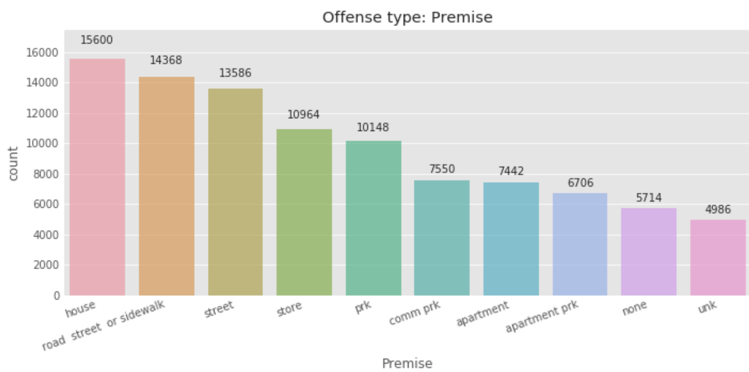
I don't know if it's because of the hot weather, but July is the highest month with 12,254 crimes committed. February is only two days short days (not counting leap years) and its 1,194 crimes shorter that March. Could 1000 crimes happen in 48 hrs?

Sum of Offenses by Beat



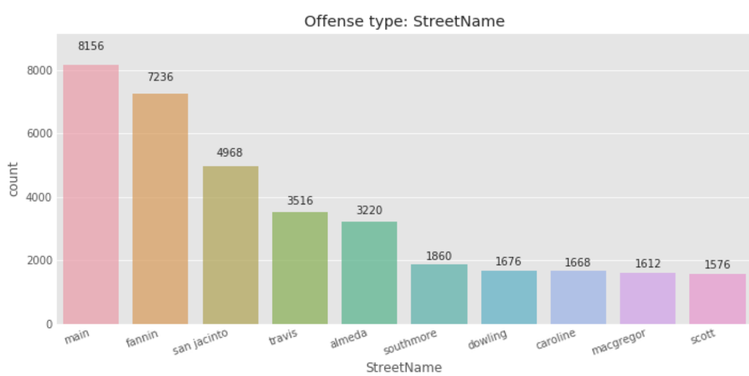
Police beat 1A10 has the highest crime with 30,650. Its probably because it's the center of downtown

Sum of Offenses by Premise



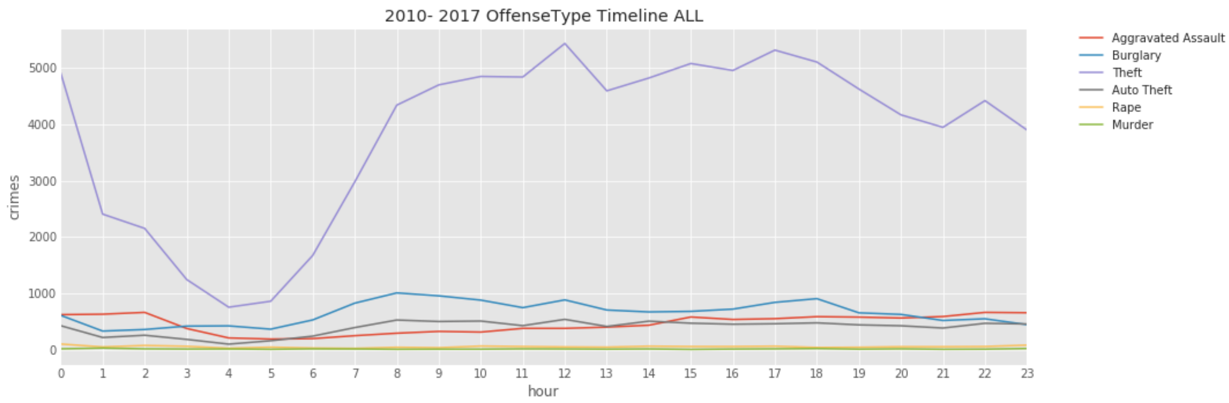
House or home is the highest with 15,600. Streets and sidewalks are also popular; a few unknown missing data fill the bottom two columns.

Sum of Offenses by Street name

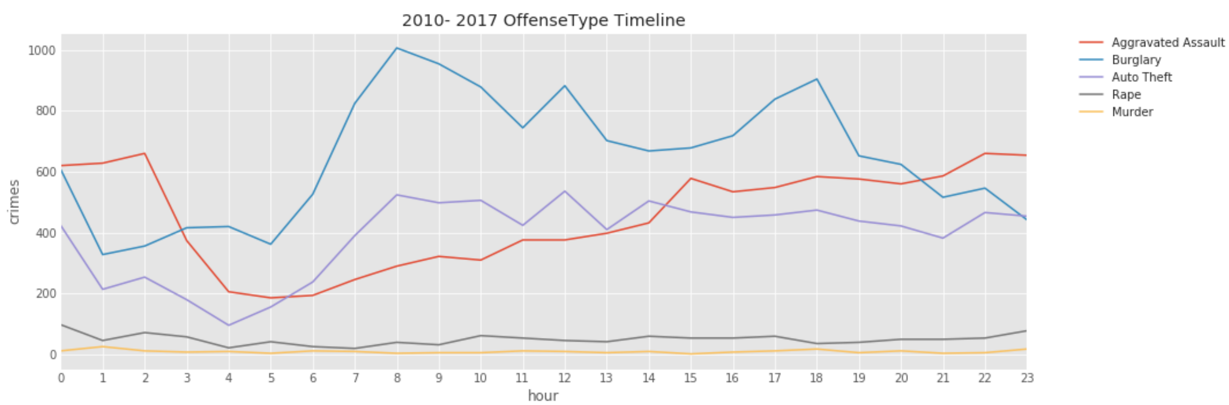


The top 4 streets are all in the area of downtown. With Main street as the most popular.

Timeline

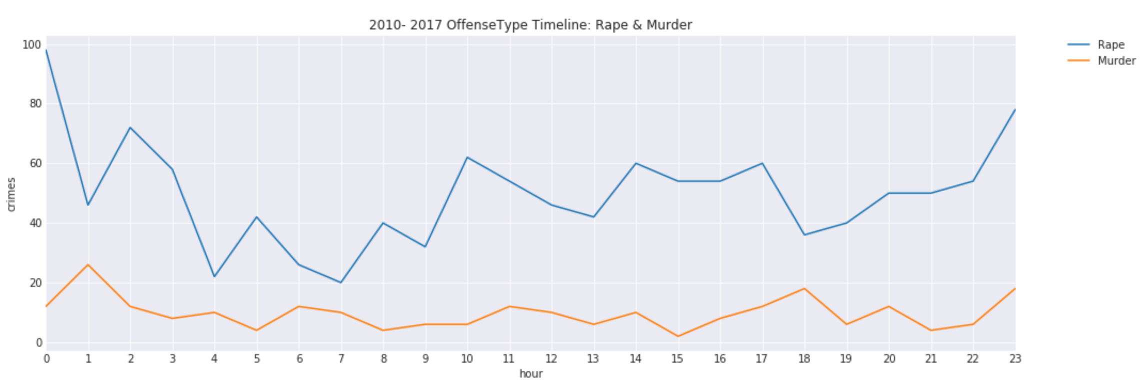


Theft seems to overflow the timeline since it has the most values. Let's remove it from the graph and plot the rest.



We can see that all crime drops between 2 am and 6 am. Burglary and auto theft are at their highest at 8 am. Aggravated Assault peaks at 3 pm.

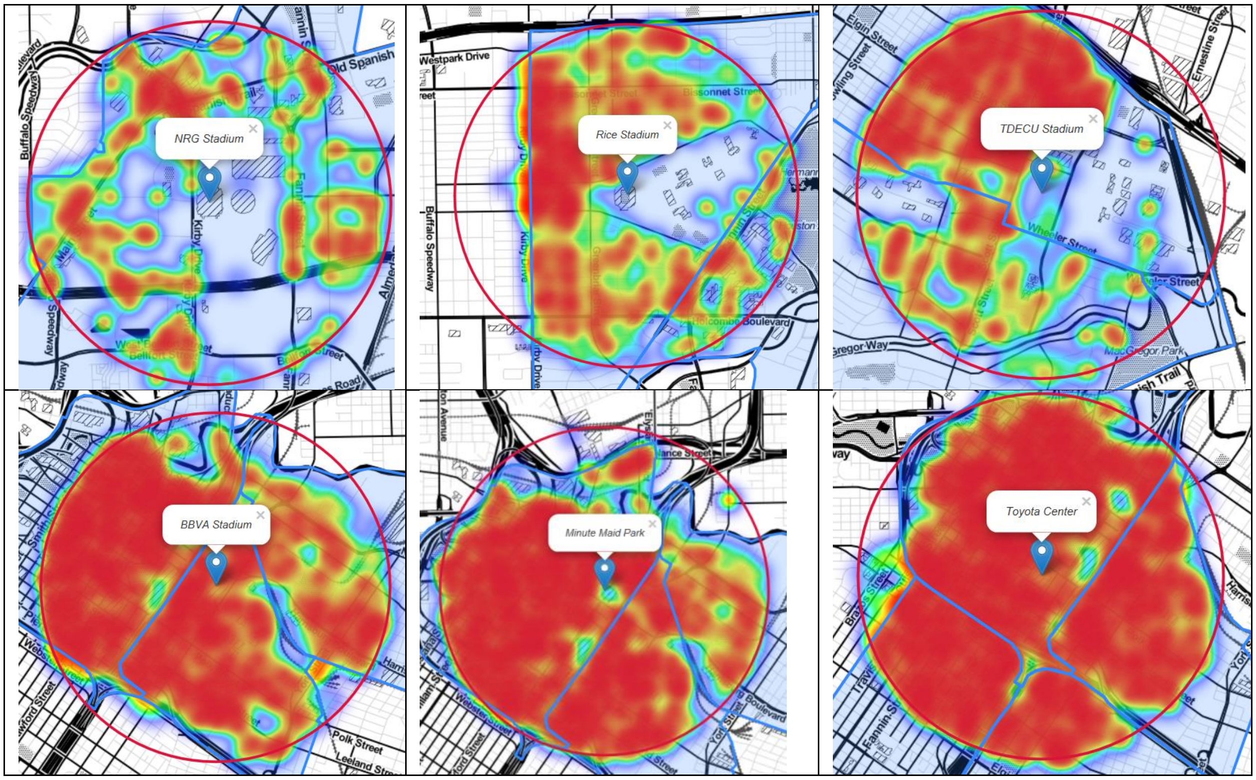
Rape and Murder are too low to differentiate. Let's plot them separately.



3 pm is the lowest value for murder, and one is the highest. Midnight is the heist with rape with 4 am, and 7 am the lowest.

Stadium data

To visualize the crime at each location, I plotted the stadium with a heat map of crime with a 1-mile radius



Just by looking at these heat maps, NRG stadiums seem to have the least crimes followed by Rice Stadium BBVA Stadium, Minute Maid Park and Toyota Center are less than half a mile apart.

Modeling

We have now cleaned the crime data that span from 2010 to 2017 and selected only eight police beats that surround each stadium; we will now join the score and schedule data for each team with the crime data.

Data Pre-processing

Created an empty data frame with days

```
days = pd.date_range(start='01/01/2010', end='12/30/2017')
days = pd.DataFrame(days)

days.columns = ['days']
days = days.set_index('days').sort_index(ascending=True)
```

Merge days data frame with crime data frame

```
calendar_crimes = pd.merge(days,date_crimes,  left_index=True, right_index=True, how='left')
calendar_crimes.head()
```

merge calendar_crimes dataframe with scores dataframe

```
merge_data = pd.merge(calendar_crimes,df,  left_index=True, right_index=True, how='left')

# change column names
merge_data.columns = ['offenses','away_team','win']
merge_data.index.name = 'date'
```

Create a function that returns values bases on game score or game scheduled

```
def game_feature(df):
    if df.win == 1:
        val = 'Won Game'
    elif df.win == 0:
        val = 'Lost Game'
    else:
        val = 'No Game'
    return val

merge_data['game'] = merge_data.apply(game_feature,axis=1)
```

```
           OffenseType     Premise  hour weekday month  year  dist_stadium  \
date
2010-01-01  Auto Theft  bar_nc prk     0  Friday   Jan  2010      0.137184
2010-01-01       Theft      bar_nc     0  Friday   Jan  2010      0.549562
2010-01-01    Burglary  office bld     0  Friday   Jan  2010      0.480008
2010-01-01       Theft         unk     0  Friday   Jan  2010      0.734357
2010-01-01       Theft  convention     0  Friday   Jan  2010      0.403381

               game
date
2010-01-01  No Game
2010-01-01  No Game
2010-01-01  No Game
2010-01-01  No Game
```

Feature Extraction

Create a function that extract part of the day feature from hour column

```
def day_feature(df):
    mo = [6,7,8,9,10,11]  # morning, sunrise to 11
    af = [12,13,14,15,16]  # afternoon to fiveish
    ev = [17,18,19,20]   # evening to  sunset
    ni = [21,22,23,0,1,2,3,4,5]  # night, sunset to sunrise
    if df.hour in mo:
        val = 'Morning'
    elif df.hour in af:
        val = 'Afternoon'
    elif df.hour in ev:
        val = 'Evening'
    else:
        val = 'Night'
    return val

df['part_day'] = df.apply(day_feature,axis=1)
```

Create a function that extracts weather season from DateTime index

```
def season_feature(df):
    '''
    spring (March, April, May),
    summer (June, July, August),
    autumn (September, October, November)
    winter (December, January, February).
    '''
    sp = ['Mar','Apr','May']   # spring
    su = ['Jun','Jul','Aug']   # summer
    au = ['Sep','Oct','Nov'] # autumn/fall
    wi = ['Dec','Jan','Feb']  # winter
    if df.month in sp:
        val = 'Spring'
    elif df.month in su:
        val = 'Summer'
    elif df.month in au:
        val = 'Autumn'
    else:
        val = 'Winter'
    return val

df['season'] = df.apply(season_feature,axis=1)
```

```
           OffenseType     Premise  hour weekday month  year  dist_stadium  \
date
2010-01-01  Auto Theft  bar_nc prk     0  Friday   Jan  2010      0.137184
2010-01-01       Theft      bar_nc     0  Friday   Jan  2010      0.549562
2010-01-01    Burglary  office bld     0  Friday   Jan  2010      0.480008
2010-01-01       Theft         unk     0  Friday   Jan  2010      0.734357
2010-01-01       Theft  convention     0  Friday   Jan  2010      0.403381

               game part_day  season
date
2010-01-01  No Game    Night  Winter
2010-01-01  No Game    Night  Winter
2010-01-01  No Game    Night  Winter
2010-01-01  No Game    Night  Winter
2010-01-01  No Game    Night  Winter
```

Groupby date index and get mode values

```
# get mode value of of part_day column
df['part_day_mode'] = df.groupby(df.index)['part_day'].agg(lambda x: scipy.stats.mode(x, axis=None)[0][0])
# get mode value of hour column
df['hour_mode'] = df.groupby(df.index)['hour'].agg(lambda x: scipy.stats.mode(x, axis=None)[0][0])
#get mode value of Premise column
df['premise_mode'] = df.groupby(df.index)['Premise'].agg(lambda x: scipy.stats.mode(x, axis=None)[0][0])
# get mode value from offenseType column
df['offenseType_mode'] = df.groupby(df.index)['OffenseType'].agg(lambda x: scipy.stats.mode(x,
axis=None)[0][0])
```

```
           OffenseType     Premise  hour weekday month  year  dist_stadium  \
date
2010-01-01  Auto Theft  bar_nc prk     0  Friday   Jan  2010      0.137184
2010-01-01       Theft      bar_nc     0  Friday   Jan  2010      0.549562
2010-01-01    Burglary  office bld     0  Friday   Jan  2010      0.480008
2010-01-01       Theft         unk     0  Friday   Jan  2010      0.734357
2010-01-01       Theft  convention     0  Friday   Jan  2010      0.403381

               game part_day  season part_day_mode  hour_mode premise_mode  \
date
2010-01-01  No Game    Night  Winter         Night          0       bar_nc
2010-01-01  No Game    Night  Winter         Night          0       bar_nc
2010-01-01  No Game    Night  Winter         Night          0       bar_nc
2010-01-01  No Game    Night  Winter         Night          0       bar_nc
2010-01-01  No Game    Night  Winter         Night          0       bar_nc

           offenseType_mode
date
2010-01-01            Theft
2010-01-01            Theft
2010-01-01            Theft
2010-01-01            Theft
```

Select specific columns

```
df = df[['OffenseType', 'weekday', 'month', 'year',
'dist_stadium', 'game', 'season', 'part_day_mode',
'hour_mode', 'premise_mode', 'offenseType_mode']]
```

Finalize dataset by grouped by date index and getting median value of dist_stadium

```
cdf = df.groupby(df.index).agg(
    {'OffenseType':'count',
     'weekday':'first',
     'month':'first',
     'year': 'first',
     'dist_stadium':'median',
     'season':'first',
     'part_day_mode':'first',
     'hour_mode':'first',
     'premise_mode':'first',
     'offenseType_mode':'first',
    'game':'first'})

cdf.head()
```

```
            OffenseType   weekday month  year  dist_stadium  season  \
date
2010-01-01           16    Friday   Jan  2010      0.499216  Winter
2010-01-02           12  Saturday   Jan  2010      0.575038  Winter
2010-01-03           10    Sunday   Jan  2010      0.493969  Winter
2010-01-04            5    Monday   Jan  2010      0.648818  Winter
2010-01-05            7   Tuesday   Jan  2010      0.706555  Winter

           part_day_mode  hour_mode premise_mode offenseType_mode     game
date
2010-01-01         Night          0       bar_nc            Theft  No Game
2010-01-02         Night         14       street            Theft  No Game
2010-01-03         Night          0       street            Theft  No Game
2010-01-04     Afternoon         14        store            Theft  No Game
2010-01-05       Morning          6     comm bld            Theft  No Game
```

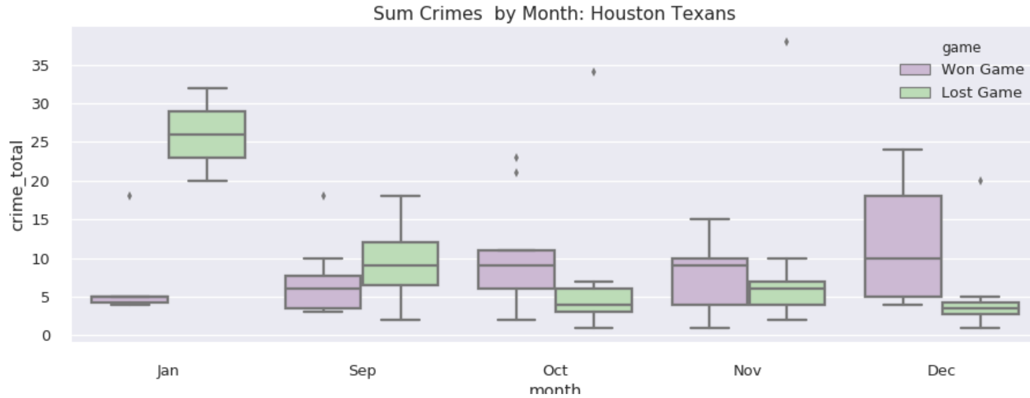
Rename columns again for simple understanding

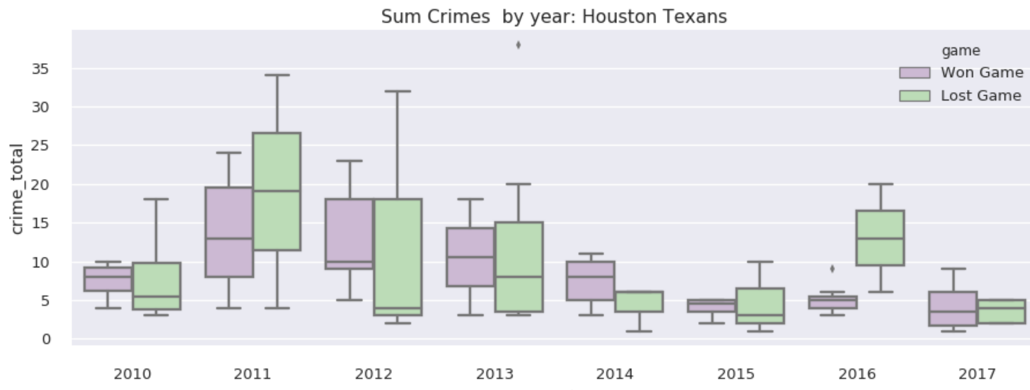
```
cdf.rename(columns={
       'OffenseType':'crime_total',
       'dist_stadium':'dist_stadium_meadian',
       'offenseType_mode':'offense_mode'}, inplace=True)
```

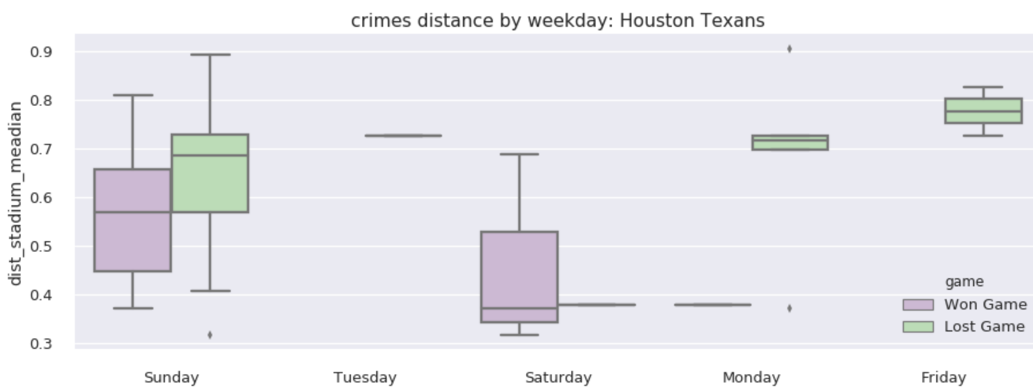
```
            crime_total   weekday month  year  dist_stadium_meadian  season  \
date
2010-01-01           16    Friday   Jan  2010              0.499216  Winter
2010-01-02           12  Saturday   Jan  2010              0.575038  Winter
2010-01-03           10    Sunday   Jan  2010              0.493969  Winter
2010-01-04            5    Monday   Jan  2010              0.648818  Winter
2010-01-05            7   Tuesday   Jan  2010              0.706555  Winter

           part_day_mode  hour_mode premise_mode offense_mode     game
date
2010-01-01         Night          0       bar_nc        Theft  No Game
2010-01-02         Night         14       street        Theft  No Game
2010-01-03         Night          0       street        Theft  No Game
2010-01-04     Afternoon         14        store        Theft  No Game
2010-01-05       Morning          6     comm bld        Theft  No Game
```

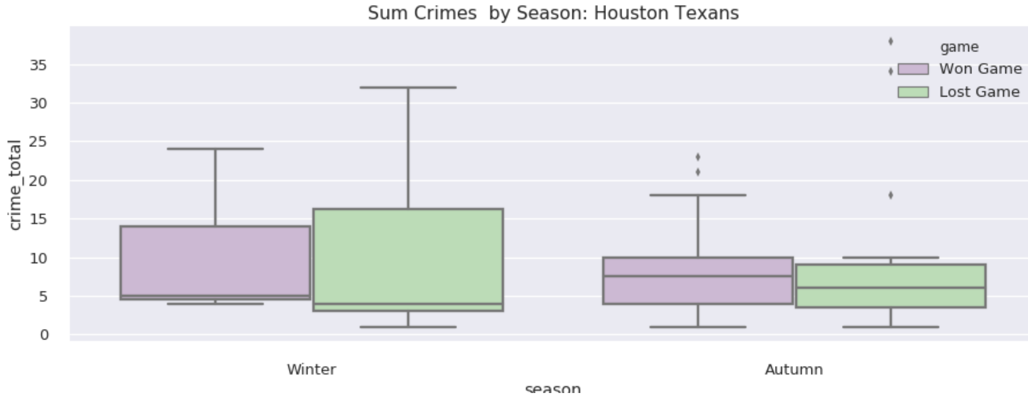
EDF

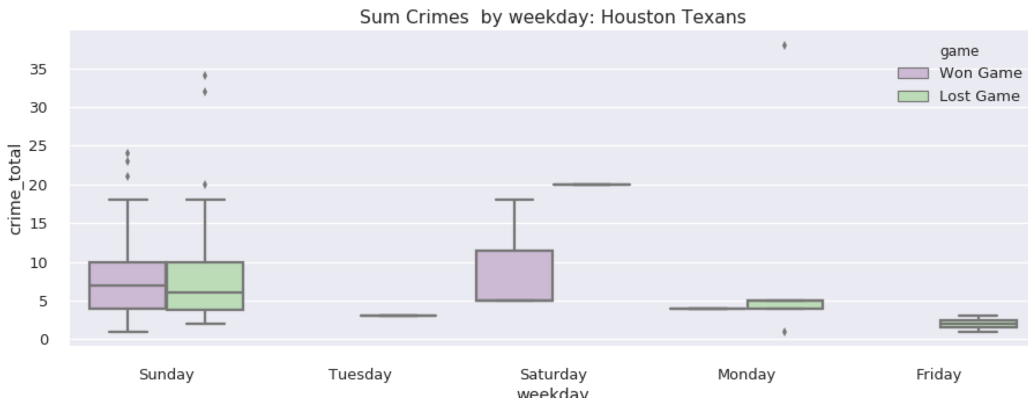
Football: Texans



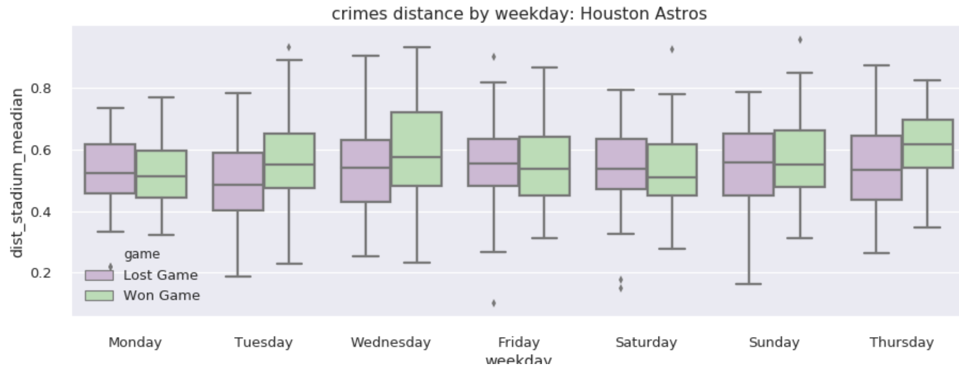


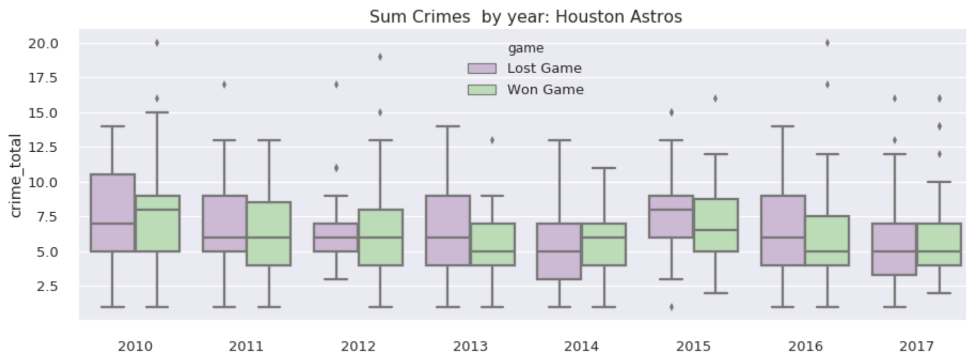


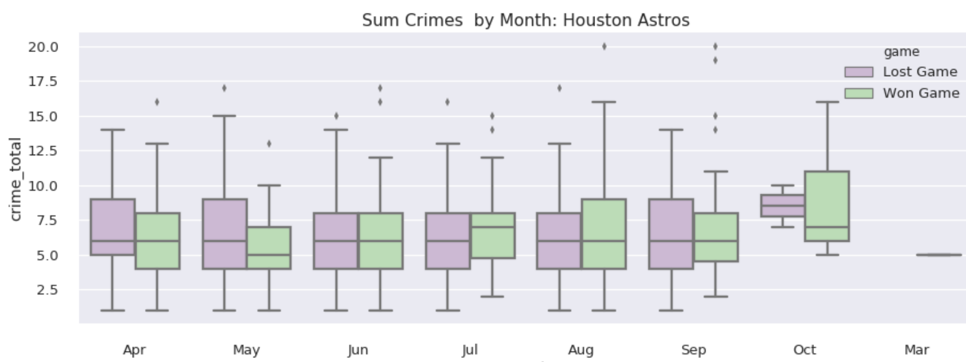

Sum Crimes by Season: Houston Texans
game
Won Game
Lost Game
crime_total
35
30
25
20
15
10
5
0
Winter
Autumn
season

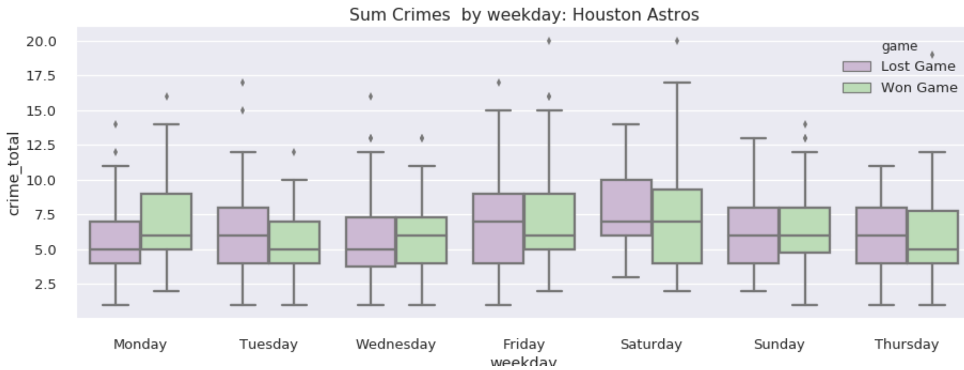

Sum Crimes by weekday: Houston Texans
game
Won Game
Lost Game
crime_total
35
30
25
20
15
10
5
0
Sunday
Tuesday
Saturday
Monday
Friday
weekday

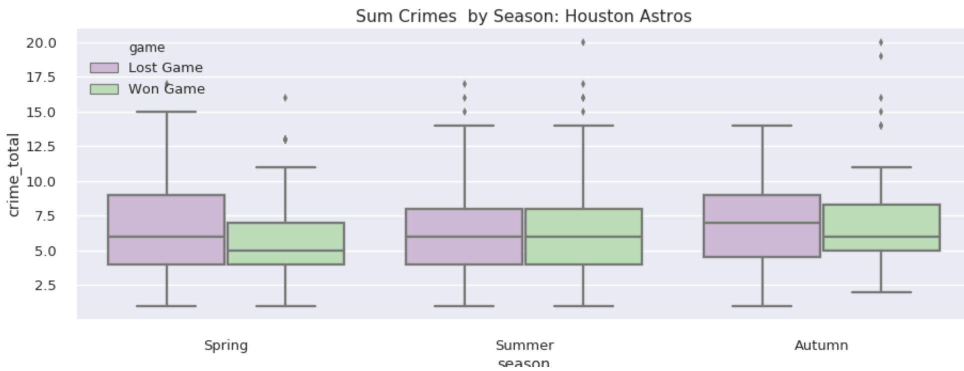
Baseball: Astros



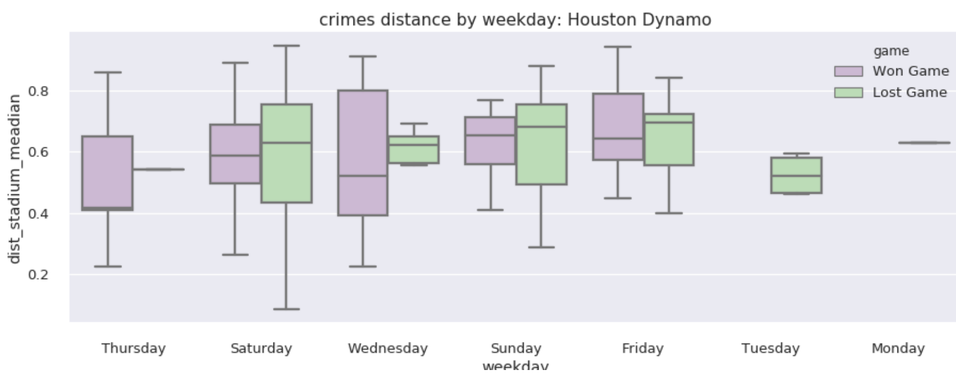


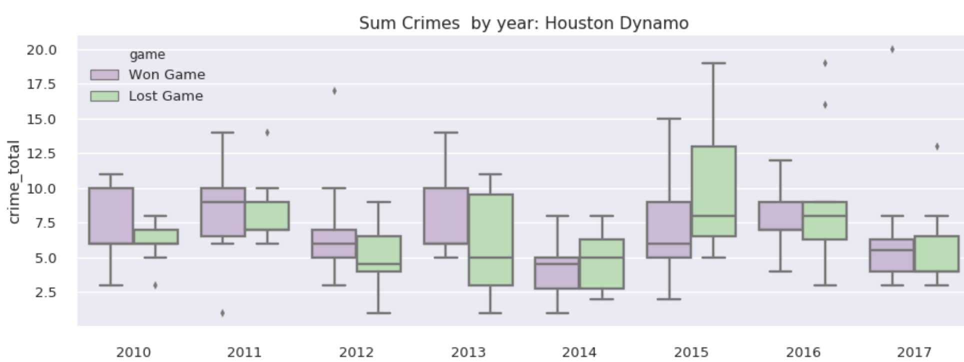


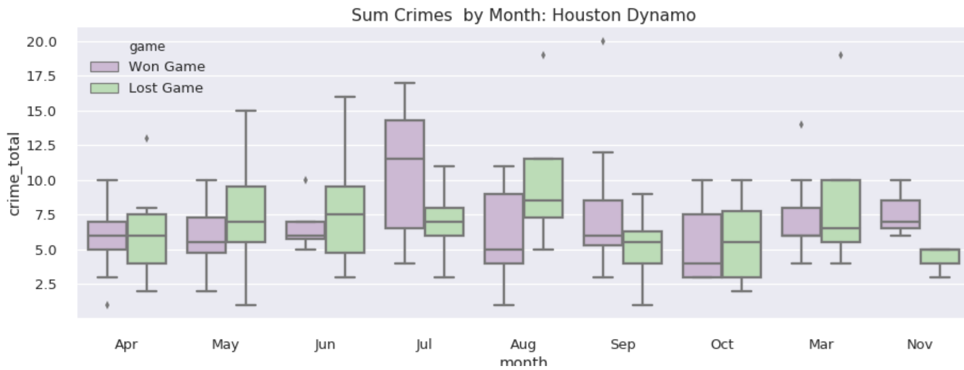


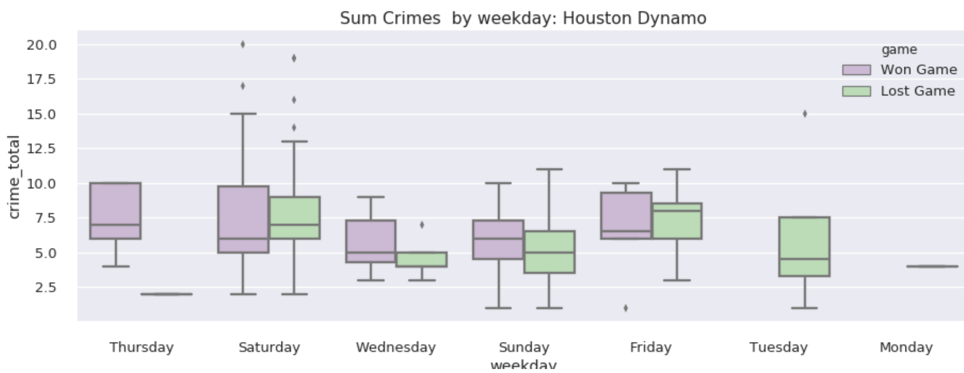


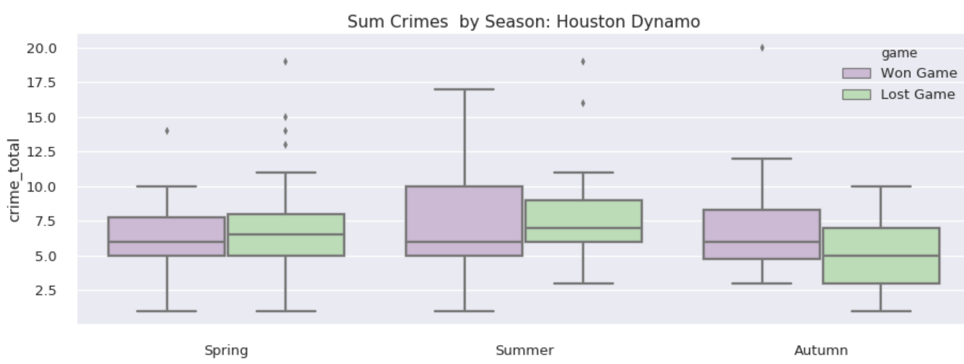
Soccer: Dynamo



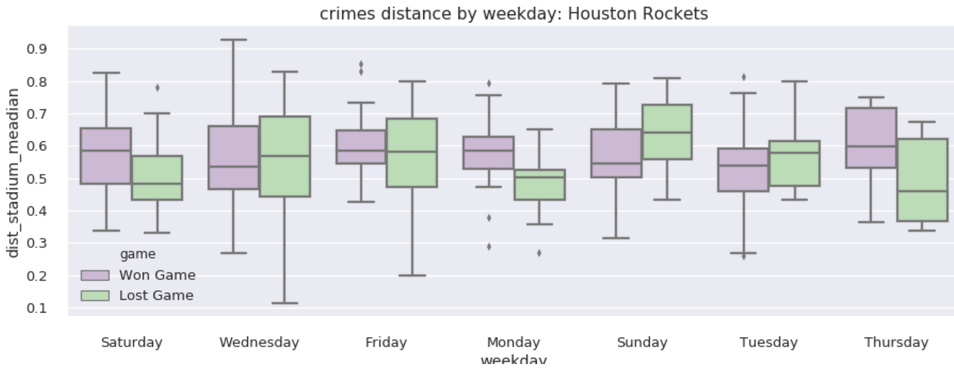


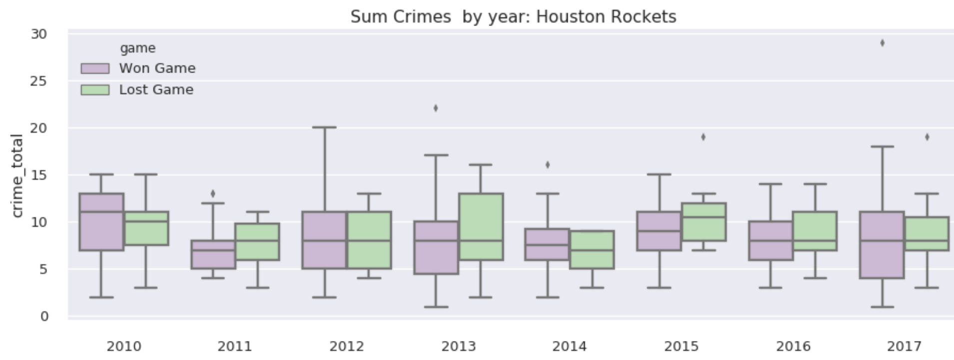

Sum Crimes by Month: Houston Dynamo
game
Won Game
Lost Game
crime_total
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
Apr
May
Jun
Jul
Aug
Sep
Oct
Mar
Nov
month

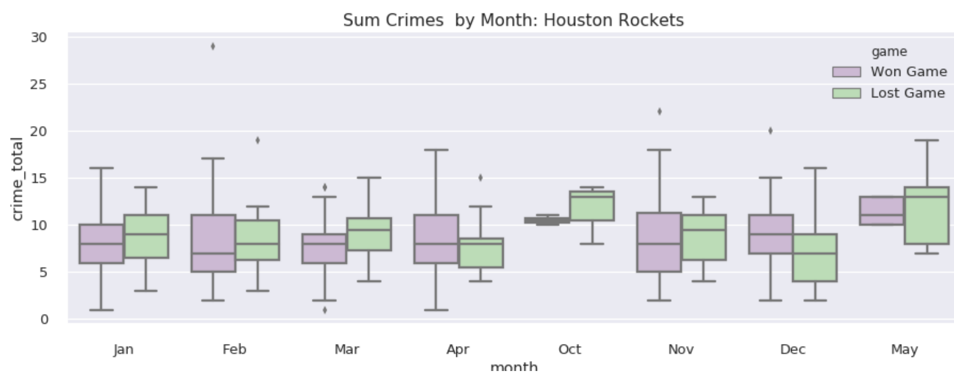

Sum Crimes by weekday: Houston Dynamo
game
Won Game
Lost Game
crime_total
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
Thursday
Saturday
Wednesday
Sunday
Friday
Tuesday
Monday
weekday

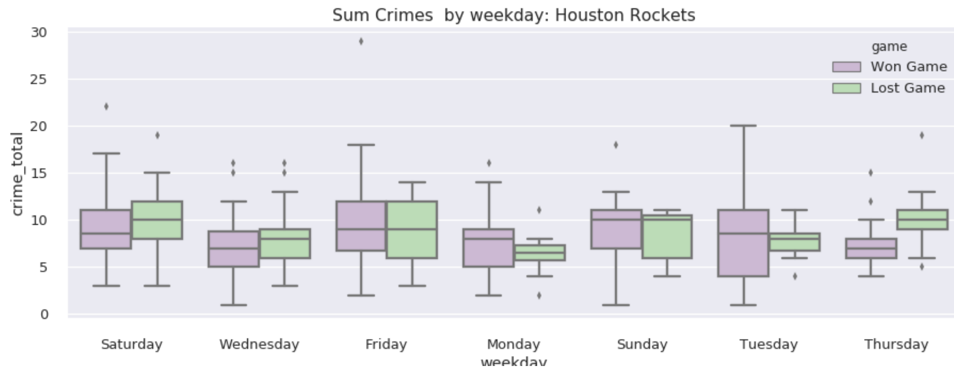

Sum Crimes by Season: Houston Dynamo
game
Won Game
Lost Game
crime_total
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
Spring
Summer
Autumn
season

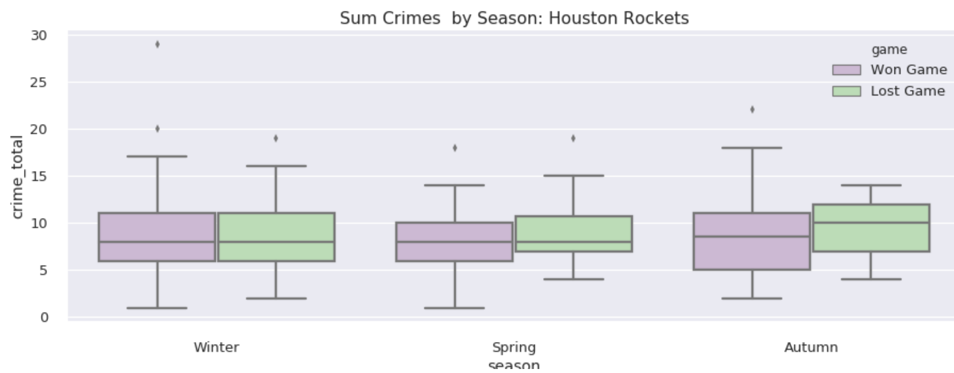
Basketball: Rockets



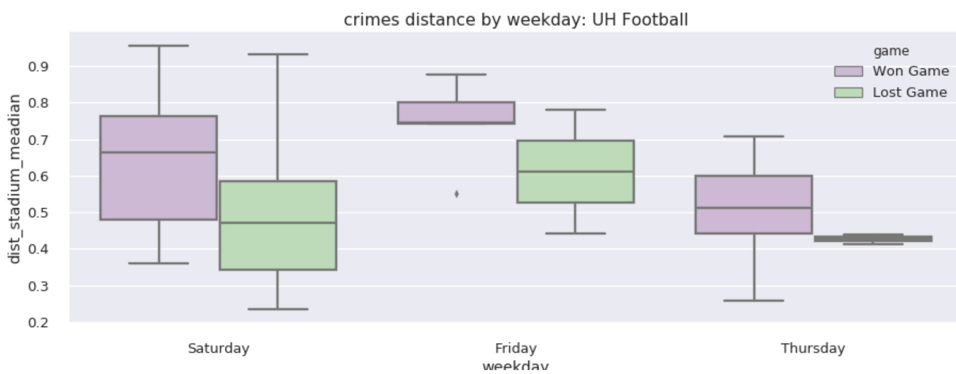


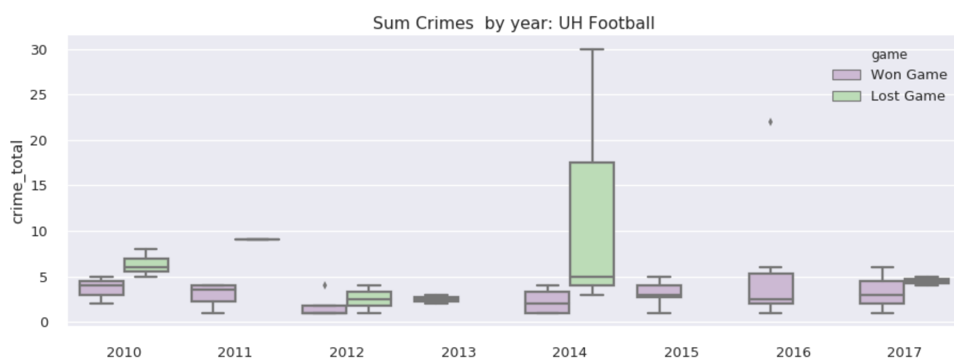


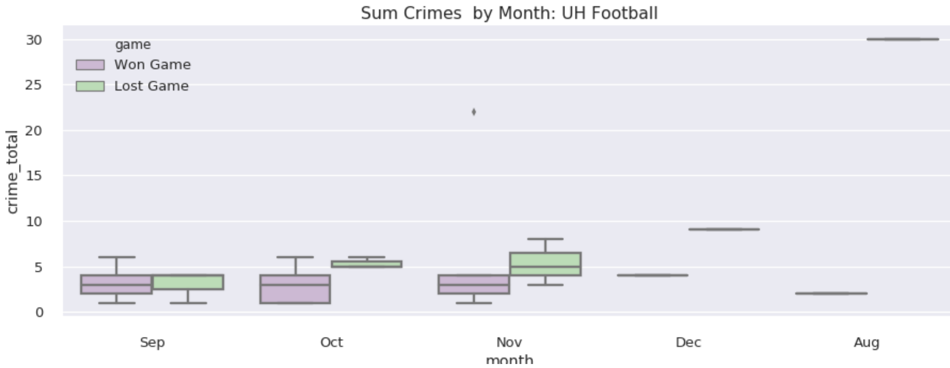


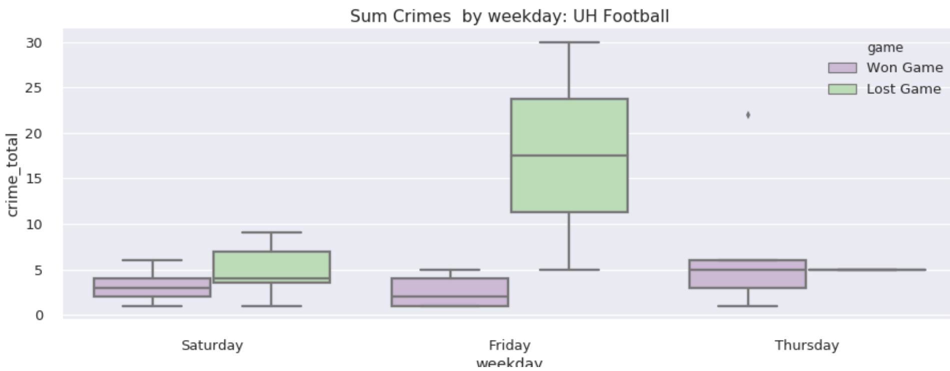


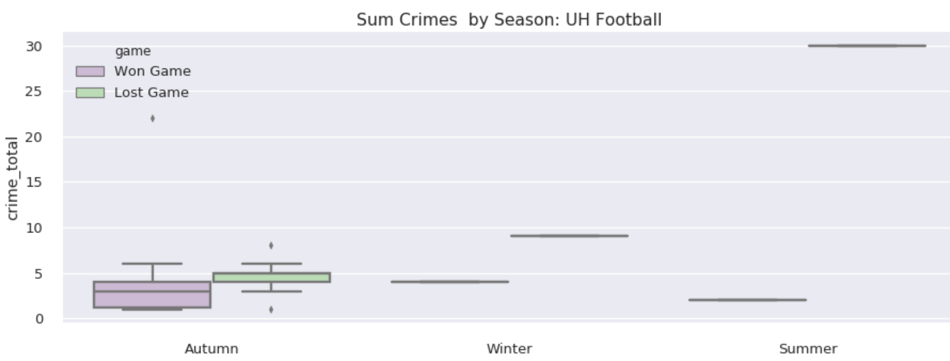
College Football: University of Houston



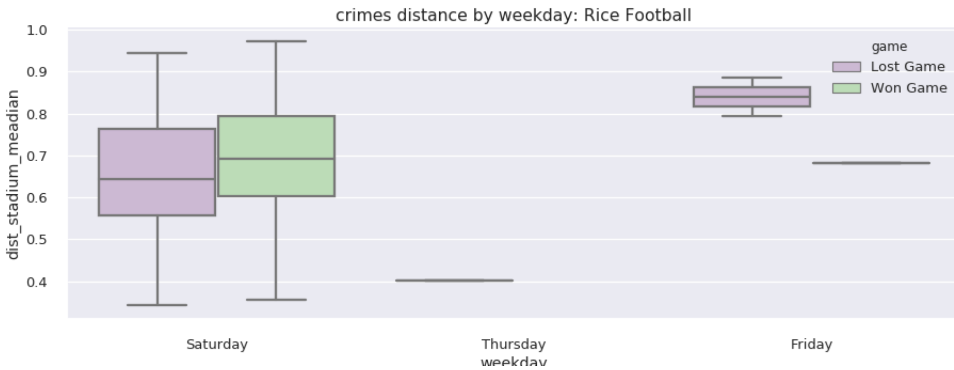


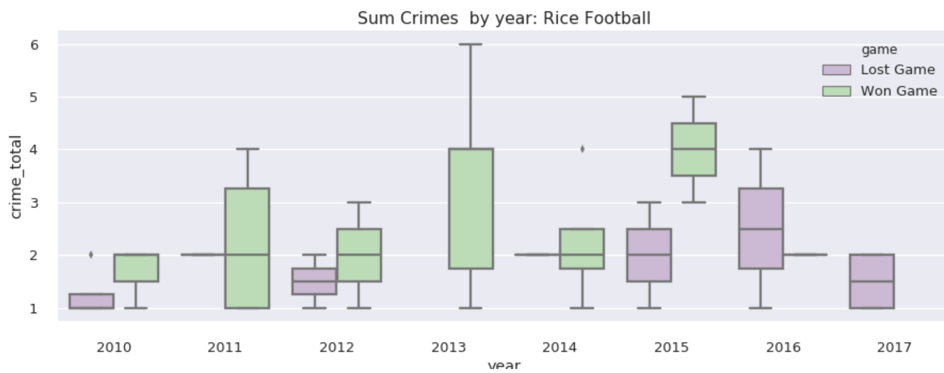


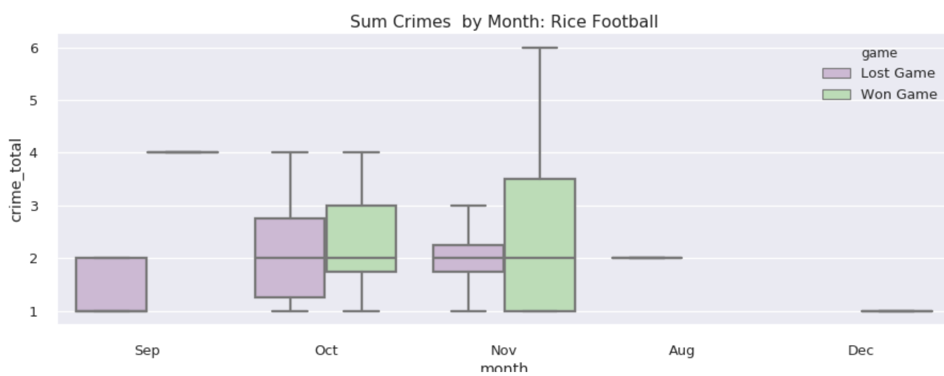


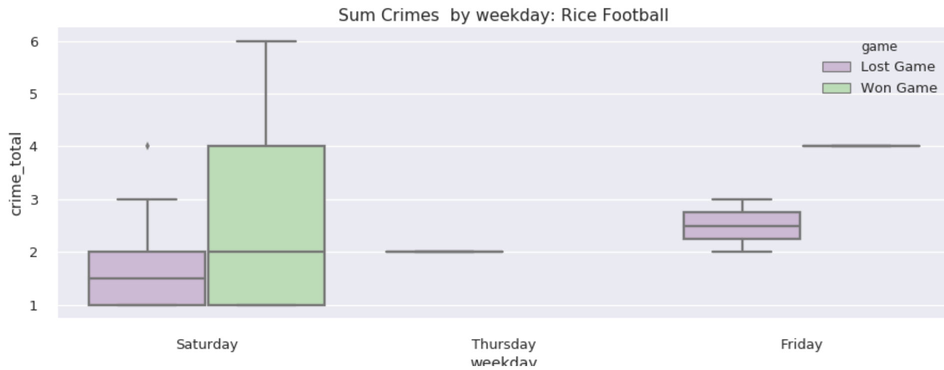


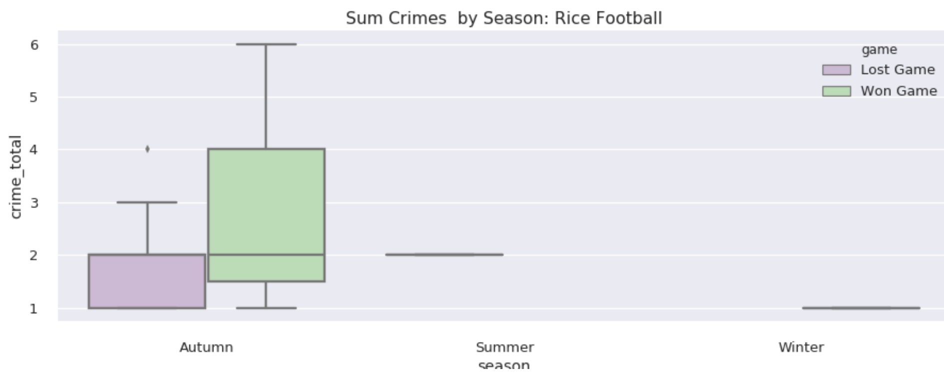
College Football: Rice University


crimes distance by weekday: Rice Football
dist_stadium_meadian
1.0
0.9
0.8
0.7
0.6
0.5
0.4
game
Lost Game
Won Game
Saturday
Thursday
Friday
weekday


Sum Crimes by year: Rice Football
crime_total
6
5
4
3
2
1
game
Lost Game
Won Game
2010
2011
2012
2013
2014
2015
2016
2017
year


Sum Crimes by Month: Rice Football
crime_total
6
5
4
3
2
1
game
Lost Game
Won Game
Sep
Oct
Nov
Aug
Dec
month





Using Models

A function was created to expedite process

```
def modelfit(alg, X,y):
    '''target = y, predictors = X, alg = algorithm used
    '''
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)
    #Fit the algorithm on the data
    alg.fit(X_train,y_train)
    #Predict training set:
    train_predictions = alg.predict(X_train)
    #Perform cross-validation:
    cv_score = cross_val_score(alg, X, y, cv=10, scoring='neg_mean_squared_error')
    cv_score = np.sqrt(np.abs(cv_score))

    #Print model report:
    print ("\nModel Report")
    print( "RMSE : %.4g" % np.sqrt(metrics.mean_squared_error(y_train, train_predictions)))
    print ("CV Score : Mean  %.4g | Std  %.4g | Min  %.4g | Max  %.4g" % \
           (np.mean(cv_score),np.std(cv_score),np.min(cv_score),np.max(cv_score)))
```

- **Linear Regression:** Ordinary least squares Linear Regression.
- **Ridge:** Linear least squares with l2 regularization.
- **Lasso:** Linear Model trained with L1 prior as regularized.

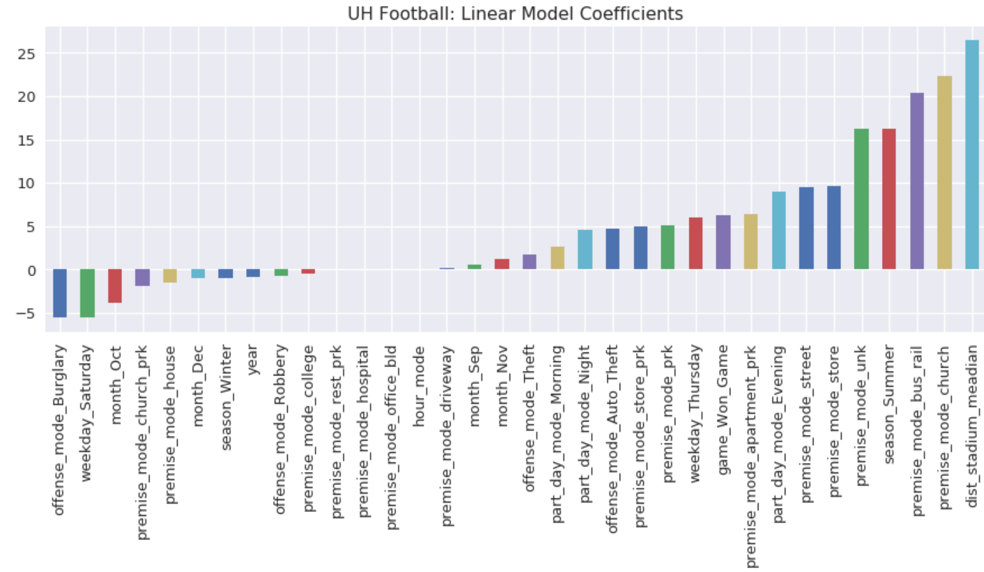
```
alg1 = LinearRegression(normalize=True)
alg2 = Ridge(alpha=0.1,normalize=True)
alg3 = Lasso(alpha=0.1,normalize=True)
```

Results

College Football: University of Houston

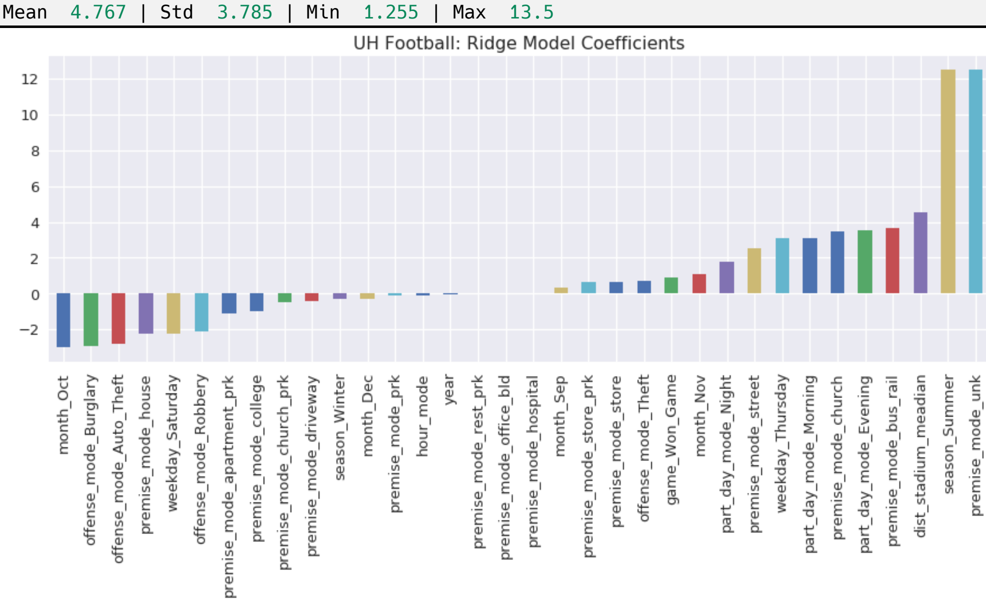
Linear

```
Model Report
RMSE : 1.077
CV Score : Mean  5.234e+13 | Std  1.317e+14 | Min  3.493 | Max  4.45e+14
```



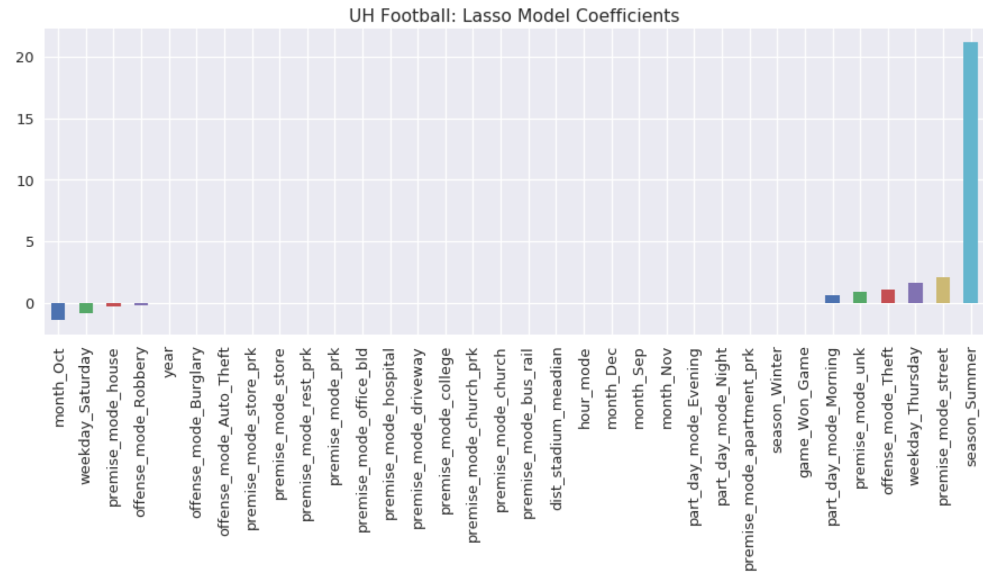
Ridge

```
Model Report
RMSE : 1.704
CV Score : Mean  4.767 | Std  3.785 | Min  1.255 | Max  13.5
```



Lasso

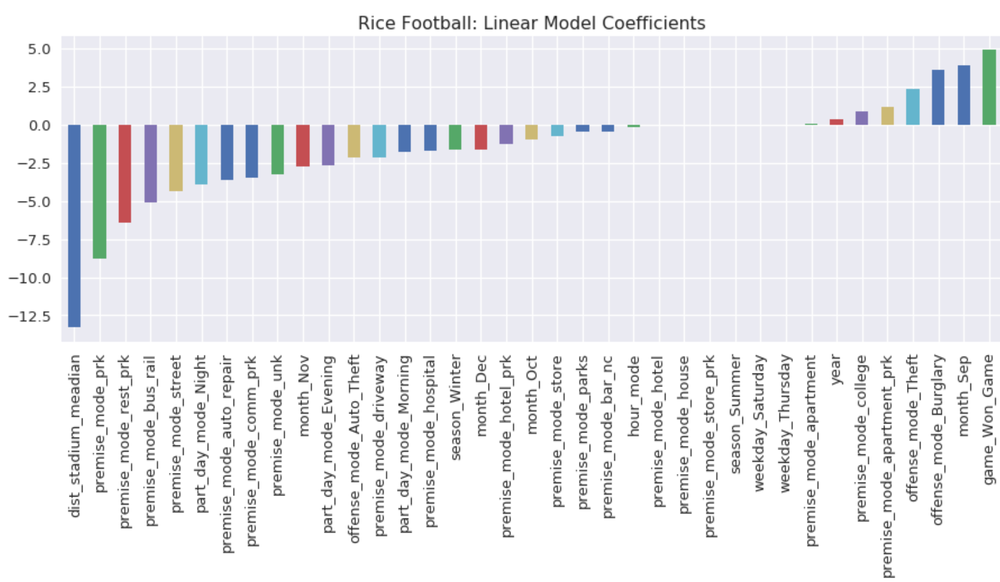
```
Model Report
RMSE : 2.782
CV Score : Mean  4.701 | Std  4.044 | Min  1.393 | Max  12.09
```



College Football: Rice University

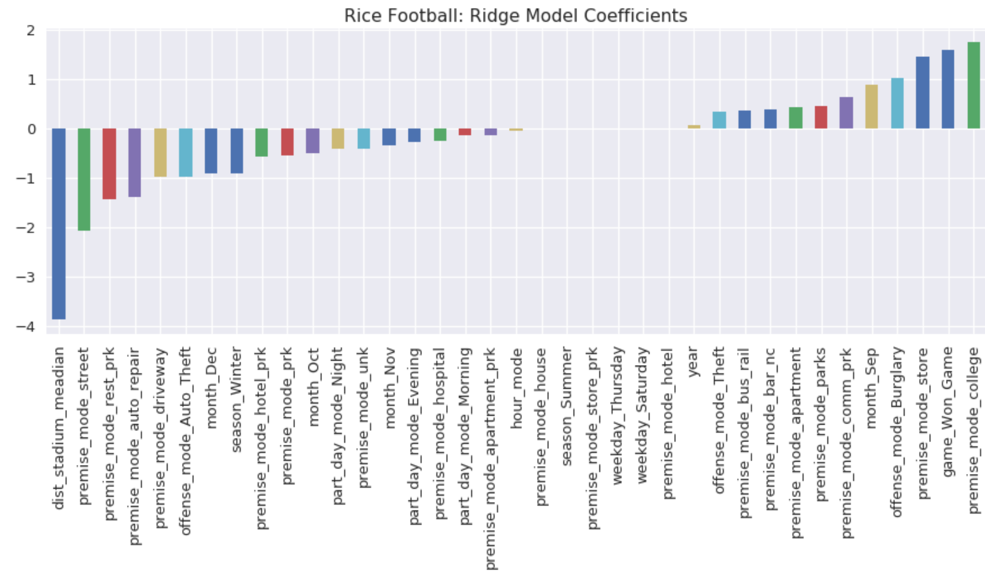
Linear

```
Model Report
RMSE : 0.0716
CV Score : Mean  2.004e+14 | Std  4.137e+14 | Min  1.32 | Max  1.32e+15
```



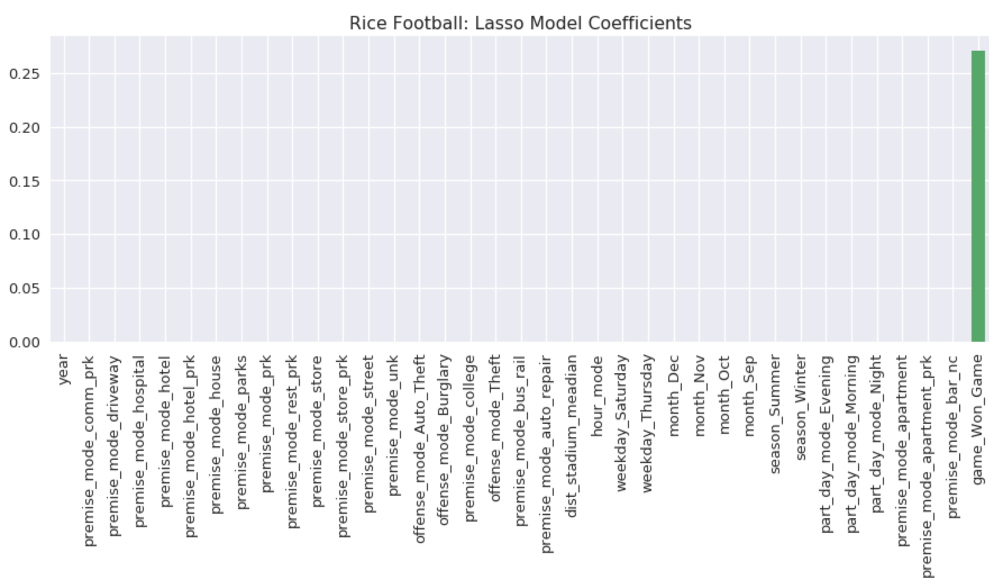
Ridge

```
Model Report
RMSE : 0.6496
CV Score : Mean  1.595 | Std  0.7066 | Min  0.2395 | Max  2.417
```



Lasso

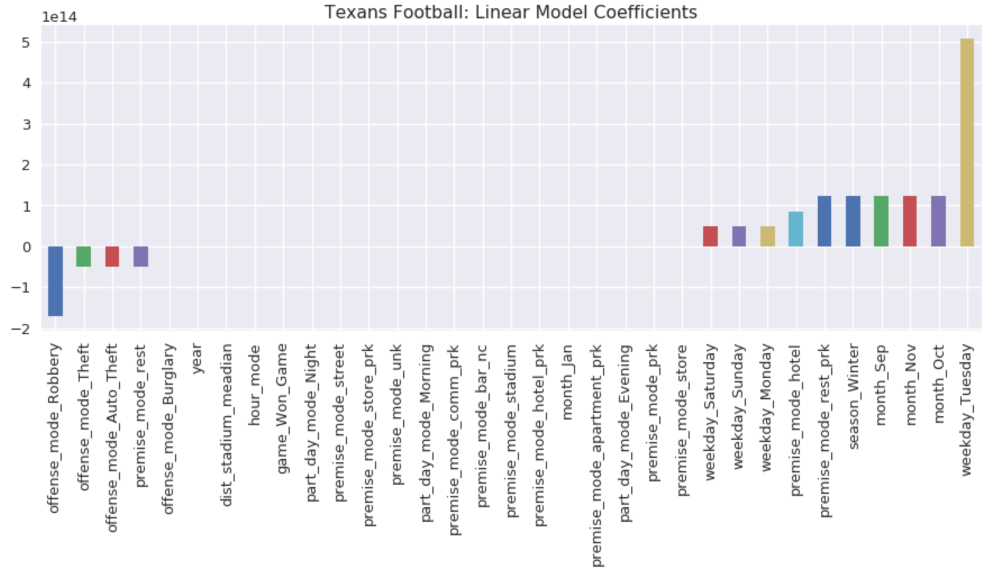
```
Model Report
RMSE : 1.357
CV Score : Mean  1.283 | Std  0.4084 | Min  0.7061 | Max  2.274
```



Football: Texans

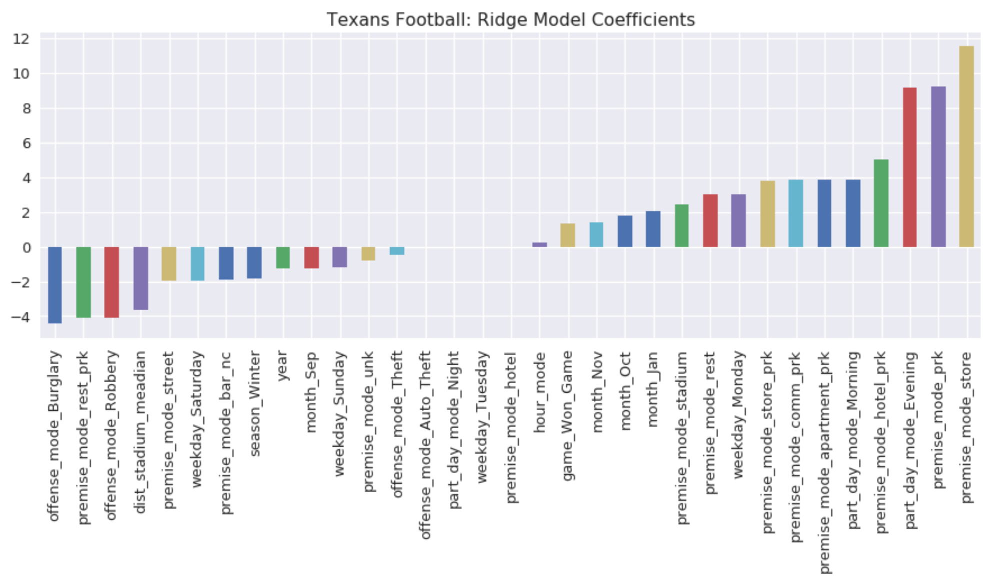
Linear

```
Model Report
RMSE : 4.733
CV Score : Mean  3.006e+13 | Std  9.019e+13 | Min  5.866 | Max  3.006e+14
```



Ridge

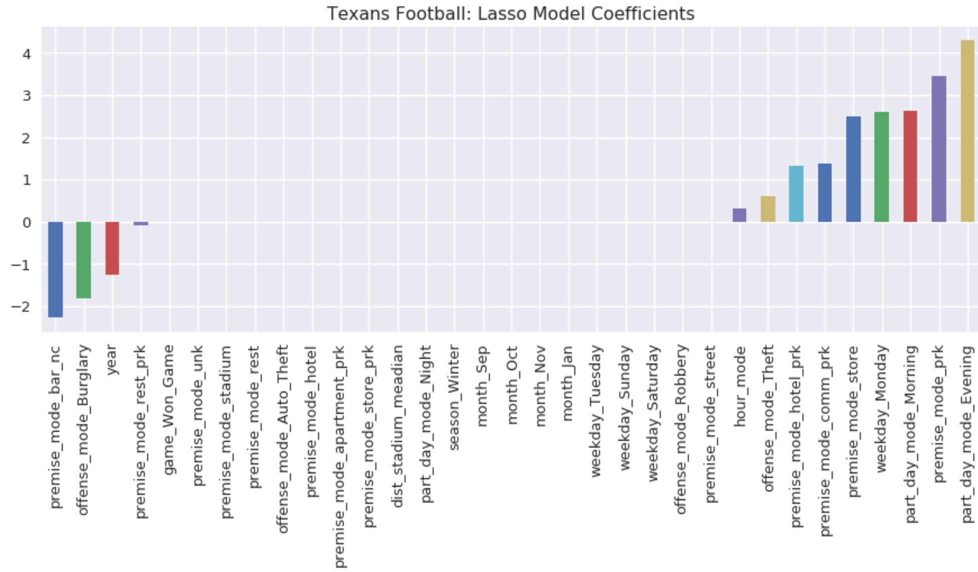
```
Model Report
RMSE : 5.195
CV Score : Mean  8.908 | Std  3.495 | Min  4.1 | Max  15.28
```



Lasso

```
Model Report
```

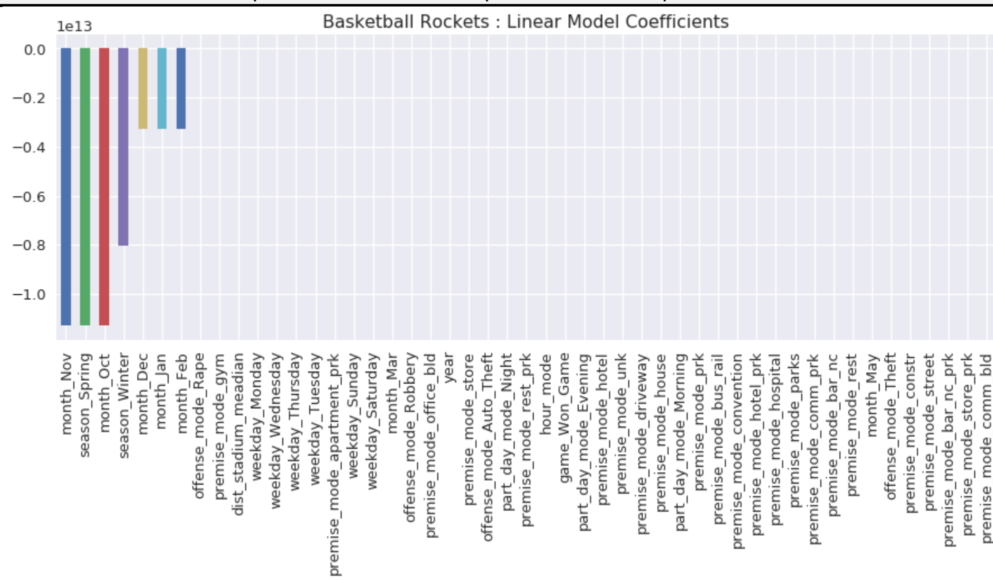
```
RMSE : 6.08
CV Score : Mean  7.667 | Std  3.242 | Min  3.67 | Max  12.71
```



Basketball: Rockets

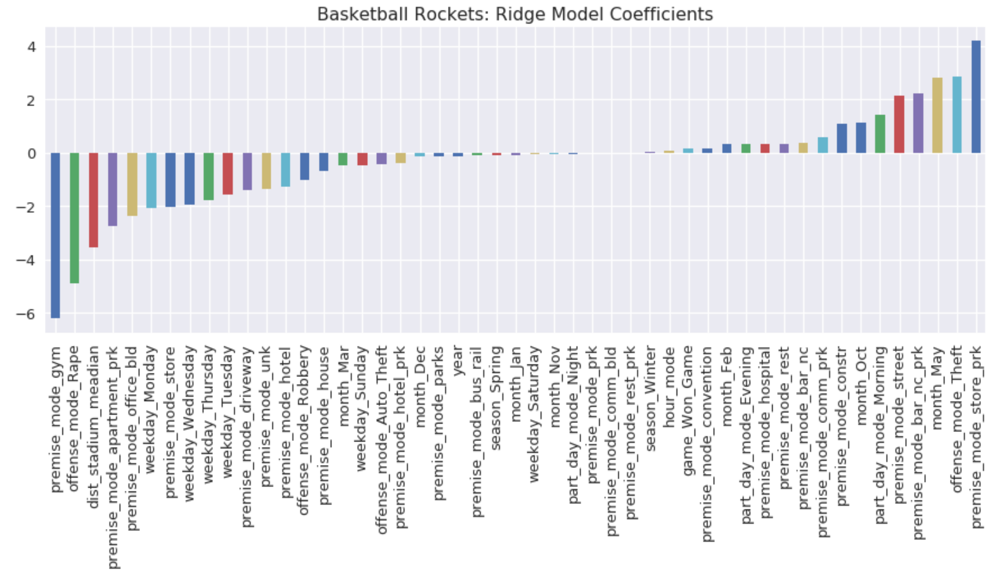
Linear

```
Model Report
RMSE : 3.04
CV Score : Mean  2.718e+06 | Std  8.084e+06 | Min  2.718 | Max  2.697e+07
```



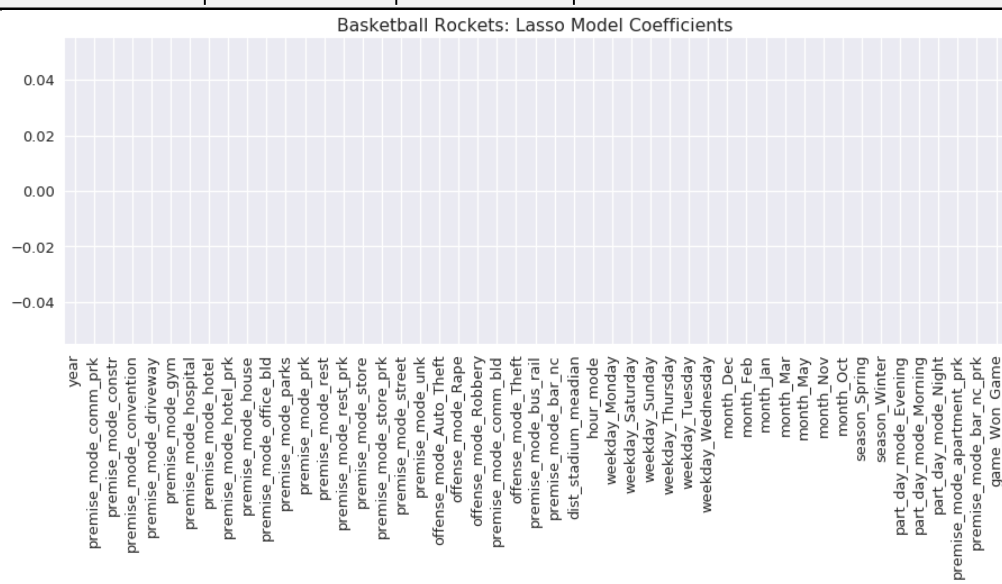
Ridge

```
Model Report
RMSE : 3.079
CV Score : Mean  3.47 | Std  0.7354 | Min  2.501 | Max  5.116
```



Lasso

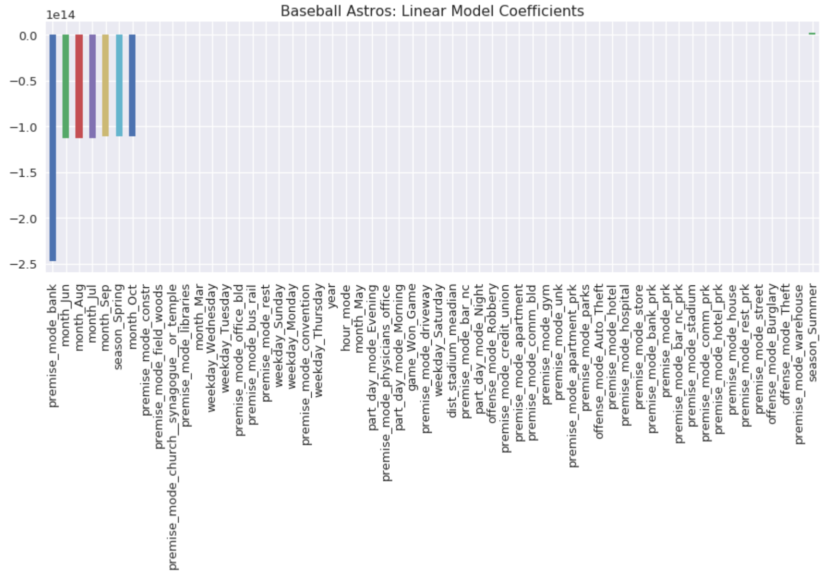
```
Model Report
RMSE : 3.907
CV Score : Mean  3.689 | Std  0.8254 | Min  2.687 | Max  5.62
```



Baseball: Astros

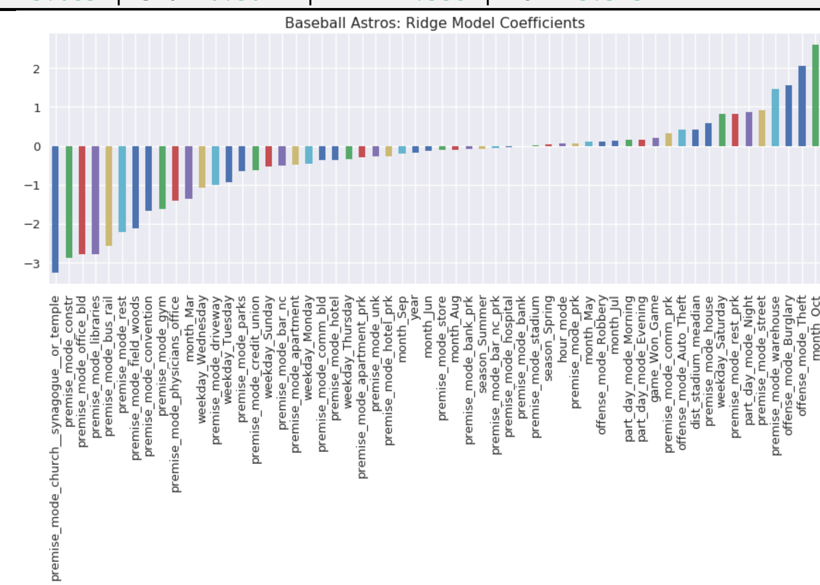
Linear

```
Model Report
RMSE : 2.811
CV Score : Mean  6.38e+13 | Std  1.081e+14 | Min  2.994 | Max  3.154e+14
```



Ridge

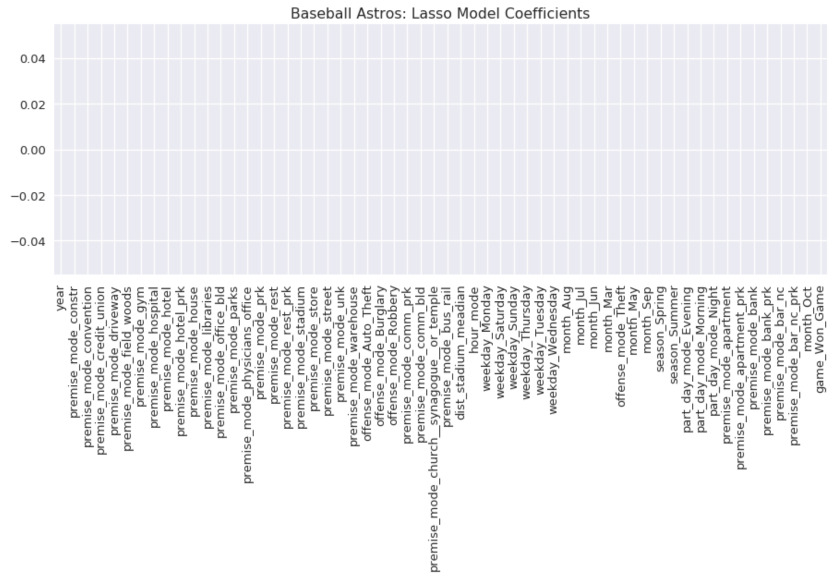
```
Model Report
RMSE : 2.818
CV Score : Mean  3.085 | Std  0.3041 | Min  2.538 | Max  3.513
```



Lasso

```
Model Report
RMSE : 3.231
```

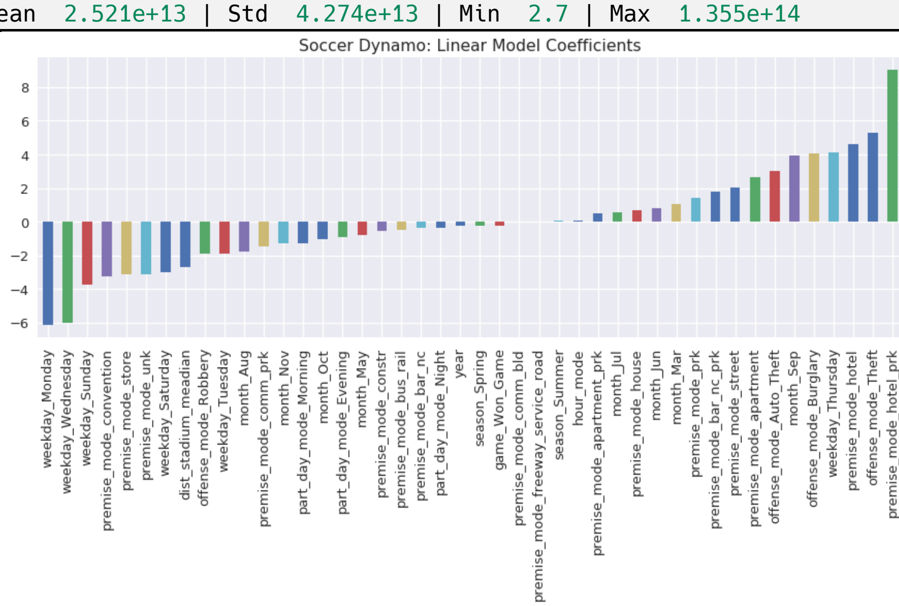
```
CV Score : Mean  3.249 | Std  0.3112 | Min  2.772 | Max  3.79
```



Soccer: Dynamo

Linear

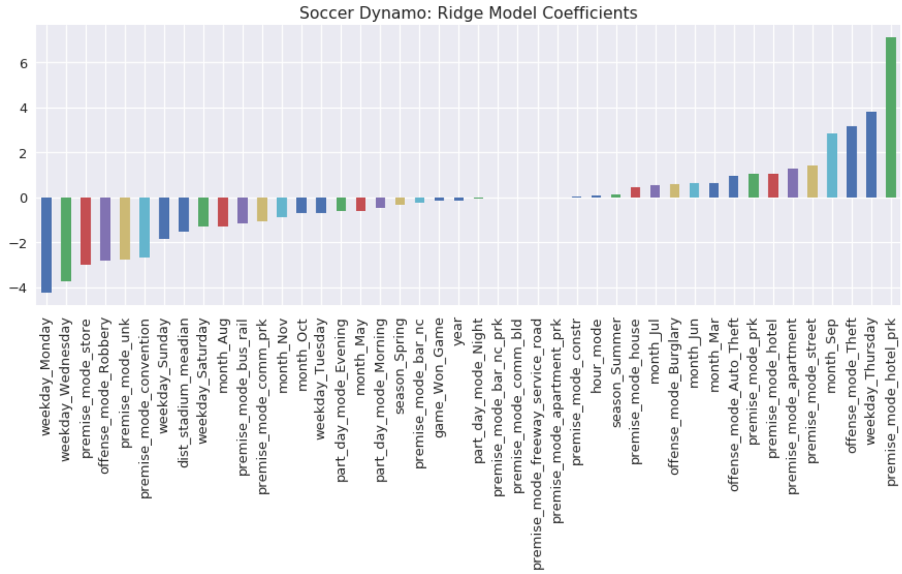
```
Model Report
RMSE : 2.478
CV Score : Mean  2.521e+13 | Std  4.274e+13 | Min  2.7 | Max  1.355e+14
```



Ridge

```
Model Report
```

```
RMSE : 2.554
CV Score : Mean  3.653 | Std  0.9231 | Min  2.467 | Max  5.433
```



Lasso

```
Model Report
RMSE : 3.515
CV Score : Mean  3.602 | Std  0.8621 | Min  2.05 | Max  5.239
```