Football Analysis

Presented by Michael Diaz

Agenda

1. Introductions
2. Project Rationale
3. Project Process
4. The Results
5. Bias, Challenges, Plans

Part 1:

Introductions

Responsibilities

- **Michael Diaz-**
  - **Gathered measurable NFL data, using:**
    - **API (MySportsFeed)**
    - **Web Scraping (ProFootballReference)**
    - **Dataframe structures with Pandas**
    - **Graph plots with Matplotlib**
    - **Make Statistical Correlations**

Part 2

Project Rationales

Objectives

Our Goals/Outcomes from the Data

- ★ Make meaningful observations and connections of statistics from all phases of Football and Winning of top and bottom 25% NFL teams.
- ➢ Target and correlate key statistics in all phases with winning
- ➢ Prove or disprove biases for meaningful statistics

Rationales

Why we used what we used?

- We looked into free and available API that showed all major NFL stats that are used for: Fantasy Sports, Sports Betting, Media, Coaching
- API dates go back to 2014 season
- Focused on only Team statistics, looking at the top and bottom 25% of teams (16) in recent history.

Introduction

Briefly explaining what/how we did this

- After looking extensively through available API and websites, we chose sources we initially thought:
  - 1.Free
  - 2. Easy to gather
- We felt we needed to combine all this data through:
  - merging and creating DataFrames
- We looked through all statistical categories and compared the results with:
  - Sixteen teams with most wins over 4 seasons.
- We put these into visuals to look for:
  - "Eye popping" correlations
  - "Eye popping" disparities

Constraints

Considerations for this project:

- $0 Budget
- 7 Days
- 2 Members

Part 3

The process

How it was made

- We created API pull call that:
  - Specifically ran all seasons, for data sets we wanted
  - Searched through Kaggle competitions that posted relevant data
- Using Python and Pandas:
  - We were able to create Data Frames that looked at all NFL scores, individual “stats” for each season, schedules, and game results
  - We then narrowed down to only looking through top and bottom 25% of NFL.
- Using Python and Matplotlib:
  - We generated scatter plots to find correlations or disparities.

Code Snippet 1 - Imports

```
#### Dependencies
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import urllib
from urllib.request import urlopen
import json
import csv
# import simplejson
import pandas as pd
# import pytest
import base64
import sys
import glob
if sys.version_info[0] < 3:
    from StringIO import StringIO
else:
    from io import StringIO
from bs4 import BeautifulSoup
import requests
import scipy.stats
import statistics as s
pd.options.display.max_rows = 999
pd.options.display.max_columns = 999
```

Code Snippet 2 - Creating DFs/.CSVs

```
#SuperBowl DataFrame
sb_dates = []
sb_number = []
sb_mvp = []
sb_winner = []
sb_winPts = []
sb_loser = []
sb_losePts = []
sb_stadium = []
sb_city = []
sb_state = []

for data in sb_json_print:
    sb_dates.append(sb_json_print["Date"])
    sb_number.append(sb_json_print["SB"])
    sb_mvp.append(sb_json_print["MVP"])
    sb_winner.append(sb_json_print["Winner"])
    sb_winPts.append(sb_json_print["Pts"])
    sb_loser.append(sb_json_print["Loser"])
    sb_losePts.append(sb_json_print["Pts.1"])
    sb_stadium.append(sb_json_print["Stadium"])
    sb_city.append(sb_json_print["City"])
    sb_state.append(sb_json_print["State"])
    SB_DF = pd.DataFrame.from_dict(sb_json_print)
SB_DF = SB_DF[["SB", "Date", "Winner", "Pts", "Loser", "Pts.1", "MVP", "Stadium", "City", "State"]]
SB_dataDF = SB_DF.rename(index=str, columns={"Pts": "Winning Score", "Pts.1": "Losing Score"})
SB_dataDF
# sb_date_pretty = sb_json_print["Date"]
# pprint(sb_date_pretty)
```

```
NFL_data = "box_scores.csv"
NFL_data = pd.read_csv(NFL_data)
NFL_data.head(5)
```

Code Snippet 3 - Colors

```
#Top 5 Teams Bar Chart
overall = "NFL_Standings/Overall/Overall_Standings_Combined.csv"
overall = pd.read_csv(overall)
overall.drop(["Unnamed: 0", "Unnamed: 0.1"], axis=1, inplace=True)
overall.set_index("Season")
overall.head(8)
new_england = mpatches.Patch(color='navy', label='Tier 1')
pit_steelers = mpatches.Patch(color='orange', label="Def Tier 1")
sea_seahawks = mpatches.Patch(color='navy')
gb_packers = mpatches.Patch(color='navy')
kc_chiefs = mpatches.Patch(color='navy')
car_panthers = mpatches.Patch(color='navy')
den_broncos = mpatches.Patch(color='navy')
ari_cardinals = mpatches.Patch(color='navy')
colors = ["navy", "navy", "navy", "navy", "navy", "navy", "navy", "navy"]
cle_browns = mpatches.Patch(color='crimson', label="Tier 2")
stl_rams = mpatches.Patch(color='green', label= "Def Tier 2")
sd_chargers = mpatches.Patch(color='crimson')
chi_bears = mpatches.Patch(color='crimson')
sf_49ers = mpatches.Patch(color='crimson')
tb_bucs = mpatches.Patch(color='crimson')
jax_jags = mpatches.Patch(color='crimson')
ny_jets = mpatches.Patch(color='crimson')
colors2 = ["crimson", "crimson", "crimson", "crimson", "crimson", "crimson", "crimson", "crimson"]
handles = [new_england, cle_browns, pit_steelers, stl_rams]
```

Code Snippet 4 - Beautiful Soup

```
#Web Scrapping
quote_page = "https://www.pro-football-reference.com/super-bowl/"

#Query the website to get HTML page of the url
html_page = urlopen(quote_page)
html_page
#Parse the html using beautiful soup and store in a variable
sb_parser = BeautifulSoup(html_page, "html.parser")
superbowl_id = sb_parser.find(id = "all_super_bowls")
super_bowl_scores = superbowl_id.text.strip() # strip() is used to remove starting and trailing
# super_bowl_scores.to_json("Super_Bowl_Scores")
super_bowl_dumps = json.dumps(super_bowl_scores)
super_bowl_json = json.loads(super_bowl_dumps)
super_bowl_json.replace("\n", " ")
# super_bowl_json.to_json("Super_Bowl_Scores")
# print(super_bowl_json)
sb_DF = StringIO(super_bowl_json)
super_bowl_DF = sb_DF.writable()
#Using Pandas to webscrape
html_df, = pd.read_html(quote_page)
#Store as JSON and CSV
superbowl_DF = html_df.to_csv("Super_Bowl_Scores.csv", sep=",")
```

Code Snippet 5 - Scatter Plot Base

In [102]:

```
yplot_ST = np.array([st_list[61]])
xplot_ST = np.array([x1["#Wins"]])
plt.scatter(x=xplot_ST,
            y=yplot_ST,
            s=15,
            facecolors=colors,
            alpha=0.8,
            linewidth=1)
yplot_ST2 = np.array([st_list[68]])
xplot_ST2 = np.array([x1["#Wins"]])
plt.scatter(x=xplot_ST2,
            y=yplot_ST2,
            s=15,
            facecolors="green",
            alpha=0.8,
            linewidth=1)
yplot_ST3 = np.array([st_list2[61]])
xplot_ST3 = np.array([x2["#Wins"]])
plt.scatter(x=xplot_ST3,
            y=yplot_ST3,
            s=15,
            facecolors=colors2,
            alpha=0.8,
            linewidth=1)
yplot_ST4 = np.array([st_list2[68]])
xplot_ST4 = np.array([x2["#Wins"]])
plt.scatter(x=xplot_ST4,
            y=yplot_ST4,
            s=15,
            facecolors="orange",
            alpha=0.8,
            linewidth=1)
plt.grid(True)
plt.legend(handles=handles)
plt.title("Overall Wins vs Special Teams Stats", fontsize=20)
plt.ylabel("Wins", fontsize=20)
plt.ylim(0,320)
plt.xlabel("Total Special Teams Stats", fontsize=20)
plt.xlim(0,70)
plt.figure(figsize=(10,10), dpi=80)
plt.show()
print(st_list[61], st_list[68], st_list2[61], st_list2[68])
```

Part 4

The results

Quick Grouped Statistics

Offense

#Team Abbr.	NE	PIT	SEA	GB	KC	CAR	DEN	ARI
count	41.000000	41.000000	41.000000	41.000000	41.000000	41.000000	41.000000	41.000000
mean	2347.768293	2189.926829	1991.973171	2009.429268	1793.236585	1886.109756	1869.187805	1858.563415
std	5693.566161	5297.601887	4612.385070	4702.314289	4174.589397	4320.465804	4472.418847	4549.019682
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	55.100000	66.000000	48.000000	48.800000	44.000000	52.000000	38.700000	39.300000
50%	279.000000	264.000000	236.000000	237.000000	199.000000	218.800000	214.000000	216.000000
75%	1112.000000	1210.000000	1329.000000	1274.000000	992.000000	1236.000000	1235.000000	1119.000000
max	22106.000000	20528.000000	17755.000000	18285.000000	16050.000000	16563.000000	17404.000000	17743.000000

#Team Abbr.	CLE	STL	SD	CHI	SF	TB	JAX	NYJ
count	41.000000	41.000000	41.000000	41.000000	41.000000	41.000000	41.000000	41.000000
mean	1562.765854	756.075610	1367.082927	1618.829268	1585.824390	1720.356098	1746.790244	1593.821951
std	3861.298891	1751.151426	3477.517202	3952.057889	3784.419643	4255.748380	4174.864804	3817.423672
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	37.000000	22.300000	33.000000	34.000000	37.000000	36.000000	37.000000	41.000000
50%	199.000000	70.000000	155.000000	166.000000	181.000000	185.000000	200.000000	176.000000
75%	856.800000	452.000000	696.000000	828.000000	859.000000	961.300000	1088.200000	855.600000
max	15096.000000	6687.000000	13582.000000	15203.000000	14588.000000	16554.000000	16188.000000	14584.000000

Quick Grouped Statistics

Defense

#Team Abbr.	NE	PIT	SEA	GB	KC	CAR	DEN	ARI
count	23.000000	23.000000	23.000000	23.000000	23.000000	23.000000	23.000000	23.000000
mean	722.613043	697.513043	670.378261	705.147826	614.395652	685.530435	645.069565	660.213043
std	1198.012304	1137.736804	1130.657954	1155.399173	1049.024556	1120.395108	1071.982531	1074.477014
min	-70.000000	-91.000000	-278.000000	-43.000000	-196.000000	-278.000000	-121.000000	-205.000000
25%	39.000000	38.000000	41.000000	38.000000	37.000000	45.100000	41.500000	44.000000
50%	88.100000	89.800000	83.000000	96.400000	84.100000	87.000000	85.000000	125.900000
75%	995.500000	993.500000	867.000000	936.000000	752.500000	1017.000000	767.000000	767.000000
max	4671.000000	4296.000000	4418.000000	4416.000000	3938.000000	4280.000000	4027.000000	3917.000000

#Team Abbr.	CLE	STL	SD	CHI	SF	TB	JAX	NYJ
count	23.000000	23.000000	23.000000	23.000000	23.000000	23.000000	23.000000	23.000000
mean	608.500000	316.386957	442.943478	554.560870	594.121739	597.726087	631.986957	600.882609
std	1063.633492	540.805509	779.170573	998.400247	1053.625160	1078.579832	1076.496765	1029.010919
min	-81.000000	-62.000000	-16.000000	-158.000000	-70.000000	-82.000000	0.000000	-169.000000
25%	37.250000	20.500000	27.350000	36.950000	36.000000	43.000000	39.500000	37.000000
50%	92.000000	46.000000	50.000000	77.000000	78.000000	89.000000	90.000000	79.000000
75%	671.500000	374.500000	502.500000	589.500000	687.500000	640.000000	727.000000	723.000000
max	4244.000000	2077.000000	2987.000000	3830.000000	4094.000000	4228.000000	4111.000000	3968.000000

Quick Grouped Statistics

Special Teams

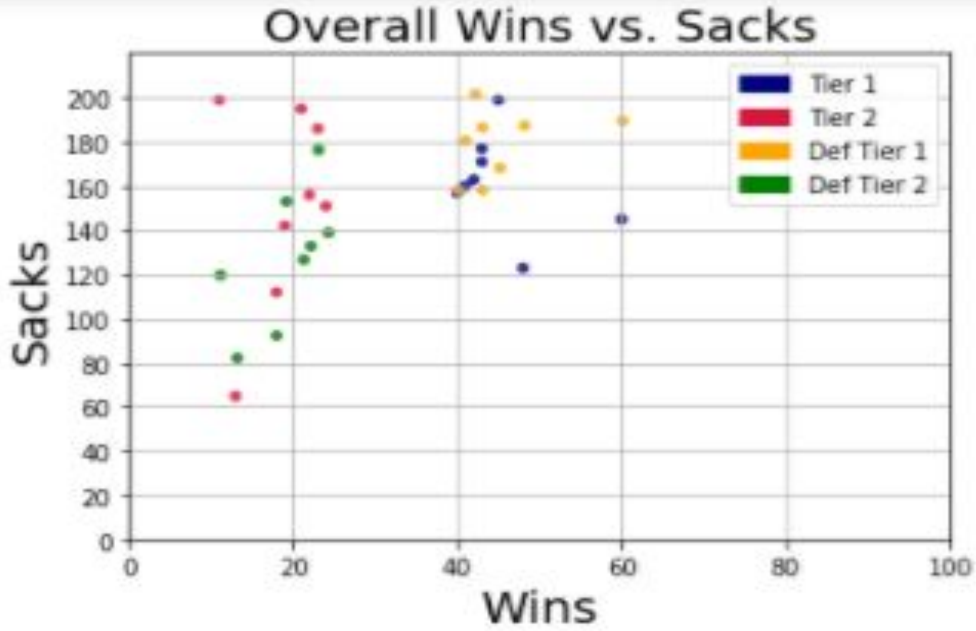
	ARI	CAR	CHI	CLE	DEN
count	69.000000	69.000000	2.000000	2.000000	69.000000
mean	939.942029	990.395652	127.000000	126.500000	940.886957
std	3333.455957	3610.412249	179.605122	178.898016	3392.698579
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	7.000000	7.000000	63.500000	63.250000	9.000000
50%	107.000000	100.000000	127.000000	126.500000	100.000000
75%	307.000000	311.300000	190.500000	189.750000	309.000000
max	21550.000000	24717.000000	254.000000	253.000000	21614.000000

	GB	JAX	KC	NE	NYJ
count	69.000000	2.000000	69.00000	69.000000	2.000000
mean	932.175362	155.000000	993.15942	1073.113043	143.500000
std	3330.263002	219.203102	3450.48370	3966.585226	202.939646
min	0.000000	0.000000	0.00000	0.000000	0.000000
25%	9.000000	77.500000	11.00000	8.000000	71.750000
50%	91.000000	155.000000	107.00000	102.000000	143.500000
75%	320.400000	232.500000	313.70000	355.900000	215.250000
max	23579.000000	310.000000	23218.00000	28633.000000	287.000000

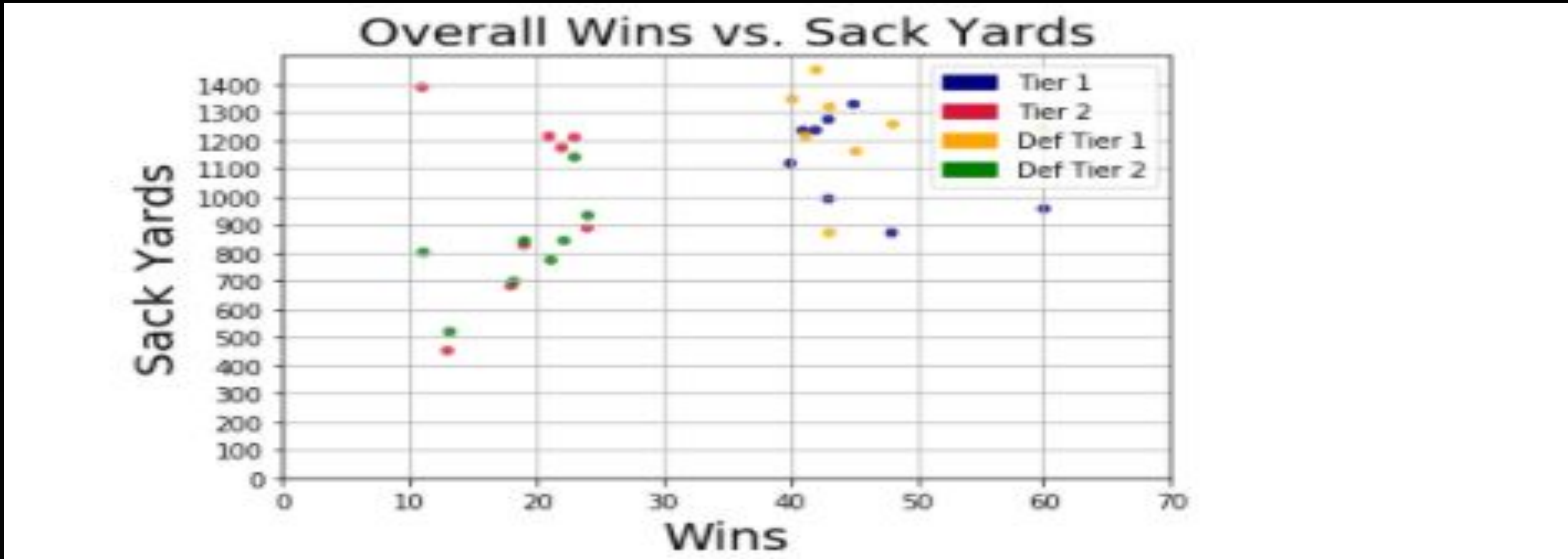
	PIT	SD	SEA	SF	STL
count	69.000000	2.000000	69.000000	2.000000	2.000000
mean	922.315942	117.000000	999.646377	136.000000	68.500000
std	3329.829143	165.462987	3502.004570	192.333044	96.873629
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	4.000000	58.500000	7.000000	68.000000	34.250000
50%	98.000000	117.000000	110.000000	136.000000	68.500000
75%	299.200000	175.500000	304.800000	204.000000	102.750000
max	23997.000000	234.000000	23809.000000	272.000000	137.000000

	TB
count	2.000000
mean	146.500000
std	207.182287
min	0.000000
25%	73.250000
50%	146.500000
75%	219.750000
max	293.000000

Positive Correlations & Linear Stagnation

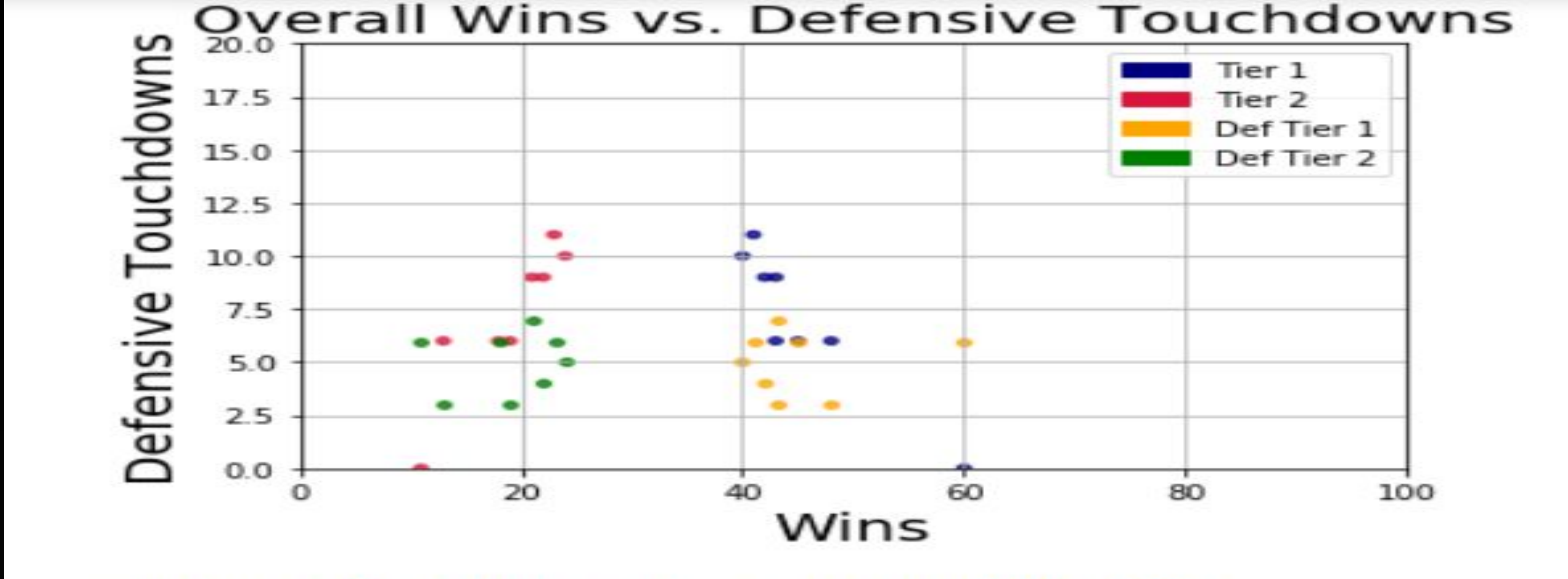



```
<matplotlib.figure.Figure at 0x2c7563de518>

#Team Abbr.
NE      145
PIT     123
SEA     199
GB      177
KC      171
CAR     163
DEN     160
ARI     157
Name: #PassSacks, dtype: int64 #Team Abbr.
CLE     199
STL      65
SD      112
CHI     142
SF      195
TB      156
JAX     186
NYJ     151
```

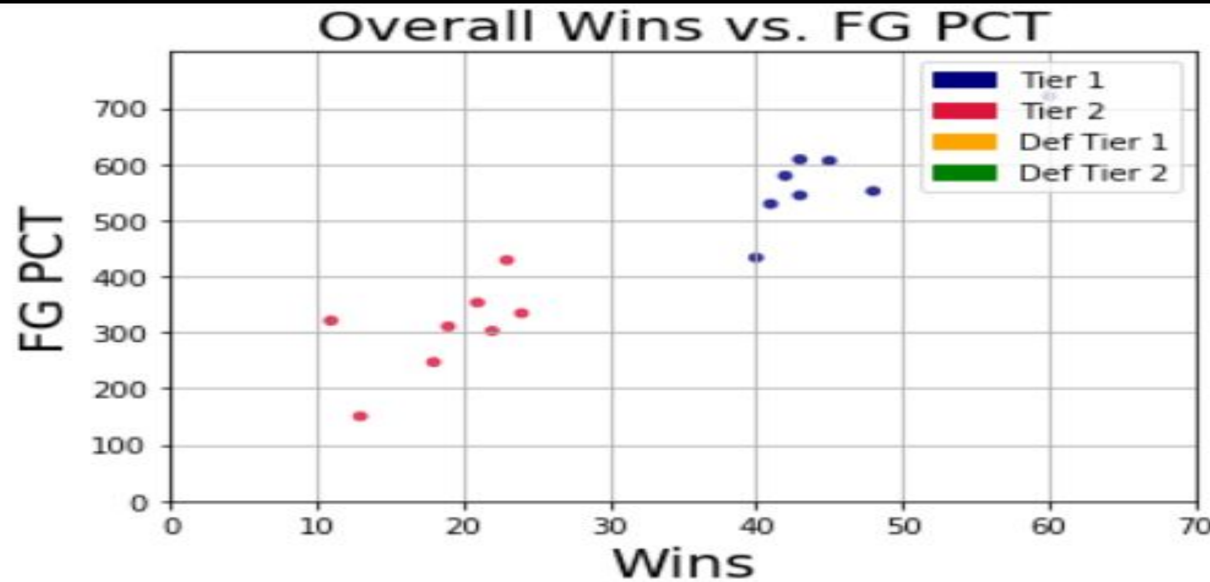



```
#Team Abbr.
NE        957
PIT       871
SEA      1329
GB       1274
KC        992
CAR      1236
DEN      1235
ARI      1119
Name: #PassSackY, dtype: int64 #Team Abbr.
CLE      1389
STL       452
SD        682
CHI       828
SF       1215
TB       1173
JAX      1211
NYJ       888
Name: #PassSackY, dtype: int64 #Team Abbr.
NE       1249
PIT      1261
SEA      1162
GB       1325
KC        873
CAR      1457
DEN      1223
ARI      1349
Name: #SackYds, dtype: int64 #Team Abbr.
NE       1249
PIT      1261
SEA      1162
GB       1325
KC        873
CAR      1457
DEN      1223
ARI      1349
Name: #SackYds, dtype: int64
```

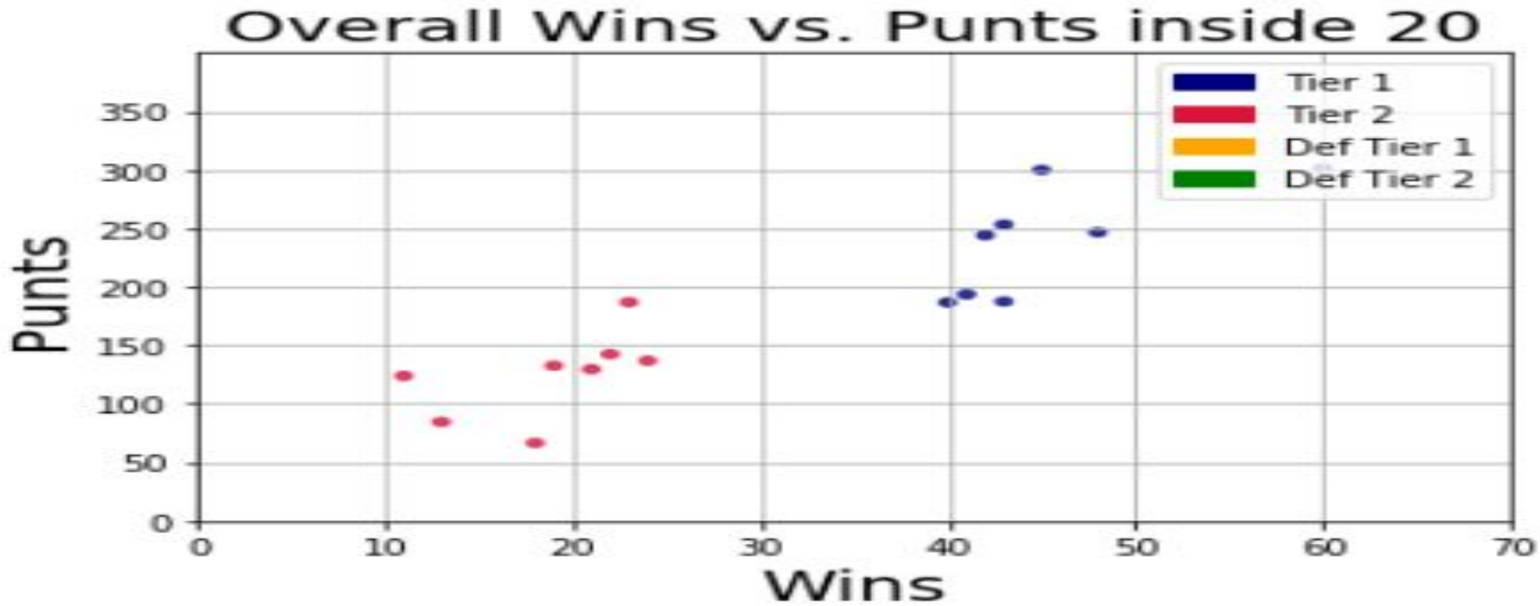

```
<matplotlib.figure.Figure at 0x2c756a00c18>

#Team Abbr.
NE      98.7250
PIT     82.4000
SEA     83.4750
GB      82.3000
KC      81.5875
CAR     77.4125
DEN     60.5250
ARI     55.9500
Name: #QBRating, dtype: float64 #Team Abbr.
CLE     36.8375
STL     20.0250
SD      34.5500
CHI     42.7375
SF      41.5750
TB      42.2750
JAX     51.8750
NYJ     39.7125
Name: #QBRating, dtype: float64
```



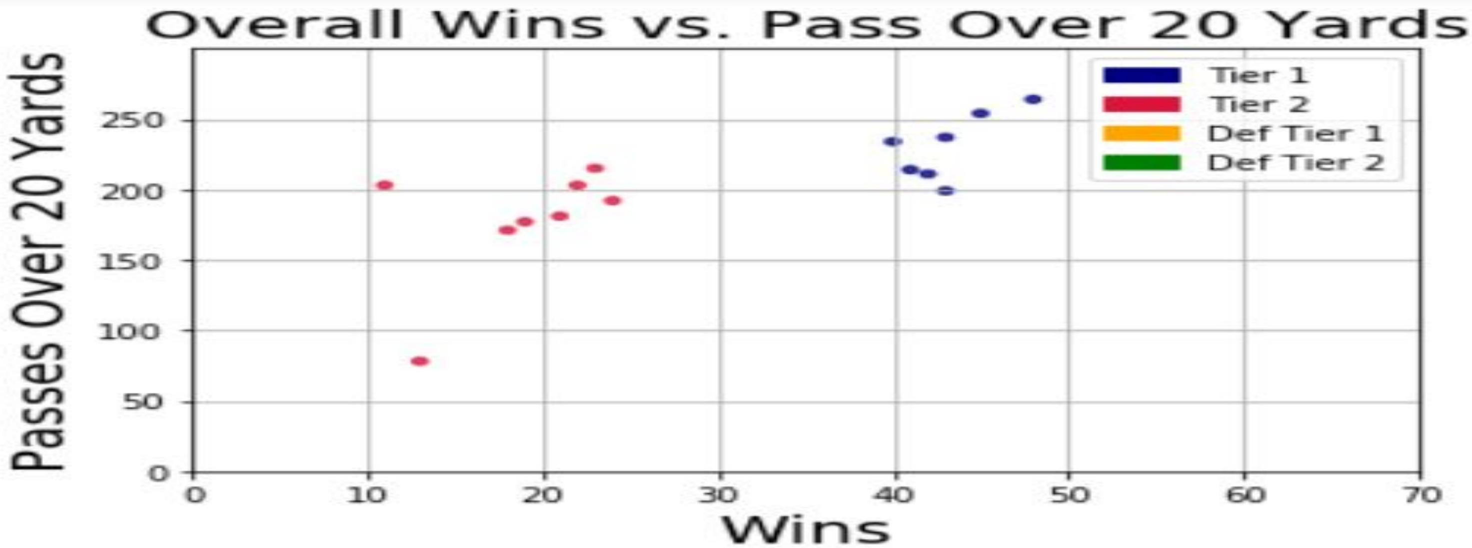

```
#Team Abbr.
NE      0
PIT     6
SEA     6
GB      6
KC      9
CAR     9
DEN    11
ARI    10
Name: #IntTD, dtype: int64 #Team Abbr.
CLE     6
STL     4
SD      4
CHI     4
SF      4
TB     10
JAX     5
NYJ     0
Name: #IntTD, dtype: int64 #Team Abbr.
NE      6
PIT     3
SEA     6
GB      3
KC      7
CAR     4
DEN     6
ARI     5
Name: #FumTD, dtype: int64 #Team Abbr.
CLE      1
STL      2
SD       5
CHI      2
SF       1
TB       5
JAX     10
NYJ      1
Name: #FumTD, dtype: int64
```



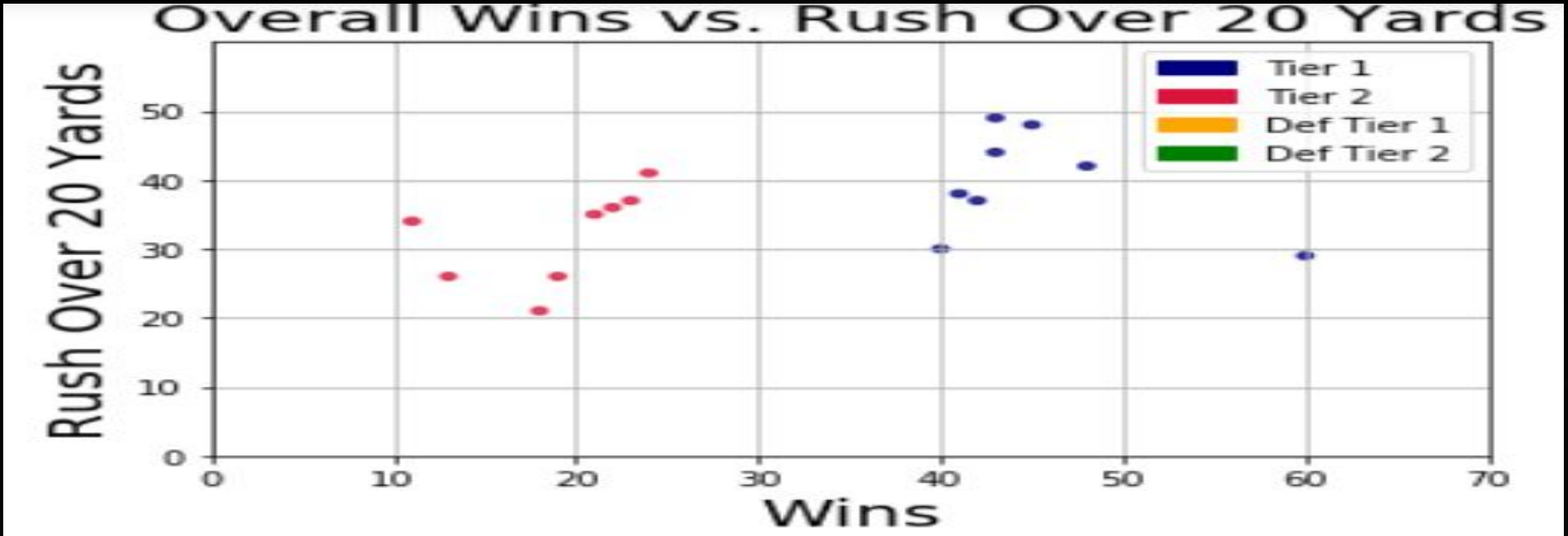

```
<matplotlib.figure.Figure at 0x1e86fb0dd30>

#Team Abbr.
NE      720.7
PIT     551.7
SEA     605.7
GB      608.2
KC      544.1
CAR     579.0
DEN     528.8
ARI     433.2
Name: #FgPct, dtype: float64 #Team Abbr.
CLE     320.9
STL     150.5
SD      247.0
CHI     310.6
SF      353.0
TB      302.6
JAX     428.7
NYJ     334.2
Name: #FgPct, dtype: float64
```



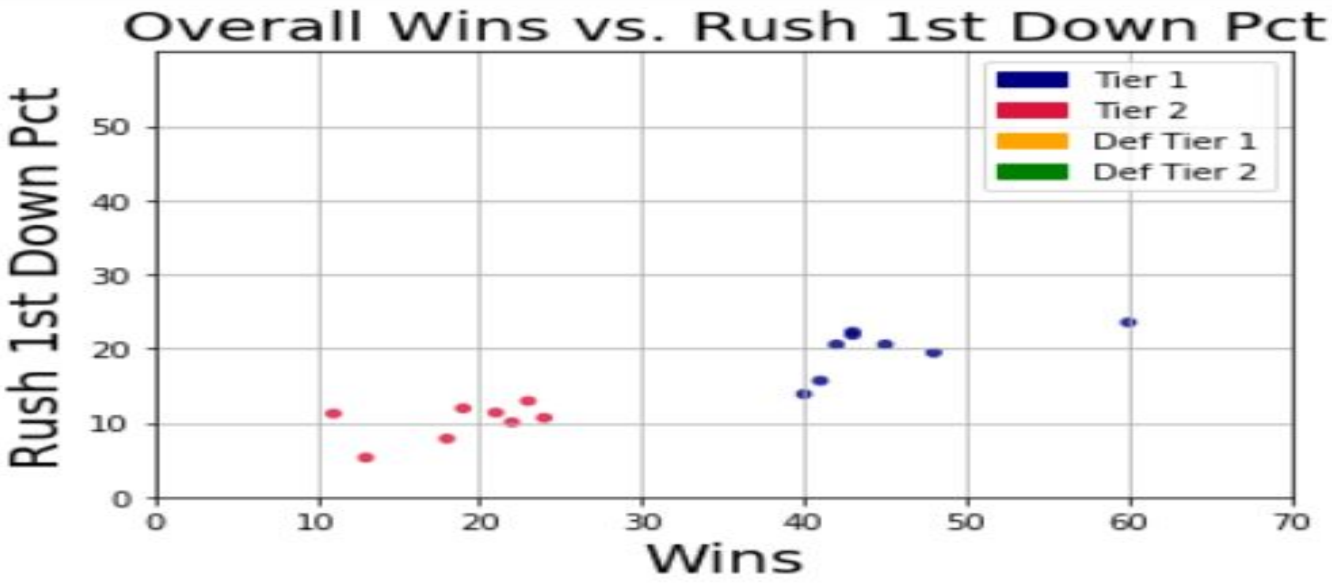

```
<matplotlib.figure.Figure at 0x1cc9b8db470>

#Team Abbr.
NE      301.2
PIT     246.6
SEA     299.9
GB      187.4
KC      253.3
CAR     244.1
DEN     193.5
ARI     186.7
Name: #PuntIn20Pct, dtype: float64 #Team Abbr.
CLE     123.8
STL      84.5
SD       66.5
CHI     132.5
SF      129.6
TB      142.4
JAX     186.8
NYJ     136.8
Name: #PuntIn20Pct, dtype: float64
```



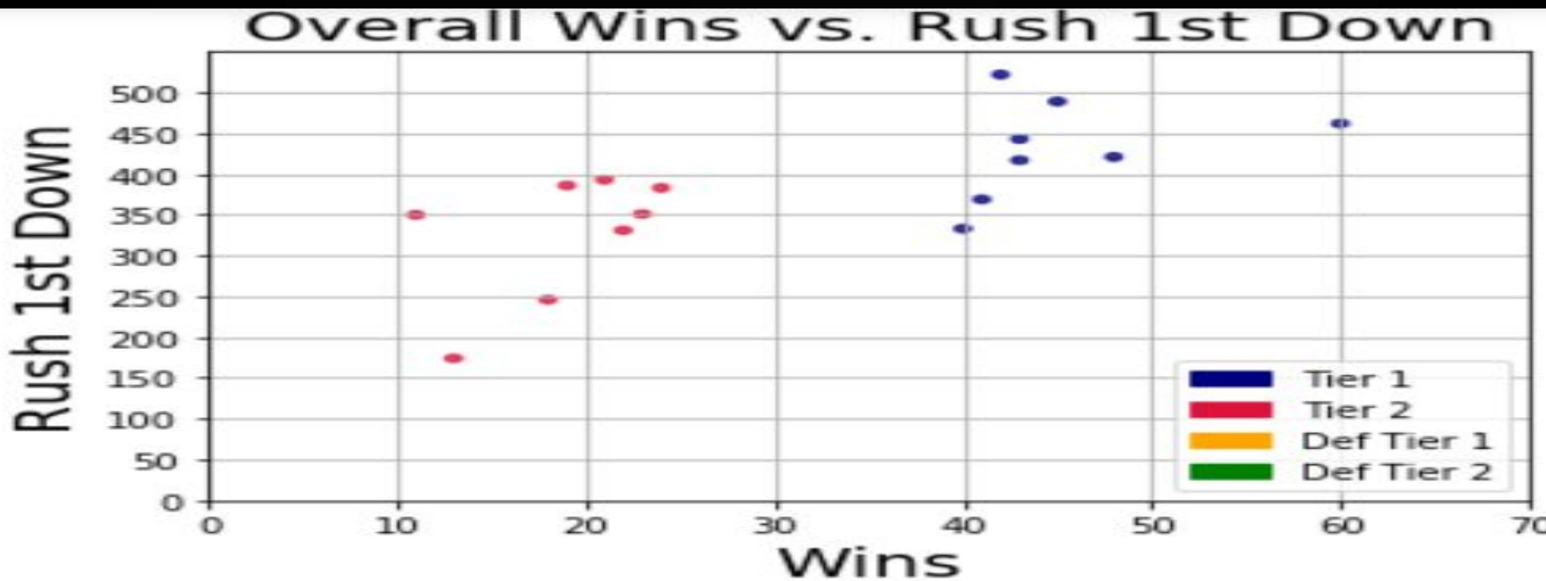

```
<matplotlib.figure.Figure at 0x1e87034bac8>

#Team Abbr.
NE       279
PIT      264
SEA      254
GB       237
KC       199
CAR      211
DEN      214
ARI      234
Name: #Pass20Plus, dtype: int64 #Team Abbr.
CLE      203
STL       78
SD       171
CHI      177
SF       181
TB       203
JAX      215
NYJ      192
Name: #Pass20Plus, dtype: int64
```



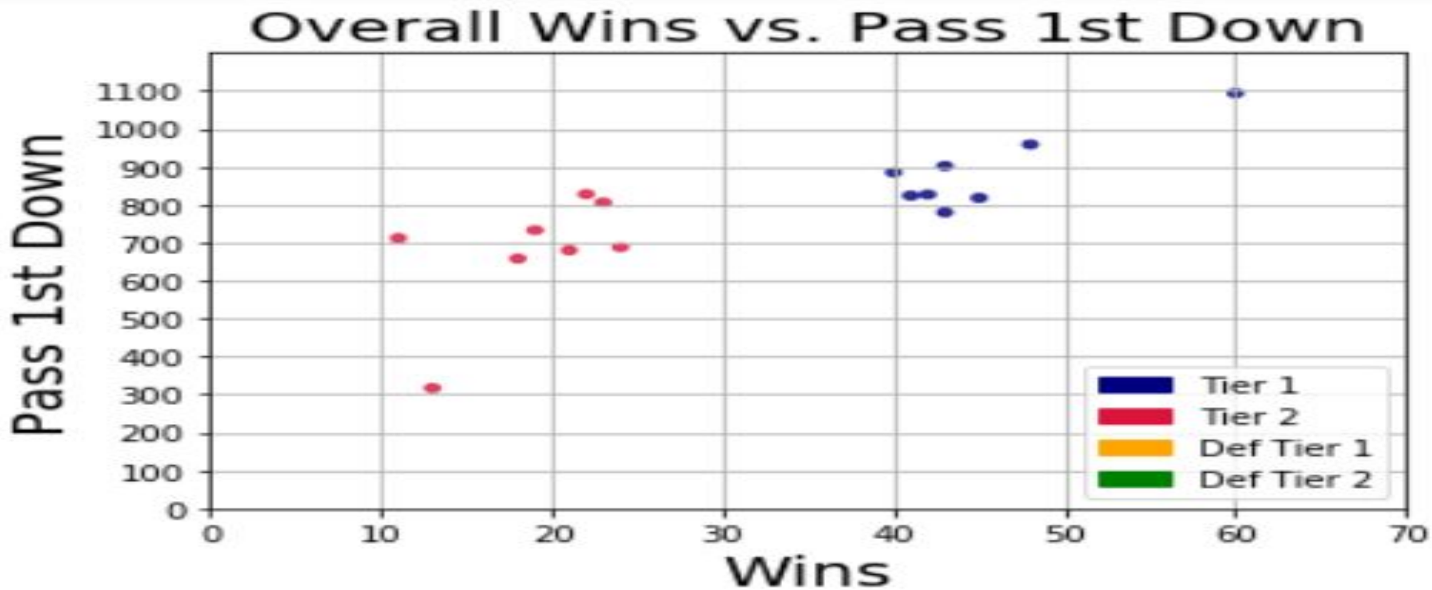

```
<matplotlib.figure.Figure at 0x1e86fcda940>

#Team Abbr.
NE      29
PIT     42
SEA     48
GB      49
KC      44
CAR     37
DEN     38
ARI     30
Name: #Rush20Plus, dtype: int64 #Team Abbr.
CLE     34
STL     26
SD      21
CHI     26
SF      35
TB      36
JAX     37
NYJ     41
Name: #Rush20Plus, dtype: int64
```




```
<matplotlib.figure.Figure at 0x1e86fcd4240>

#Team Abbr.
NE      23.5125
PIT     19.4375
SEA     20.5375
GB      22.2500
KC      21.8250
CAR     20.5250
DEN     15.6500
ARI     13.8500
Name: #Rush1stDownsPct, dtype: float64 #Team Abbr.
CLE     11.2125
STL      5.2750
SD       7.8375
CHI     11.9250
SF      11.3500
TB      10.0375
JAX     12.9125
NYJ     10.6500
Name: #Rush1stDownsPct, dtype: float64
```




```
<matplotlib.figure.Figure at 0x1e86b7629b0>

#Team Abbr.
NE      462
PIT     421
SEA     489
GB      443
KC      417
CAR     522
DEN     369
ARI     333
Name: #Rush1stDowns, dtype: int64 #Team Abbr.
CLE     350
STL     174
SD      246
CHI     386
SF      393
TB      331
JAX     351
NYJ     383
Name: #Rush1stDowns, dtype: int64
```




```
<matplotlib.figure.Figure at 0x1e86b7a50b8>

#Team Abbr.
NE      1094
PIT      959
SEA      818
GB       902
KC       780
CAR      827
DEN      824
ARI      885
Name: #Rec1stDowns, dtype: int64 #Team Abbr.
CLE     712
STL     317
SD      658
CHI     733
SF      680
TB      828
JAX     806
NYJ     688
Name: #Rec1stDowns, dtype: int64
```

Part 5

Bias, Challenges, Plans

Data Bias

Due to constraints stated above, we:

- Did not consider other major factors that can influence the outcome of a game in the “stats” we used.
- We only made a statistical model based on four cumulative season and playoff totals.
- Correlations between prior success and statistics does not always represent or determine future success.

Challenges to Overcome

- API was extremely time consuming to properly pull.
- API had severe restrictions on number of pulls
- Data available with relation to deadline of assignment
- Assignment due date was pushed forward by 4 days, while well into assignment
- Questions to guide our project were constantly changing as they needed to become more narrow to focus data
- Objective/Goal for project needed to chang
- Duties had to be constantly shifted as new r arrived while moving forward with project

Plans for Improvement

- Data will include evaluation and correlation between each year
- Individual “stats” will be included as factors of team success
- Standard Deviations, Variance, and P-Values will be put in place to show level of accuracy
- Regression models and Machine Learning to create predictions for future success, rather than just connections to past success.
- Potentially ask user input for “stats” they wish to know and output charts with rating and correlation

Credits

- Data made available from:
  - Mysportsfeed
  - ProFootballReference
  - Kaggle
- Logo property of:
  - NFL (National Football League)