IST718 - Lab Exercise 1 - Predict College Footbal Coach Salary

John Fields

18 April 2020

Summary

Model Summary Table

Model	Variables	Adj. R-squared	Range of p-values	Method
Without Conference	Capacity,YearsCurrentSchool,RankedPost,Latitude	.672	.000 - .082	Traditional
Without Conference	Capacity,YearsCurrentSchool,RankedPost,Latitude	n/a	n/a	Bayesian
With Sub-Conference	Sub-Conference,Capacity,YearsCurrentSchool,RankedPost,Latitude	.723	.000 - .965	Traditional
With Conference	Conference,Capacity,YearsCurrentSchool,RankedPost,Latitude	.780	.000 - .732	Traditional

Goals

1. Provide a recommendation for the salary of the next Syracuse University football coach. The salary was $2.4 million in 2016.

- $2,565,451 using the model without conference (recommendation)
- $3,059,430 with sub-conference (not recommended due to high p-values)
- Predicted Increase in Salary if Team Ranked at End of Year - $582 thousand
- See Predictions Section for details.

2. Predict the salary if Syracuse was still in the Big East (Now American - East).

- $1,238,314 with sub-conference model.
- See Predictions Section for details.

3. Predict the salary if Syracuse moved to the Big Ten.

- $3,268,349 with sub-conference model.
- See Predictions Section for details.

4. Summarize what schools and data are included as part of the analysis.

- Total Schools Evaluated 119 (of 129) from NCAA Division I Football
- Schools removed due to missing coach salary data
  - Baylor
  - Brigham Young
  - Rice
  - Southern Methodist
- Schools removed due to missing graduation data
  - Army
  - LSU
  - Navy
  - Buffalo
- Schools removed due to missing coach tenure
  - Connecticut
- Schools removed due to missing stadium data
  - Massachusetts
- Summary Statistics
  - Minimum Total Compensation = $390 thousand
  - Maximum Total Compensation = $8.8 million
  - Mean Total Compensation = $2.6 million
  - Median Total Compensation = $2.0 million
  - Mean Total Compensation Atlantic = $3.6 million
  - Median Total Compensation Atlantic = $3.2 million
- Removing outliers was evaluated and the decision was made to include all coaches Total Compensation in the model.

5. Describe the effect of graduation success rates (GSR) on the projected salary.

- There was not a linear relationship between GSR and salary as shown in the plot in the Summary Statistics Section.
- The GSR also had a high p-value of 0.62 on test data which indicates it is not statistically significant so it was removed from the regression equations.
- There does appear to be a correlation between high GSR and private schools in the ACC as shown in the grid plot in the Summary Statistics Section.

6. Explain the single biggest variable used to predict salary.

- The single biggest variable that predicts salary in all regression models is stadium capacity.
- It should also be noted that conference was included in the sub-conference regression equation to answer Questions #2 and #3 above. However, since many variables in the complex model have high p-values, it is not recommended to use this model.
- Regression Analysis Section for details.

Bonus 1: Develop a geographic visualization of salary by conference.

- See the Geographic Analysis section.

Assumptions

To calculate an appropriate salary for the next Syracuse Football coach, total compensation is used instead of only Total Pay or School Pay. The reason for this decision is that some schools pay bonuses and some do not. So, by looking at Total Pay + Bonus Payout, there is a more realistic picture of the amount that each coach made annually.

Terms

- Total Pay = Sum of School Pay and athletically related compensation received from non-university sources.
- School Pay = Base salary; income from contract provisions other than base salary that are paid, or guaranteed, by the university or affiliated organizations, such as a foundation.
- Bonus Payout = Amount coach was paid for meeting personal or team-performance goals.
- Total Compensation = Total Pay + Bonuses Paid
- Graduation Success Rate = In the GSR calculation, student-athletes who depart a school while in good academic standing (would have met NCAA's and school's progress-toward-degree standards) are essentially passed from that school's cohort to another school's cohort. In the Federal rate, they are all considered non-graduates.

Other Sources:

- https://sports.usatoday.com/2019/10/16/2019-ncaa-football-head-coach-salaries-methodology/
- https://ncaaorg.s3.amazonaws.com/research/gradrates/RES_HowGradRateCalculated.pdf

Import Data and Merge

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# import packages for analysis and modeling
import pandas as pd  # data frame operations

import numpy as np  # arrays and math functions
from scipy.stats import uniform  # for training-and-test split
import statsmodels.api as sm  # statistical models (including regression)
import statsmodels.formula.api as smf  # R-like model specification
import matplotlib.pyplot as plt  # 2D plotting
import seaborn as sns  # PROVIDES TRELLIS AND SMALL MULTIPLE PLOTTING

#Read in the Coaches9.csv file supplied as part of this lab
coaches = pd.read_csv("/content/drive/My Drive/Colab Notebooks/IST718/Lab1/Coaches9.csv")
#coaches.head(10)


# Remove unneeded fields
coaches = coaches.drop(['AssistantPay','Buyout'], axis=1)

# Replace -- with zeros (NOTE: ZEROS ARE UTILIZED ONLY FOR CALCULATING TOTAL COMPENSATION AND NOT REGRESSION)
for value in list(coaches):
  coaches[value].replace('--', "$0", inplace=True)


# Remove $ and convert to float
coaches['SchoolPay'] = coaches['SchoolPay'].apply(lambda x: x.replace('$', '')).apply(lambda x: x.replace(',','')).astype(float)
coaches['TotalPay'] = coaches['TotalPay'].apply(lambda x: x.replace('$', '')).apply(lambda x: x.replace(',','')).astype(float)
coaches['Bonus'] = coaches['Bonus'].apply(lambda x: x.replace('$', '')).apply(lambda x: x.replace(',','')).astype(float)
coaches['BonusPaid'] = coaches['BonusPaid'].apply(lambda x: x.replace('$', '')).apply(lambda x: x.replace(',','')).astype(float)
```

```
#coaches.info()


# Create a new column TotalComp = TotalPay + BonusPaid
coaches['TotalComp'] = coaches['TotalPay'] + coaches['BonusPaid']

# Pay in thousands for plotting
coaches['pay_000'] = coaches['TotalPay']/1000
coaches['comp_000'] = coaches['TotalComp']/1000


#How many coaches have TotalComp of 0?
#coaches.loc[coaches['TotalComp'] == 0]


#Remove the four coaches with a TotalComp amount of 0
coaches = coaches.loc[coaches['TotalComp'] != 0]
#coaches.head(10)
```

Merge Stadium Data

```
#Import stadium data from GitHub

url = 'https://raw.githubusercontent.com/gboeing/data-visualization/master/ncaa-football-stadiums/data/stadiums-geocoded.csv'
stadiums = pd.read_csv(url, error_bad_lines=False)
#stadiums


#Fuzzy Matching of Coaches and Stadium file
#!pip install python_Levenshtein-0.12.0-cp36-cp36m-win_amd64.whl
!pip install fuzzymatcher
import fuzzymatcher

left_on = ["School", "Conference"]
right_on = ["team", "conference"]

dfjoined = fuzzymatcher.fuzzy_left_join(coaches,stadiums, left_on, right_on) #Merge on School and Conference
dfjoined = pd.DataFrame(dfjoined) #Dataframe creation
dfjoined = dfjoined.sort_values(by='best_match_score', ascending = False) #
dfjoined
dfjoined.to_excel("dfjoined2.xlsx")
coaches2 = dfjoined
```

```
Requirement already satisfied: fuzzymatcher in /usr/local/lib/python3.6/dist-packages (0.0.5)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from fuzzymatcher) (1.0.3)
Requirement already satisfied: python-Levenshtein in /usr/local/lib/python3.6/dist-packages (from fuzzymatcher) (0.12.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packages (from fuzzymatcher) (2.8.1)
Requirement already satisfied: fuzzywuzzy in /usr/local/lib/python3.6/dist-packages (from fuzzymatcher) (0.18.0)
Requirement already satisfied: metaphone in /usr/local/lib/python3.6/dist-packages (from fuzzymatcher) (0.6)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.6/dist-packages (from pandas->fuzzymatcher) (1.18.2)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas->fuzzymatcher) (2018.9)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from python-Levenshtein->fuzzymatcher) (46.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil->fuzzymatcher) (1.12.0)
```

```
# Total missing values for each feature
#print (coaches2.isnull().sum())


#Review schools with null values for team
coaches2[coaches2['team'].isnull()]
```

	best_match_score	__id_left	__id_right	School	Conference	Coach	SchoolPay	TotalPay	Bonus	BonusPaid	TotalComp
92	NaN	56_left	None	Massachusetts	Ind.	Mark Whipple	500000.0	500000.0	305000.0	20000.0	520000.0

```
# Remove Massachussetts due to missing stadium information (impute later if there is time)
coaches2 = coaches2[coaches2.School != 'Massachusetts']


# Check again for missing values - expanded is missing from 20
#print (coaches2.isnull().sum())


# Delete the expanded column and those from fuzzy match (_id_right _id_left, best_match_score columns)
coaches2 = coaches2.drop(['expanded'], axis=1)
coaches2 = coaches2.drop(['__id_right'], axis=1)
coaches2 = coaches2.drop(['__id_left'], axis=1)
coaches2 = coaches2.drop(['best_match_score'], axis=1)
```

Merge Standings Data

```
# Fuzzy Match the 2019 stats from https://www.sports-reference.com/cfb/years/2019-standings.html
# The use of the Python Selenium package was explored to automate the downloading of this csv file from sports-reference.com.
#However, since Java Script is used to mask the file, this was not attempted due to time constraints.
standings = pd.read_csv("/content/drive/My Drive/Colab Notebooks/IST718/Lab1/2019 CFB Standings.csv")
standings.head(100)

left_on = ["School", "Conference"]
right_on = ["School1", "Conf"]

dfjoined2 = fuzzymatcher.fuzzy_left_join(coaches2,standings, left_on, right_on) #Merge on School and Conference
dfjoined2 = pd.DataFrame(dfjoined2) #Dataframe creation
dfjoined2 = dfjoined2.sort_values(by='best_match_score', ascending = False) #
#dfjoined2.to_excel("dfjoined2.xlsx")
coaches3 = dfjoined2


# Check again for missing values
#print (coaches3.isnull().sum())


# Delete unused columens
coaches3 = coaches3.drop(['ConferenceW'], axis=1)
coaches3 = coaches3.drop(['ConferenceL'], axis=1)
coaches3 = coaches3.drop(['ConferencePct'], axis=1)
coaches3 = coaches3.drop(['Polls-APPre'], axis=1)
coaches3 = coaches3.drop(['Polls-APHigh'], axis=1)
coaches3 = coaches3.drop(['Polls-APPost'], axis=1)
coaches3 = coaches3.drop(['__id_right'], axis=1)
coaches3 = coaches3.drop(['__id_left'], axis=1)
coaches3 = coaches3.drop(['best_match_score'], axis=1)


# Check again for missing values
#print (coaches3.isnull().sum())
```

Scrape and Merge Coach Tenure Data

```
#Scrape data from Wikipedia - Source: https://simpleanalytical.com/how-to-web-scrape-wikipedia-python-urllib-beautiful-soup-pandas
# import the library we use to open URLs
import urllib.request
# specify which URL/web page we are going to be scraping
url = "https://en.wikipedia.org/wiki/List_of_current_NCAA_Division_I_FBS_football_coaches"


# open the url using urllib.request and put the HTML into the page variable
page = urllib.request.urlopen(url)


# import the BeautifulSoup library so we can parse HTML and XML documents
from bs4 import BeautifulSoup


# parse the HTML from our URL into the BeautifulSoup parse tree format
soup = BeautifulSoup(page, "lxml")


#print(soup.prettify())


soup.title
```

```
<title>List of current NCAA Division I FBS football coaches - Wikipedia</title>
```

```
soup.title.string
```

```
'List of current NCAA Division I FBS football coaches - Wikipedia'
```

```
# use the 'find_all' function to bring back all instances of the 'table' tag in the HTML and store in 'all_tables' variable
all_tables=soup.find_all("table")
#all_tables


right_table = soup.find('table',class_='wikitable sortable')
#right_table


A=[]
```

```
B=[]
C=[]
D=[]
E=[]
F=[]
G=[]
H=[]
I=[]
J=[]
K=[]
L=[]
M=[]


for row in right_table.findAll('tr'):
    cells=row.findAll('td')
    if len(cells)==13:
        A.append(cells[0].find(text=True))
        B.append(cells[1].find(text=True))
        C.append(cells[2].find(text=True))
        D.append(cells[3].find(text=True))
        E.append(cells[4].find(text=True))
        F.append(cells[5].find(text=True))
        G.append(cells[6].find(text=True))
        H.append(cells[7].find(text=True))
        I.append(cells[8].find(text=True))
        J.append(cells[9].find(text=True))
        K.append(cells[10].find(text=True))
        L.append(cells[11].find(text=True))
        M.append(cells[12].find(text=True))


hc=pd.DataFrame(A,columns=['School2'])
hc['Conference2']=B
hc['HeadCoach']=C
hc['FirstSeasonCurrentSchool']=D
hc['W-Current']=E
hc['L-Current']=F
hc['W-Current%']=G
hc['W-Career']=H
hc['L-Career']=I
hc['W-Career%']=J
hc['Offensive Coordinator']=K
hc['Defensive Coordinator']=L
hc['Special Teams Coordinator']=M
#hc


#Drop unused columns
hc = hc.drop(['Offensive Coordinator','Defensive Coordinator','Special Teams Coordinator','W-Current','L-Current','W-Current%','W-Car
# Remove \n in numeric data
hc = hc.replace('\n','', regex=True)
#hc


hc["FirstSeasonCurrentSchool"] = hc["FirstSeasonCurrentSchool"].apply(pd.to_numeric)
hc['YearsCurrentSchool'] = 2021-(hc['FirstSeasonCurrentSchool'])


#hc.head(10)


# Merge the 2020 stats from https://en.wikipedia.org/wiki/List_of_current_NCAA_Division_I_FBS_football_coaches

left_on = ["Coach", "School"]
right_on = ["HeadCoach", "School2"]

dfjoined3 = fuzzymatcher.fuzzy_left_join(coaches3,hc, left_on, right_on) #Merge on School and Conference
dfjoined3 = pd.DataFrame(dfjoined3) #Dataframe creation
dfjoined3 = dfjoined3.sort_values(by='best_match_score', ascending = False) #
#dfjoined3.to_excel("dfjoined3.xlsx")
coaches4 = dfjoined3
#coaches4.head(5)


# Check again for missing values
#print (coaches4.isnull().sum())
```

Merge Graduation Success Rate Data

```
# Import CSV file from NCAA with GSR data and fuzzy match of NCCA data and coaches4
```

```
# NCAA file available from https://www.icpsr.umich.edu/icpsrweb/NCAA/studies/30022/datadocumentation#
# File - DS4 Division I School Student-Athletes and Student Body
# The use of the Python Selenium package was explored to automate the downloading of this csv file from sports-reference.com.
#However, since Java Script is used to mask the file, this was not attempted due to time constraints.

NCAA = pd.read_excel("/content/drive/My Drive/Colab Notebooks/IST718/Lab1/ICPSR_30022/2018RES_File5-DISquadAggregationSA.xlsx")


#Include only those records from men's football
NCAA = NCAA.loc[NCAA['SPORT'] == 'MFB']


#Fix the 10 schools with mismatched data
coaches4.loc[coaches4['School'] == 'Texas', 'School'] = "University of Texas at Austin"
coaches4.loc[coaches4['School'] == 'North Carolina', 'School'] = "University of North Carolina, Chapel Hill"
coaches4.loc[coaches4['School'] == 'Penn State', 'School'] = "Pennsylvania State University"
coaches4.loc[coaches4['School'] == 'Georgia Tech', 'School'] = "Georgia Institute of Technology"
coaches4.loc[coaches4['School'] == 'Virginia Tech', 'School'] = "Virginia Polytechnic University"
coaches4.loc[coaches4['School'] == 'UCLA', 'School'] = "University of California Los Angeles"
coaches4.loc[coaches4['School'] == 'Missouri', 'School'] = "University of Missouri"
coaches4.loc[coaches4['School'] == 'Arkansas', 'School'] = "University of Arkansas"
coaches4.loc[coaches4['School'] == 'Miami (Fla.)', 'School'] = "University of Miami"


#Delete 4 schools with no graduation data and fuzzy match columns
coaches4 = coaches4[coaches4.School != 'Army']
coaches4 = coaches4[coaches4.School != 'LSU']
coaches4 = coaches4[coaches4.School != 'Navy']
coaches4 = coaches4[coaches4.School != 'Buffalo']
coaches4 = coaches4.drop(['__id_right'], axis=1)
coaches4 = coaches4.drop(['__id_left'], axis=1)
coaches4 = coaches4.drop(['best_match_score'], axis=1)


#Fuzzy Matching of Coaches9 and NCAA file
#!pip install python_Levenshtein-0.12.0-cp36-cp36m-win_amd64.whl
#!pip install fuzzymatcher
import fuzzymatcher

left_on = ["School", "Conference"]
right_on = ["SCL_NAME", "SCL_CONFERENCE"]

dfjoined4 = fuzzymatcher.fuzzy_left_join(coaches4,NCAA, left_on, right_on) #Merge on School and Conference
dfjoined4 = pd.DataFrame(dfjoined4) #Dataframe creation
dfjoined4 = dfjoined4.sort_values(by='best_match_score', ascending = False) #
dfjoined4
#dfjoined4.to_excel("dfjoined.xlsx")
coaches5 = dfjoined4


#Drop unused columns
coaches5 = coaches5.drop(['SCL_CONFERENCE','SCL_DIVISION','SCL_SUBDIVISION','SPORT','SPONSORED','SCL_HBCU', 'FED_RATE', '__id_right',

# print the first ten of the data frame
#coaches5.head(100)


# Total missing values for each feature
#print (coaches5.isnull().sum())


coaches5['RankedPost'] = pd.Categorical(coaches5.RankedPost)


#coaches5.dtypes
```

Create new subsets by conference

```
#List unique values in the Conference column
#coaches.Conference.unique()
MtWest = coaches5[coaches5['Conference'] == 'Mt. West']
MidAmerica = coaches5[coaches5['Conference'] == 'MAC']
Southeast = coaches5[coaches5['Conference'] == 'SEC']
CUSA = coaches5[coaches5['Conference'] == 'C-USA']
SunBelt = coaches5[coaches5['Conference'] == 'Sun Belt']
Pac12 = coaches5[coaches5['Conference'] == 'Pac-12']
American = coaches5[coaches5['Conference'] == 'AAC']
Atlantic = coaches5[coaches5['Conference'] == 'ACC']
BigTen = coaches5[coaches5['Conference'] == 'Big Ten']
Big12 = coaches5[coaches5['Conference'] == 'Big 12']
Ind = coaches5[coaches5['Conference'] == 'Ind.']
```

```
# convert into list of vectors for box plot
data = [MtWest['comp_000'], MidAmerica['comp_000'],
     Southeast['comp_000'], CUSA['comp_000'],
     SunBelt['comp_000'], Pac12['comp_000'],
     American['comp_000'], Atlantic['comp_000'], BigTen['comp_000'], Big12['comp_000'], Ind['comp_000']]
ordered_conference_names = ['MtWest','MidAmerica','Southeast','CUSA','SunBelt','Pac12','American','Atlantic','BigTen','Big12','Ind']
```

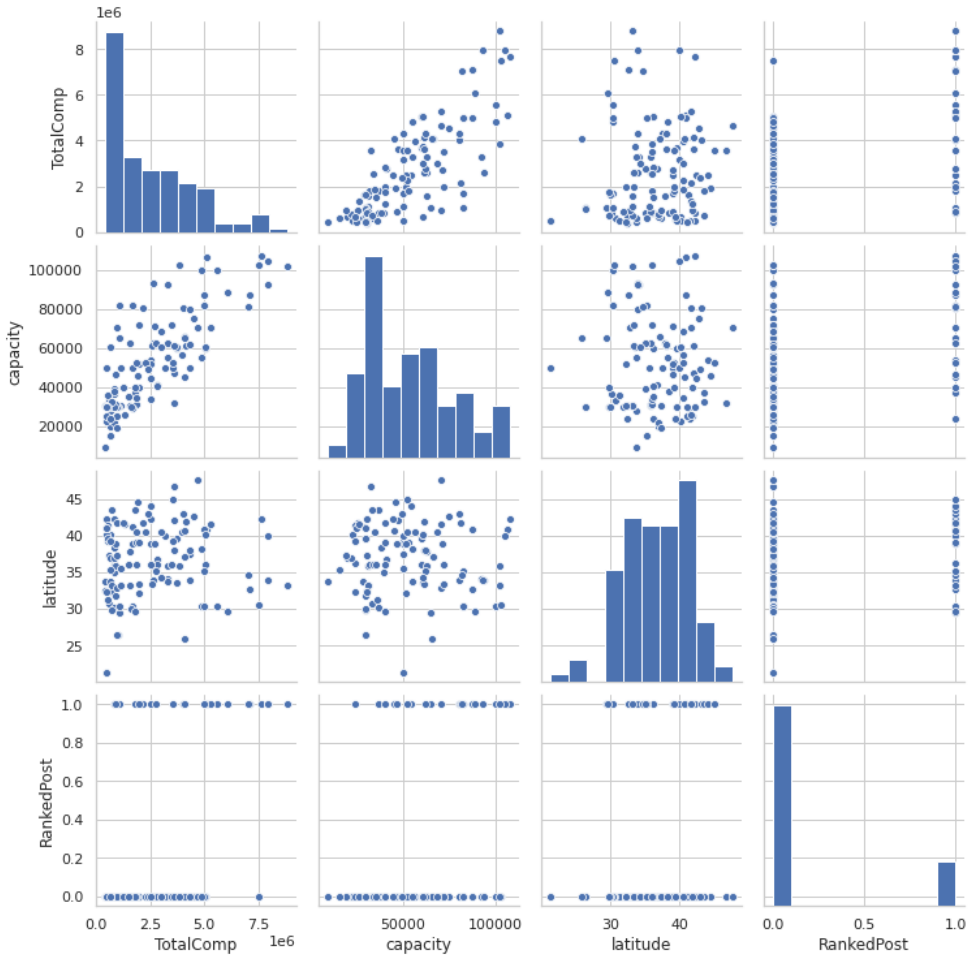
```
#Look at the final data for Syracuse
syracuse = coaches5.loc[coaches5['School'] == 'Syracuse']
syracuse
```

	School	Conference	Coach	SchoolPay	TotalPay	Bonus	BonusPaid	TotalComp	pay_000	comp_000	stadium	city	state	tea
81	Syracuse	ACC	Dino Babers	2401206.0	2401206.0	0.0	0.0	2401206.0	2401.206	2401.206	Carrier Dome	Syracuse	NY	Syracu

```
# Create subsets plots below
coaches5_subset = coaches5[["TotalComp"\
, "capacity", "latitude","GSR","longitude","YearsCurrentSchool"]]
coaches5_subset2 = coaches5[["TotalComp"\
, "capacity", "latitude","RankedPost"]]
```

Bivariate Plot of Some Continuous Variables

```
#import matplotlib.pyplot as plt
import seaborn as sns

sns.pairplot(coaches5_subset2)
plt.show()
```



Summary Statistics

```
## SUMMARY STATISTICS

len(coaches5.index)

#np.mean(coaches5['TotalComp'])
```

```
120
```

```
np.min(coaches5['TotalComp'])
```

```
390000.0
```

```
np.max(coaches5['TotalComp'])
```

```
8807000.0
```

```
np.mean(coaches5['TotalComp'])
```

```
2563497.6
```

```
np.median(coaches5["TotalComp"])
```

```
2000000.0
```

```
np.mean(Atlantic["TotalComp"])
```

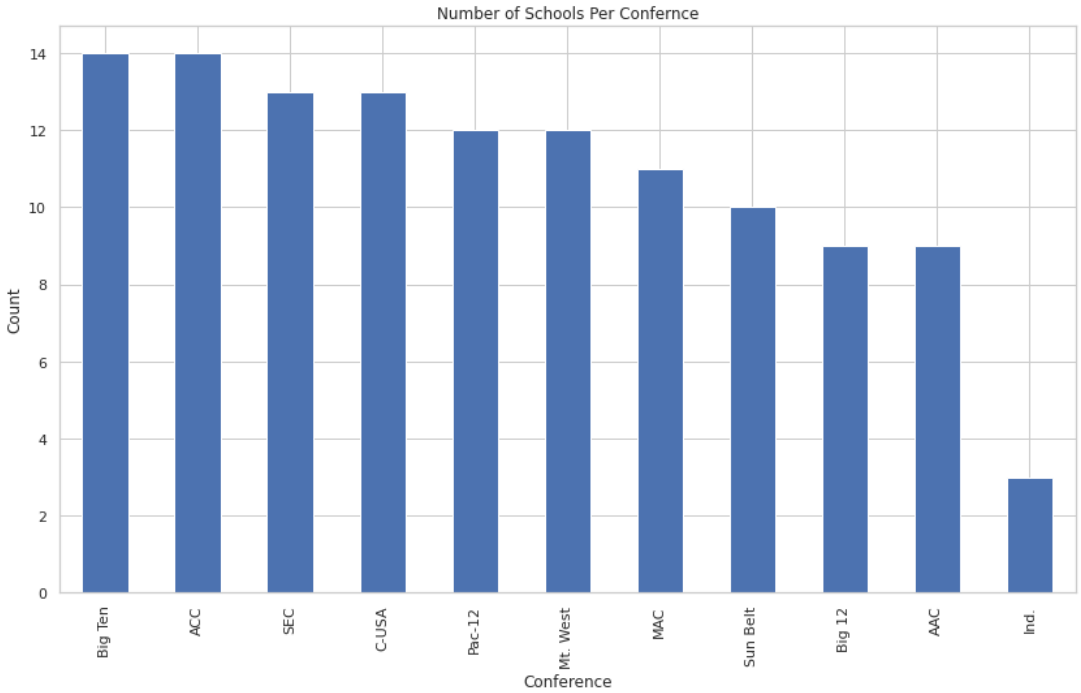
```
3581647.9285714286
```

```
np.median(Atlantic["TotalComp"])
```

```
3280009.0
```

```
# NUMBER OF SCHOOLS PER CONFERENCE

ax = coaches5['Conference'].value_counts().plot(kind='bar',
                                    figsize=(14,8),
                                    title="Number of Schools Per Confernce")
ax.set_xlabel("Conference")
ax.set_ylabel("Count")
```

```
Text(0, 0.5, 'Count')
```

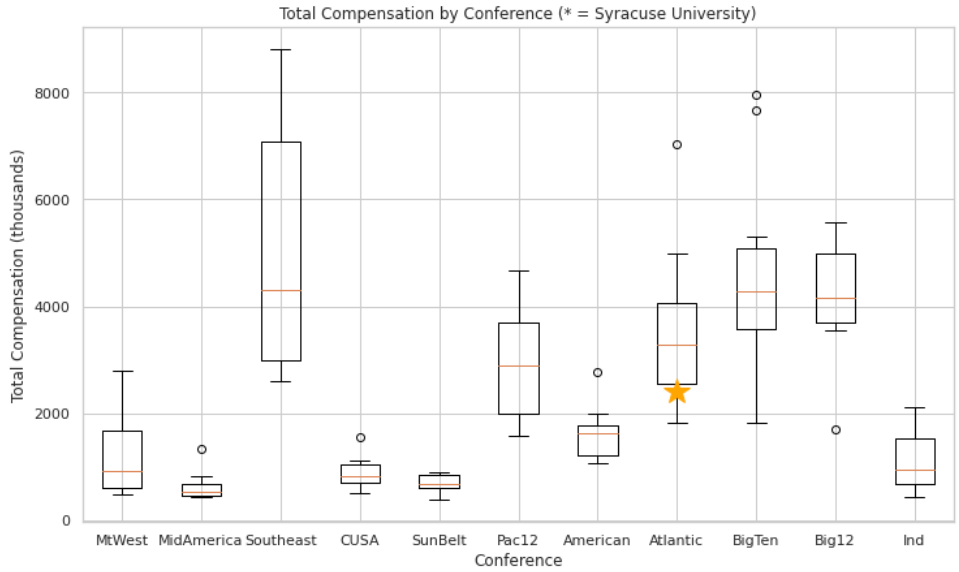



```
# CREATE A HISTOGRAM OF TOTAL COMPENSATION

# CREATE BINS
# plt.hist(coaches5['TotalComp'], stacked = False, rwidth = .9)
# plt.title("Total Compensation Histogram")
# plt.xlabel('Millions USD')
```

```
# plt.ylabel('Frequency')

# plt.show()
```

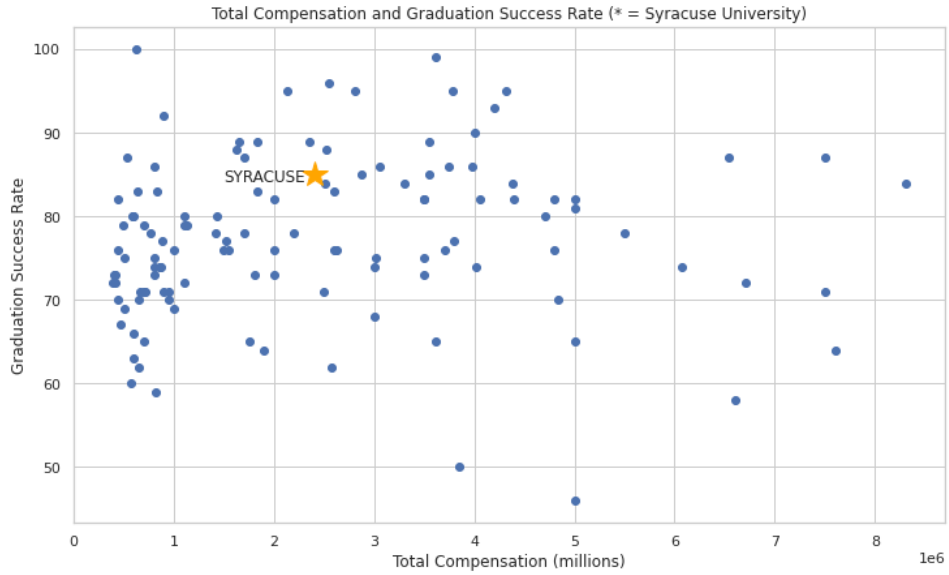
Total Compensation by Conference

```
# exploratory data analysis: box plot for Total Pay by Conference
plt.rcParams['figure.figsize'] = [12, 7]
fig, axis = plt.subplots()

axis.set_xlabel('Conference')
axis.set_ylabel('Total Compensation (thousands)')
plt.title('Total Compensation by Conference (* = Syracuse University)')
conf_plot = plt.boxplot(data, sym='o', vert=1, whis=1.5)
plt.setp(conf_plot['boxes'], color = 'black')
plt.setp(conf_plot['whiskers'], color = 'black')
plt.setp(conf_plot['fliers'], color = 'black', marker = 'o')

axis.set_xticklabels(ordered_conference_names)

plt.plot(8,2401,'*',color="orange",markersize=20)
plt.show()
```

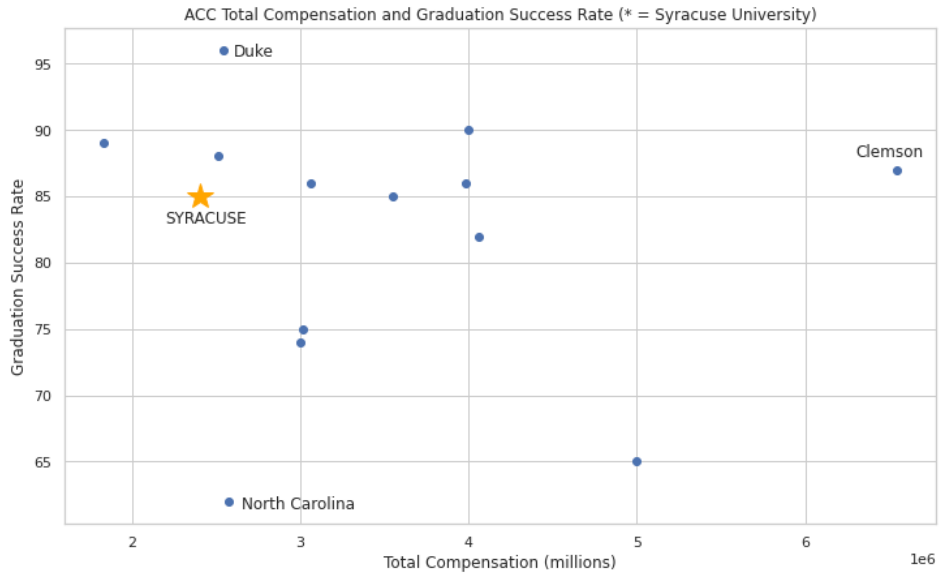


Total Compensation and Graduation Success Rate

```
# Plot all schools pay vs GSR
g = plt.scatter(x="TotalPay", y="GSR", data = coaches5)
plt.title('Total Compensation and Graduation Success Rate (* = Syracuse University)')
plt.xlabel('Total Compensation (millions)')
plt.ylabel('Graduation Success Rate')
plt.plot(2401206,85,'*',color="orange",markersize=20)
plt.text(1500000,84.2,'SYRACUSE')
plt.show()
```

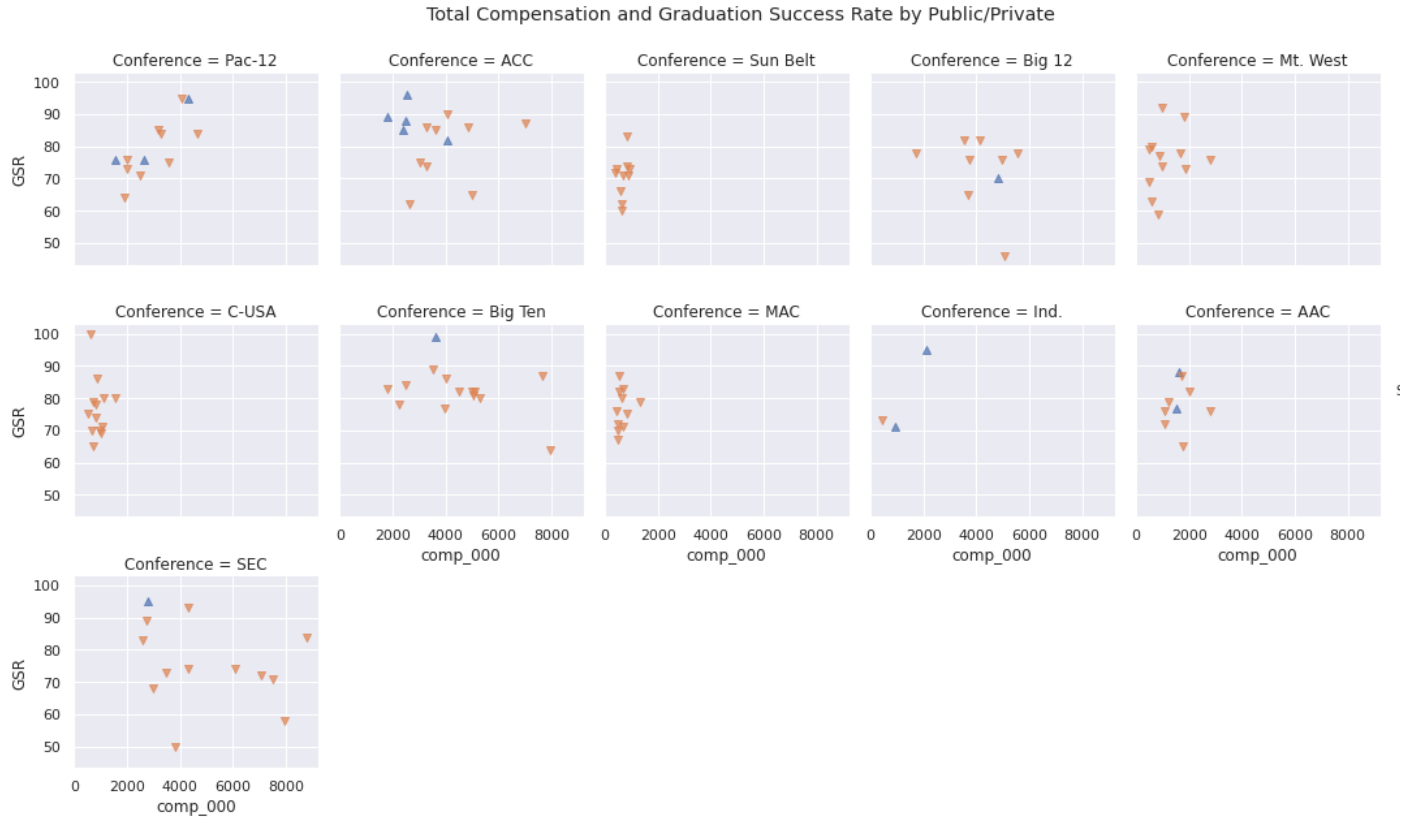



```
# Plot all schools pay vs GSR
g = plt.scatter(x="TotalPay", y="GSR", data = Atlantic)
plt.title('ACC Total Compensation and Graduation Success Rate (* = Syracuse University)')
plt.xlabel('Total Compensation (millions)')
plt.ylabel('Graduation Success Rate')
plt.plot(2401206,85,'*',color="orange",markersize=20)
plt.text(2600000,95.6,'Duke')
plt.text(2650000,61.5,'North Carolina')
plt.text(6300000,88,'Clemson')
plt.text(2200000,83,'SYRACUSE')
plt.show()
```



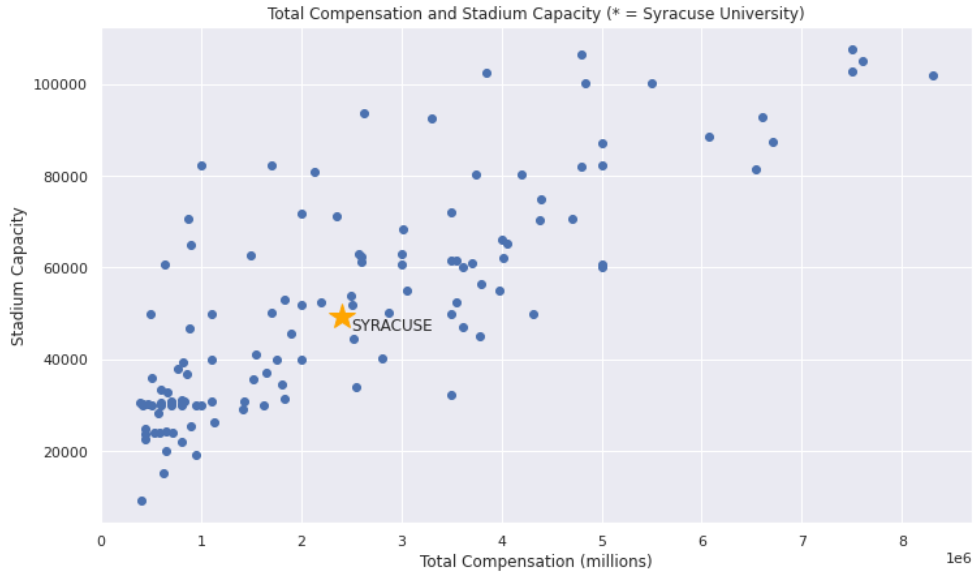

```
# trellis/lattice plot Total Compensation and GSR conditioning on Public or Private

# This plot was included to show the disproportionate amount of private schools in the Atlantic
# Coast Conference and the above average Graduation Success Scores in this conference.

import seaborn as sns

sns.set(style="darkgrid")

g = sns.FacetGrid(coaches5, col="Conference", hue="SCL_PRIVATE",
                  hue_order=[1,0],
                  col_wrap=5,
                  hue_kws=dict(marker=["^", "v"]))
g.map(plt.scatter, "comp_000", "GSR", alpha=.7)

g.add_legend();
```

```
plt.subplots_adjust(top=0.9)
g.fig.suptitle ('Total Compensation and Graduation Success Rate by Public/Private')
#plt.show()
# plt.savefig('Total_Comp_GSR_By_Conference_By_PrivPub.pdf',
#     bbox_inches = 'tight', dpi=None, facecolor='w', edgecolor='b',
#     orientation='portrait', papertype=None, format=None,
#     transparent=True, pad_inches=0, frameon=None)

plt.show()
```



Total Compensation and Stadium Capacity

```
#Note the strong positive linear relationship between Stadium Capacity and Total Compensation.  This explains visually why
#this variable was the strongest predictor in the regression equations.

# Plot all schools pay vs Stadium
g = plt.scatter(x="TotalPay", y="capacity", data = coaches5)
plt.title('Total Compensation and Stadium Capacity (* = Syracuse University)')
plt.xlabel('Total Compensation (millions)')
plt.ylabel('Stadium Capacity')
plt.plot(2401206,49250,'*',color="orange",markersize=20)
plt.text(2500000,46250,'SYRACUSE')
plt.show()
```



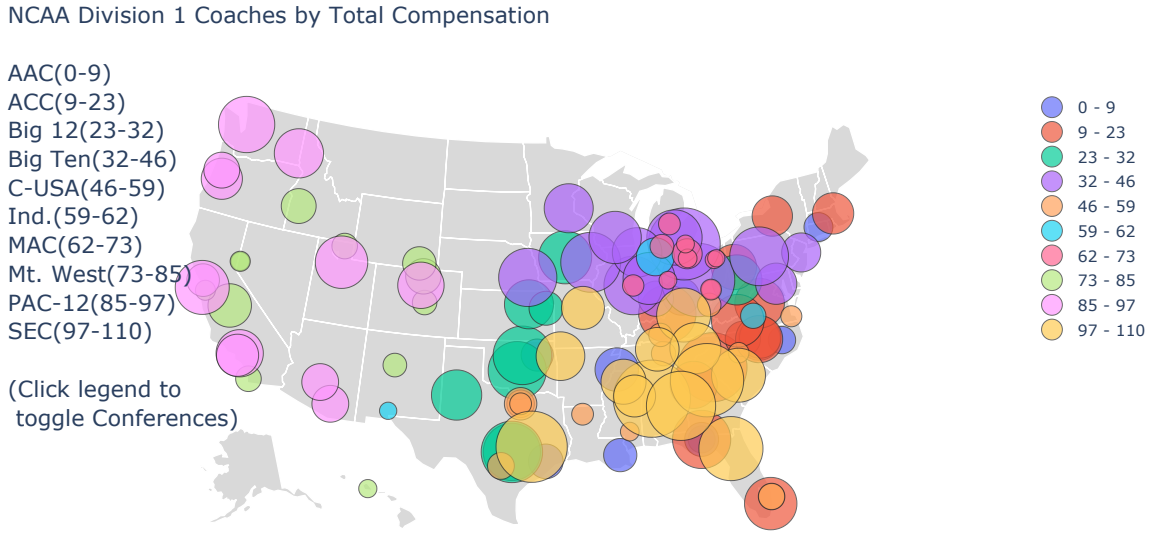
Geographic Analysis

```
#!pip install plotly
#!pip install chart-studio
import plotly.graph_objects as go
import pandas as pd
import plotly.io as pio
import plotly
pio.renderers
```

```
Renderers configuration
-----------------------
    Default renderer: 'colab'
    Available renderers:
        ['plotly_mimetype', 'jupyterlab', 'nteract', 'vscode',
         'notebook', 'notebook_connected', 'kaggle', 'azure', 'colab',
         'cocalc', 'databricks', 'json', 'png', 'jpeg', 'jpg', 'svg',
         'pdf', 'browser', 'firefox', 'chrome', 'chromium', 'iframe',
         'iframe_connected', 'sphinx_gallery']
```

```
#Create df with name,pop, lat, lon (School, TotalComp, latitude, longitude)

df = coaches5[['School','TotalComp','latitude','longitude','Conference']]
df = df.sort_values(by=['Conference'],ascending=True)
df = df.reset_index()


#Look up the index numbers for the map below
#df.loc[df['Conference'] == "AAC"]
#df.loc[df['Conference'] == "ACC"]
#df.loc[df['Conference'] == "Big 12"]
#df.loc[df['Conference'] == "Big Ten"]
#df.loc[df['Conference'] == "C-USA"]
#df.loc[df['Conference'] == "Ind."]
#df.loc[df['Conference'] == "MAC"]
#df.loc[df['Conference'] == "Mt. West"]
#df.loc[df['Conference'] == "Pac-12"]
#df.loc[df['Conference'] == "SEC"]
#df.loc[df['Conference'] == "Sunbelt"]


# THIS IS AN INTERACTIVE GRAPHIC.  CLICK THE LEGEND BUBBLES TO SHOW/HIDE DIFFERENT REGIONS.
# The plot could be improved by replacing the numbers in the legend with the names of the regions.
# However, this was difficult technically and the left legend was added as a temporary solution.

#Geographic view of Coaches Compensation by School.  Code Source: https://plotly.com/python/bubble-maps/
df['text'] = df['School'] + " " + (df['TotalComp']/1e6).astype(str)+' million'
limits = [(0,9),(9,23),(23,32),(32,46),(46,59),(59,62),(62,73),(73,85),(85,97),(97,110)]
colors = ["royalblue","crimson","lightseagreen","orange","lightgrey"]
#cities = []
scale = 5000
```

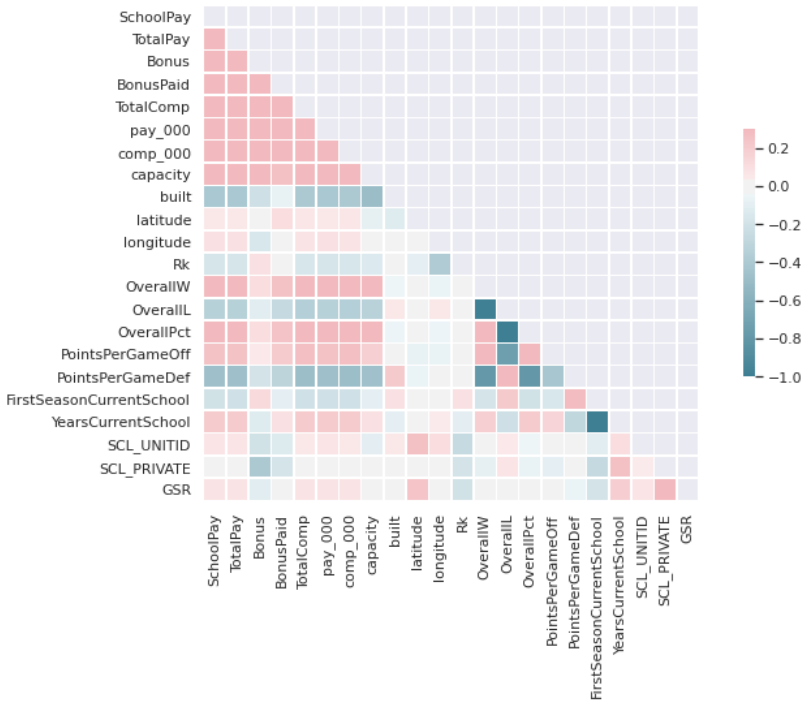
```
fig = go.Figure()

for i in range(len(limits)):
    lim = limits[i]
    df_sub = df[lim[0]:lim[1]]
    fig.add_trace(go.Scattergeo(
        locationmode = 'USA-states',
        lon = df_sub['longitude'],
        lat = df_sub['latitude'],
        #text = df_sub['School'],
        text = df_sub['text'],
        marker = dict(
            size = df_sub['TotalComp']/scale,
            #color = colors[i],
            line_color='rgb(40,40,40)',
            line_width=0.5,
            sizemode = 'area'
        ),
        name = '{0} - {1}'.format(lim[0],lim[1])))

fig.update_layout(
        title_text = 'NCAA Division 1 Coaches by Total Compensation\
        <br><br>AAC(0-9)<br>ACC(9-23)\
        <br>Big 12(23-32)<br>Big Ten(32-46)<br>C-USA(46-59)<br>Ind.(59-62)\
        <br>MAC(62-73)<br>Mt. West(73-85)<br>PAC-12(85-97)<br>SEC(97-110)\
        <br><br>(Click legend to<br> toggle Conferences)',
        showlegend = True,
        geo = dict(
            scope = 'usa',
            landcolor = 'rgb(217, 217, 217)',
        )
    )
fig.show()
```



Correlation Analysis

```
# The correlation analysis was used to determine which variables to test in the regression equations below.

corr = coaches5.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
#f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)
```

```
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.show()
```



Regression Analysis

Without Conference Model - Traditional

Process:

1. Review the correlation matrix above to determine which variables are most highly correlated with TotalComp.
2. Test different combinations of variables to find which provide the highest adjusted R-squared values. A Principal Component Analysis could be conducted to provide a more comprehensive analysis of variables but this was not done due to time constraints.
3. The final variables selected were - Stadium Capacity + Years Coach at Current School + Ranked at the End of Season + Latitude

```
# employ training-and-test regimen for model validation - Simple Model
np.random.seed(1234)
coaches5['runiform'] = uniform.rvs(loc = 0, scale = 1, size = len(coaches5))
coaches5_train = coaches5[coaches5['runiform'] >= 0.33]
coaches5_test = coaches5[coaches5['runiform'] < 0.33]
# check training data frame
print('\ncoaches5_train data frame (rows, columns): ',coaches5_train.shape)
print(coaches5_train.head())
# check test data frame
print('\ncoaches5_test data frame (rows, columns): ',coaches5_test.shape)
print(coaches5_test.head())

# specify a simple model
my_model = str('TotalComp ~ capacity + YearsCurrentSchool + RankedPost + latitude')

# fit the model to the training set
train_model_fit = smf.ols(my_model, data = coaches5_train).fit()
# summary of model fit to the training set
print(train_model_fit.summary())
# training set predictions from the model fit to the training set
coaches5_train['predict_totalcomp'] = train_model_fit.fittedvalues

# test set predictions from the model fit to the training set
coaches5_test['predict_totalcomp'] = train_model_fit.predict(coaches5_test)
```

```
coaches5_train data frame (rows, columns):  (86, 40)
                                    School Conference  ...   GSR  runiform
148  University of North Carolina, Chapel Hill        ACC  ...  62.0  0.622109
2                          Georgia Southern   Sun Belt  ...  62.0  0.437728
101            University of Texas at Austin     Big 12  ...  78.0  0.785359
118                          West Virginia     Big 12  ...  65.0  0.779976
188                         Nevada-Las Vegas   Mt. West  ...  63.0  0.801872

[5 rows x 40 columns]

coaches5_test data frame (rows, columns):  (34, 40)
                                School Conference  ...   GSR  runiform
88   University of California Los Angeles     Pac-12  ...  84.0  0.191519
145       Georgia Institute of Technology        ACC  ...  86.0  0.272593
5                        San Jose State   Mt. West  ...  80.0  0.276464
149                      Michigan State    Big Ten  ...  82.0  0.013768
135                        Old Dominion      C-USA  ...  70.0  0.075381

[5 rows x 40 columns]
                            OLS Regression Results
==============================================================================
Dep. Variable:              TotalComp   R-squared:                       0.704
Model:                            OLS   Adj. R-squared:                  0.690
Method:                 Least Squares   F-statistic:                     48.24
Date:                Sat, 18 Apr 2020   Prob (F-statistic):           1.09e-20
Time:                        17:01:14   Log-Likelihood:                -1322.1
No. Observations:                  86   AIC:                             2654.
Df Residuals:                      81   BIC:                             2667.
Df Model:                           4
Covariance Type:            nonrobust
================================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept          -3.419e+06   1.07e+06     -3.191      0.002   -5.55e+06   -1.29e+06
RankedPost[T.1]     8.885e+05   3.73e+05      2.380      0.020    1.46e+05    1.63e+06
capacity              62.1659      5.700     10.907      0.000       50.825      73.507
YearsCurrentSchool  5.378e+04   3.05e+04      1.763      0.082   -6916.371    1.14e+05
latitude            6.417e+04   2.68e+04      2.393      0.019    1.08e+04    1.18e+05
==============================================================================
Omnibus:                        1.803   Durbin-Watson:                   1.463
Prob(Omnibus):                  0.406   Jarque-Bera (JB):                1.440
Skew:                          -0.315   Prob(JB):                        0.487
Kurtosis:                       3.064   Cond. No.                     4.99e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.99e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:20: SettingWithCopyWarning:


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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:23: SettingWithCopyWarning:


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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
```

```
# compute the proportion of response variance
# accounted for when predicting out-of-sample
print('\nProportion of Test Set Variance Accounted for: ',\
    round(np.power(coaches5_test['TotalComp'].corr(coaches5_test['predict_totalcomp']),2),3))

# use the full data set to obtain an estimate of the increase in
# pay due to being ranked at the end of the season, controlling for other factors
my_model_fit = smf.ols(my_model, data = coaches5).fit()
print(my_model_fit.summary())
```

```
Proportion of Test Set Variance Accounted for:  0.6
                            OLS Regression Results
==============================================================================
Dep. Variable:              TotalComp   R-squared:                       0.683
Model:                            OLS   Adj. R-squared:                  0.672
Method:                 Least Squares   F-statistic:                     62.00
Date:                Sat, 18 Apr 2020   Prob (F-statistic):           7.97e-28
Time:                        17:01:14   Log-Likelihood:                -1841.4
No. Observations:                 120   AIC:                             3693.
Df Residuals:                     115   BIC:                             3707.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept          -2.733e+06   8.87e+05     -3.080      0.003   -4.49e+06   -9.75e+05
RankedPost[T.1]     5.819e+05   3.09e+05      1.884      0.062   -2.99e+04    1.19e+06
capacity              62.2009      4.813     12.922      0.000      52.666      71.736
YearsCurrentSchool   4.64e+04   2.46e+04      1.886      0.062   -2330.433    9.51e+04
latitude            4.654e+04   2.22e+04      2.096      0.038    2547.782    9.05e+04
==============================================================================
Omnibus:                        2.204   Durbin-Watson:                   1.484
Prob(Omnibus):                  0.332   Jarque-Bera (JB):                1.672
Skew:                          -0.247   Prob(JB):                        0.433
Kurtosis:                       3.302   Cond. No.                     4.92e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.92e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
#Calculate the Skewness of variables used in regression
#!pip install scipy.stats
from scipy.stats import kurtosis, skew

print( 'Skewness of Capacity (should be ~0): {}'.format(skew(coaches5.capacity)))
print( 'Skewness of YearsCurrentSchool (should be ~0): {}'.format(skew(coaches5.YearsCurrentSchool)))
print( 'Skewness of Latitude (should be ~0): {}'.format(skew(coaches5.latitude)))
```

```
Skewness of Capacity (should be ~0): 0.5789985767768407
Skewness of YearsCurrentSchool (should be ~0): 1.754857547468558
Skewness of Latitude (should be ~0): -0.3275696671757959
```

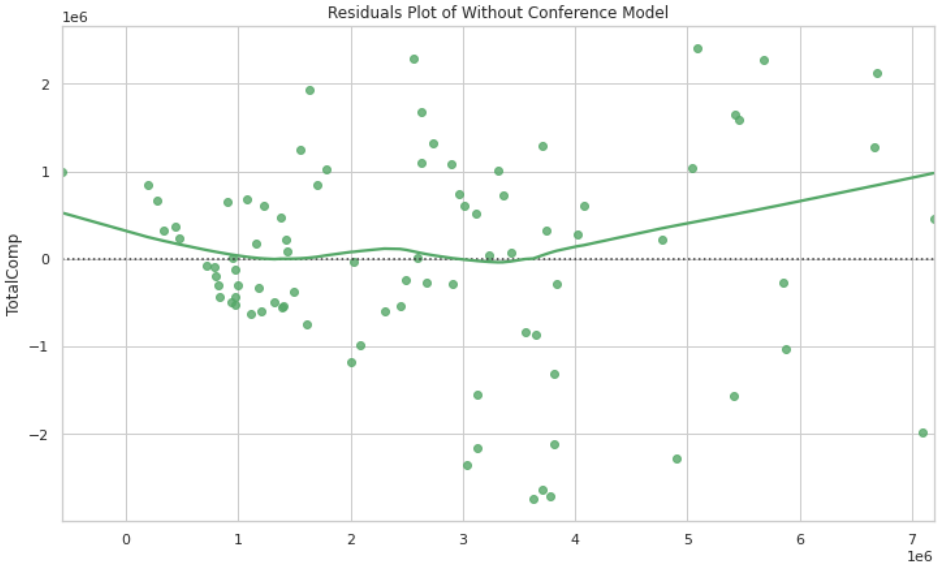
```
#Check for multi-collinearity - since the values are below 5, we do not need to drop variables due to colinearity
#Code adapted from https://etav.github.io/python/vif_factor_python.html
# For each X, calculate VIF and save in dataframe
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Get variables for which to compute VIF and add intercept term
X = coaches5_train[['capacity','YearsCurrentSchool','latitude']]
X['intercept'] = 1

# Compute and view VIF
vif = pd.DataFrame()
vif["variables"] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

# View results using print
print(vif)
```

```
             variables        VIF
0             capacity   1.019660
1  YearsCurrentSchool   1.004843
2             latitude   1.015333
3            intercept  67.644680
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:


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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
```

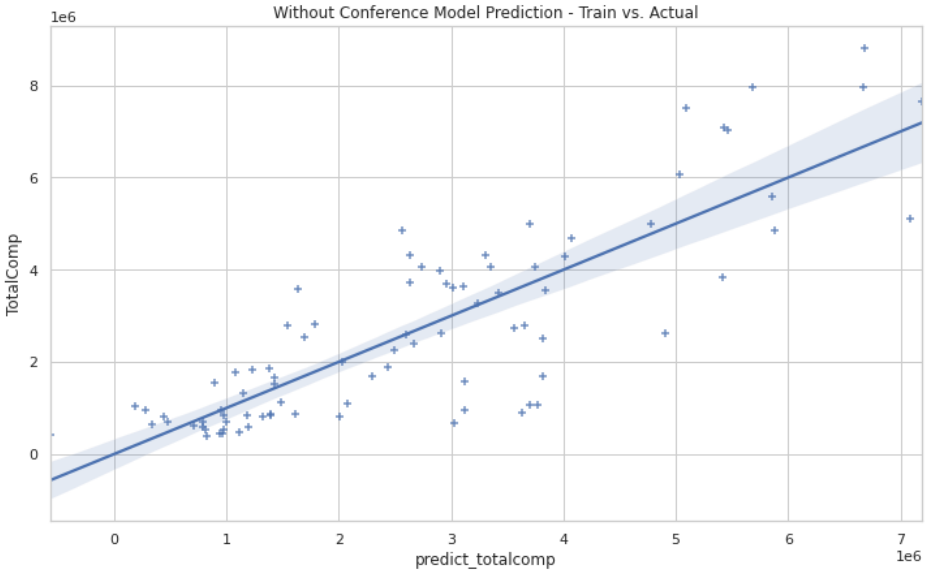
```
#The residplot() function checks whether the simple regression model is appropriate for the dataset.
#It plots the residual values for each observation. Ideally, these values should be randomly scattered around y = 0.
#Code adapted from https://seaborn.pydata.org/examples/residplot.html

sns.set(style="whitegrid")
```

```
sns.set(style= whitegrid )

# Plot the residuals after fitting a linear model
rx = sns.residplot(train_model_fit.fittedvalues, coaches5_train['TotalComp'], lowess=True, color="g")
rx.set_title('Residuals Plot of Without Conference Model')
```

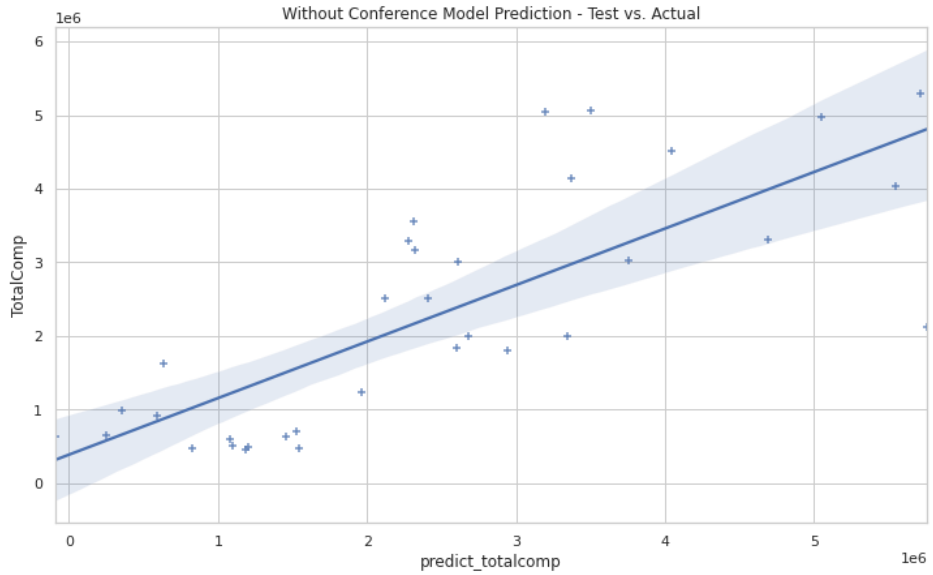
```
Text(0.5, 1.0, 'Residuals Plot of Without Conference Model')
```




```
ax = sns.regplot('predict_totalcomp', 'TotalComp', marker="+", data=coaches5_train).set_title('Without Conference Model Prediction -
```




```
ax = sns.regplot('predict_totalcomp', 'TotalComp', marker="+", data=coaches5_test).set_title('Without Conference Model Prediction - T
```



Without Conference Model - Bayesian

```
n Bayesian regression with Pystan
te: The MCMC regression does not support categorical variables so RankedPost is not included.
ode source: http://marubon-ds.blogspot.com/2017/10/bayesian-multiple-regression-by-stan.html
ort pystan
a = {'N':coaches5_subset.shape[0], 'TotalComp': coaches5_subset['TotalComp'], 'capacity':coaches5_subset['capacity'], 'YearsCurrentSc
 = pystan.stan(file='/content/drive/My Drive/Colab Notebooks/IST718/Lab1/coaches5stan.stan', data=data, seed=42)

e code below was copied to the file coaches5stan.stan
ata {
    int N;
    real TotalComp[N];
    real capacity[N];
    real YearsCurrentSchool[N];
    real latitude[N];


arameters {
    real b1;
    real b2;
    real b3;
    real b4;
    real<lower=0> sigma;


odel {
    for (i in 1:N)
        TotalComp[i] ~ normal(b1 + b2 * capacity[i] + b3 * YearsCurrentSchool[i] + b4 * latitude[i], sigma);
```

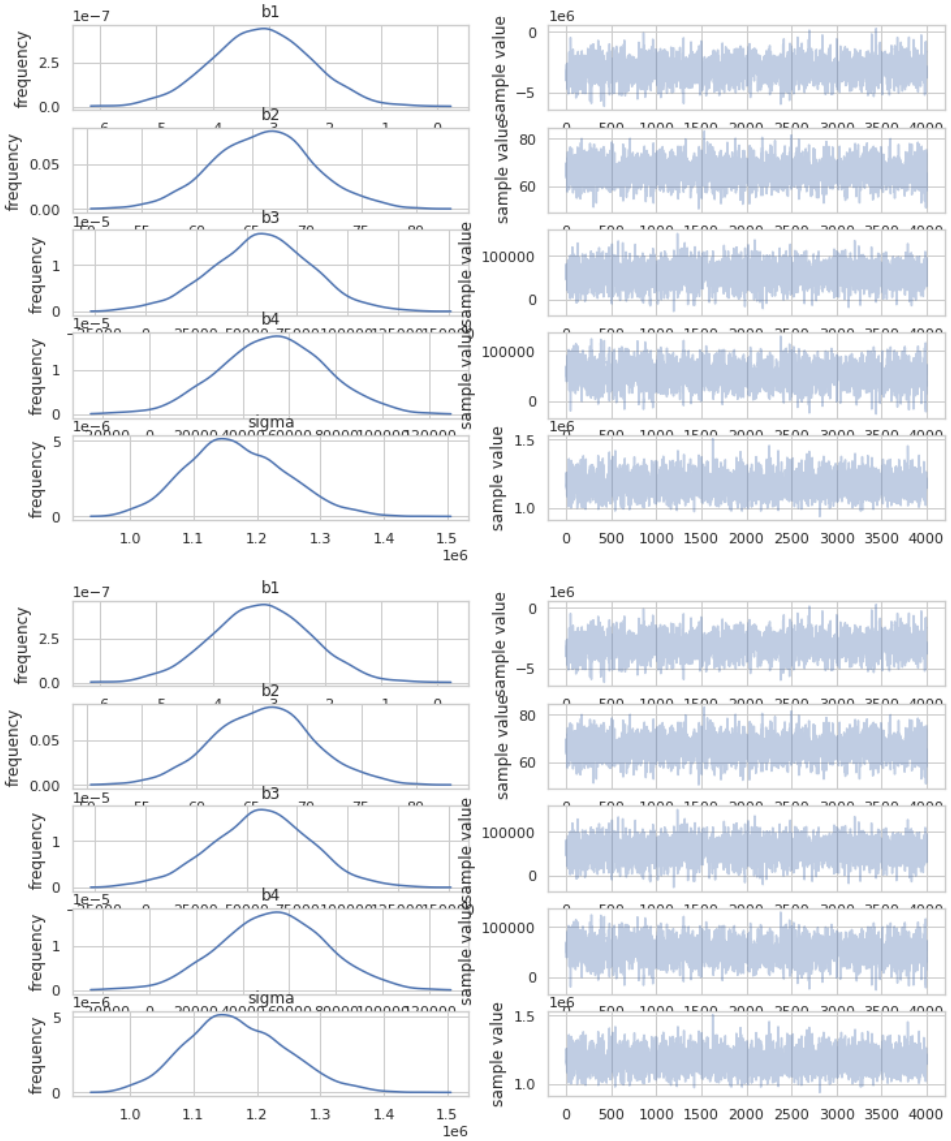
```
WARNING:pystan:DeprecationWarning: pystan.stan was deprecated in version 2.17 and will be removed in version 3.0. Compile and us
INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_8ad35686f86add28a7ace4592bef2cff NOW.
```

```
print(fit)
```

```
Inference for Stan model: anon_model_8ad35686f86add28a7ace4592bef2cff.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

        mean se_mean     sd   2.5%    25%    50%    75%  97.5%  n_eff   Rhat
b1    -3.1e6   2.2e4  8.8e5 -4.9e6 -3.7e6 -3.1e6 -2.5e6 -1.4e6   1643    1.0
b2      66.1     0.09   4.64  57.07  62.92  66.18   69.1  75.44   2470    1.0
b3     5.7e4  471.04  2.5e4 6178.1  4.1e4  5.8e4  7.4e4  1.1e5   2768    1.0
b4     5.3e4  545.28  2.3e4 9128.2  3.8e4  5.3e4  6.8e4  9.7e4   1704    1.0
sigma  1.2e6  1499.7  7.8e4  1.0e6  1.1e6  1.2e6  1.2e6  1.3e6   2715    1.0
lp__   -1721    0.04   1.66  -1725  -1722  -1721  -1720  -1719   1597    1.0

Samples were drawn using NUTS at Sat Apr 18 17:02:33 2020.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

```
fit.plot()

#Note that none of the variables in the 2.5% - 97.5% confidence interval crosses zero.
#The left plots below show a mostly gaussian distribution with some slight right or left skew
#in some variables.  The right plots show the Markov Chain Monte Carlo simulation has
#values that mostly in the same range with a few outliers.
```

```
WARNING:pystan:Deprecation warning. PyStan plotting deprecated, use ArviZ library (Python 3.5+). `pip install arviz`; `arviz.plo
```



With Sub-Conference Model - Traditional

Sub-Conference (categorical) + Stadium Capacity + Years Coach at Current School + Ranked at the End of Season

```
# Employ training-and-test regimen for model validation - Conference Categories
```

```
# Employ training-and-test regimen for model validation - Conference categories

my_model2 = str('TotalComp ~ Conf + capacity + OverallPct + YearsCurrentSchool + latitude + RankedPost')

# fit the model to the training set
train_model_fit2 = smf.ols(my_model2, data = coaches5_train).fit()
# summary of model fit to the training set
print(train_model_fit2.summary())
# training set predictions from the model fit to the training set
coaches5_train['predict_totalcomp2'] = train_model_fit2.fittedvalues

# test set predictions from the model fit to the training set
coaches5_test['predict_totalcomp2'] = train_model_fit2.predict(coaches5_test)
```

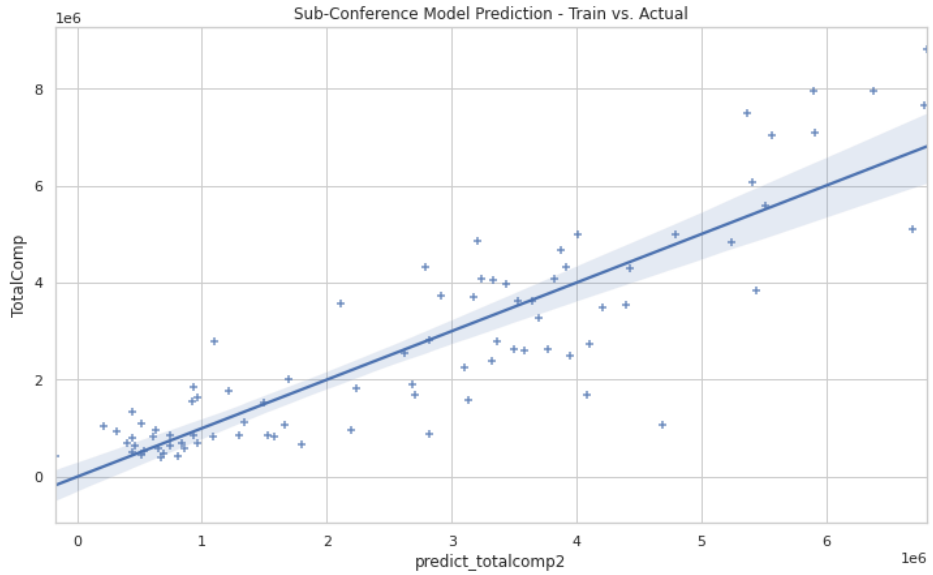
```
                            OLS Regression Results
==============================================================================
Dep. Variable:              TotalComp   R-squared:                       0.774
Model:                            OLS   Adj. R-squared:                  0.685
Method:                 Least Squares   F-statistic:                     8.703
Date:                Sat, 18 Apr 2020   Prob (F-statistic):           4.90e-12
Time:                        17:02:38   Log-Likelihood:                -1310.6
No. Observations:                  86   AIC:                             2671.
Df Residuals:                      61   BIC:                             2733.
Df Model:                          24
Covariance Type:            nonrobust
================================================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                -1.273e+06   1.93e+06     -0.661      0.511   -5.12e+06    2.58e+06
Conf[T.ACC (Coastal)]     4.653e+04   7.08e+05      0.066      0.948   -1.37e+06    1.46e+06
Conf[T.American (East)]  -2.35e+06   9.89e+05     -2.377      0.021   -4.33e+06   -3.74e+05
Conf[T.American (West)]  -8.98e+05   8.71e+05     -1.031      0.307   -2.64e+06    8.44e+05
Conf[T.Big 12]           -3.34e+05   7.31e+05     -0.457      0.649   -1.79e+06    1.13e+06
Conf[T.Big Ten (East)]   -8.859e+04  9.03e+05     -0.098      0.922   -1.89e+06    1.72e+06
Conf[T.Big Ten (West)]   -4.178e+04  8.07e+05     -0.052      0.959   -1.65e+06    1.57e+06
Conf[T.CUSA (East)]      -1.218e+06  8.51e+05     -1.431      0.157   -2.92e+06    4.84e+05
Conf[T.CUSA (West)]      -1.072e+06  8.39e+05     -1.279      0.206   -2.75e+06    6.05e+05
Conf[T.Ind]              -1.234e+06  8.96e+05     -1.377      0.173   -3.02e+06    5.57e+05
Conf[T.MAC (East)]        -1.53e+06  8.99e+05     -1.701      0.094   -3.33e+06    2.69e+05
Conf[T.MAC (West)]       -1.767e+06  8.69e+05     -2.034      0.046    -3.5e+06   -3.01e+04
Conf[T.MWC (Mountain)]   -1.435e+06  8.11e+05     -1.770      0.082   -3.06e+06    1.86e+05
Conf[T.MWC (West)]       -1.383e+06  7.93e+05     -1.745      0.086   -2.97e+06    2.02e+05
Conf[T.Pac-12 (North)]   -4.751e+05  7.31e+05     -0.650      0.518   -1.94e+06    9.86e+05
Conf[T.Pac-12 (South)]   -1.118e+06  8.31e+05     -1.346      0.183   -2.78e+06    5.43e+05
Conf[T.SEC (East)]        1.705e+05  6.55e+05      0.260      0.795   -1.14e+06    1.48e+06
Conf[T.SEC (West)]        3.617e+05  7.05e+05      0.513      0.610   -1.05e+06    1.77e+06
Conf[T.Sun Belt (East)] -1.103e+06   8.94e+05     -1.234      0.222   -2.89e+06    6.84e+05
Conf[T.Sun Belt (West)] -1.267e+06      8e+05     -1.584      0.118   -2.87e+06    3.33e+05
RankedPost[T.1]           9.371e+05  5.77e+05      1.623      0.110   -2.17e+05    2.09e+06
capacity                    45.0406    10.277      4.383      0.000      24.491      65.590
OverallPct               -5.072e+04  1.02e+06     -0.050      0.960   -2.09e+06    1.98e+06
YearsCurrentSchool        3.865e+04  3.43e+04      1.126      0.264      -3e+04    1.07e+05
latitude                  5.115e+04  4.28e+04      1.196      0.236   -3.44e+04    1.37e+05
==============================================================================
Omnibus:                        6.604   Durbin-Watson:                   1.732
Prob(Omnibus):                  0.037   Jarque-Bera (JB):                6.791
Skew:                          -0.442   Prob(JB):                       0.0335
Kurtosis:                       4.056   Cond. No.                     1.12e+06
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.12e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: SettingWithCopyWarning:


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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: SettingWithCopyWarning:


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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
```

```
# compute the proportion of response variance
# accounted for when predicting out-of-sample
print('\nProportion of Test Set Variance Accounted for: ',\
    round(np.power(coaches5_test['TotalComp'].corr(coaches5_test['predict_totalcomp2']), 2), 3))
```

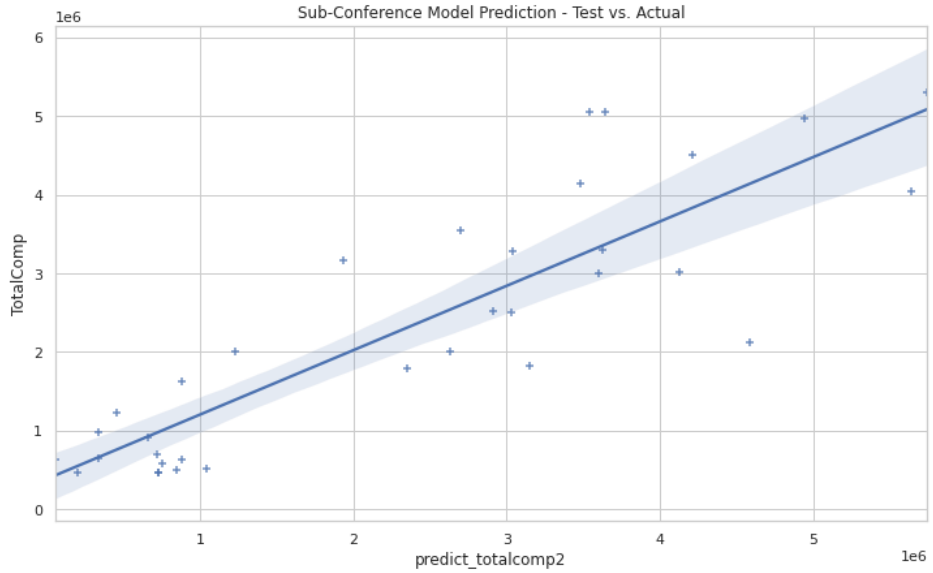
```
    round(np.power(coaches5_test['TotalComp'].corr(coaches5_test['predict_totalcomp2']),2),3))

# use the full data set to obtain an estimate of the increase in
# pay due to being ranked at the end of the season, controlling for other factors
my_model_fit2 = smf.ols(my_model2, data = coaches5).fit()
print(my_model_fit2.summary())

print('\nEstimated Value of Being ACC: ',\
    round(my_model_fit2.params['Conf[T.ACC (Coastal)]'],0))
print('\nEstimated Value of Being Big Ten East: ',\
    round(my_model_fit2.params['Conf[T.Big Ten (East)]'],0))
print('\nEstimated Value of Being Big Ten West: ',\
    round(my_model_fit2.params['Conf[T.Big Ten (West)]'],0))
print('\nEstimated Value of Being American East: ',\
    round(my_model_fit2.params['Conf[T.American (East)]'],0))
print('\nEstimated Value of Being American Wast: ',\
    round(my_model_fit2.params['Conf[T.American (West)]'],0))
```

```
Proportion of Test Set Variance Accounted for:  0.754
                            OLS Regression Results
==============================================================================
Dep. Variable:              TotalComp   R-squared:                       0.779
Model:                            OLS   Adj. R-squared:                  0.723
Method:                 Least Squares   F-statistic:                     13.94
Date:                Sat, 18 Apr 2020   Prob (F-statistic):           1.28e-21
Time:                        17:02:38   Log-Likelihood:                -1819.8
No. Observations:                 120   AIC:                             3690.
Df Residuals:                      95   BIC:                             3759.
Df Model:                          24
Covariance Type:            nonrobust
================================================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                -2.945e+05   1.41e+06     -0.209      0.835   -3.09e+06     2.5e+06
Conf[T.ACC (Coastal)]    -7.448e+04   5.46e+05     -0.137      0.892   -1.16e+06    1.01e+06
Conf[T.American (East)] -1.936e+06    6.6e+05     -2.934      0.004   -3.25e+06   -6.26e+05
Conf[T.American (West)] -8.352e+05   6.82e+05     -1.224      0.224   -2.19e+06    5.19e+05
Conf[T.Big 12]            1.308e+05   5.15e+05      0.254      0.800   -8.92e+05    1.15e+06
Conf[T.Big Ten (East)]   -2.583e+04   5.86e+05     -0.044      0.965   -1.19e+06    1.14e+06
Conf[T.Big Ten (West)]    9.633e+04   5.83e+05      0.165      0.869   -1.06e+06    1.25e+06
Conf[T.CUSA (East)]      -1.243e+06   6.11e+05     -2.034      0.045   -2.46e+06      -3e+04
Conf[T.CUSA (West)]      -1.396e+06   6.56e+05     -2.130      0.036    -2.7e+06   -9.48e+04
Conf[T.Ind]              -1.889e+06   6.64e+05     -2.846      0.005   -3.21e+06   -5.71e+05
Conf[T.MAC (East)]       -1.748e+06   6.45e+05     -2.709      0.008   -3.03e+06   -4.67e+05
Conf[T.MAC (West)]       -1.825e+06   6.02e+05     -3.031      0.003   -3.02e+06    -6.3e+05
Conf[T.MWC (Mountain)]   -1.554e+06   6.43e+05     -2.417      0.018   -2.83e+06   -2.77e+05
Conf[T.MWC (West)]       -1.602e+06   5.91e+05     -2.710      0.008   -2.78e+06   -4.28e+05
Conf[T.Pac-12 (North)]    -3.63e+05   5.99e+05     -0.606      0.546   -1.55e+06    8.26e+05
Conf[T.Pac-12 (South)]   -9.801e+05   5.69e+05     -1.724      0.088   -2.11e+06    1.49e+05
Conf[T.SEC (East)]        3.247e+05   5.53e+05      0.587      0.559   -7.73e+05    1.42e+06
Conf[T.SEC (West)]        4.251e+05   5.67e+05      0.749      0.455   -7.01e+05    1.55e+06
Conf[T.Sun Belt (East)] -1.304e+06   6.87e+05     -1.898      0.061   -2.67e+06    5.96e+04
Conf[T.Sun Belt (West)] -1.503e+06   6.59e+05     -2.279      0.025   -2.81e+06   -1.94e+05
RankedPost[T.1]           7.784e+05      4e+05      1.944      0.055   -1.66e+04    1.57e+06
capacity                    38.3492      7.378      5.197      0.000      23.701      52.997
OverallPct                5.972e+04    6.9e+05      0.087      0.931   -1.31e+06    1.43e+06
YearsCurrentSchool        3.625e+04   2.44e+04      1.488      0.140   -1.21e+04    8.46e+04
latitude                  3.339e+04   3.08e+04      1.084      0.281   -2.77e+04    9.45e+04
==============================================================================
Omnibus:                        7.066   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.029   Jarque-Bera (JB):               11.407
Skew:                          -0.176   Prob(JB):                      0.00333
Kurtosis:                       4.469   Cond. No.                     1.15e+06
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.15e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

Estimated Value of Being ACC:  -74483.0

Estimated Value of Being Big Ten East:  -25828.0

Estimated Value of Being Big Ten West:  96326.0

Estimated Value of Being American East:  -1935563.0
```

```
ax = sns.regplot('predict_totalcomp2', 'TotalComp', marker="+", data=coaches5_train).set_title('Sub-Conference Model Prediction - Tra
```




```
ax = sns.regplot('predict_totalcomp2', 'TotalComp', marker="+", data=coaches5_test).set_title('Sub-Conference Model Prediction - Test
```



With Conference Model - Traditional

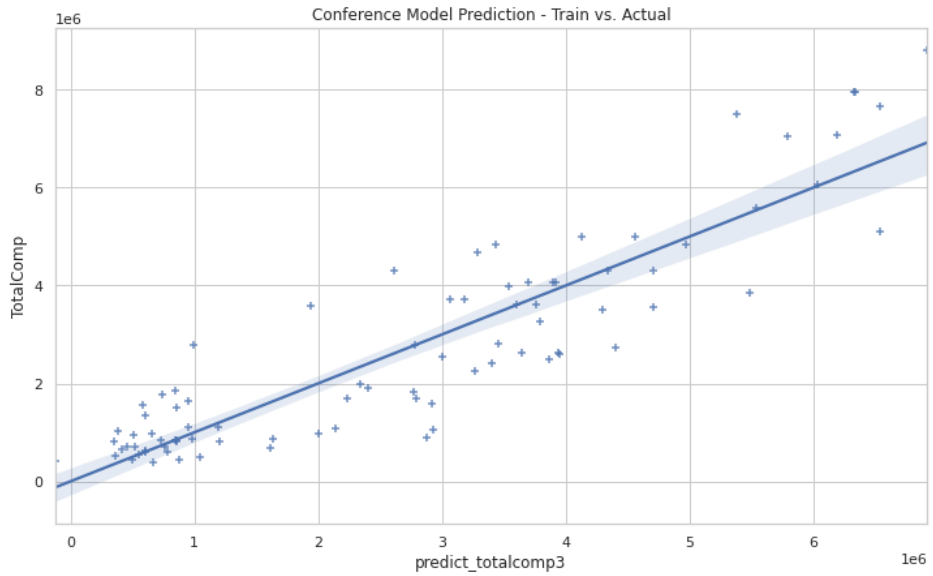
```
# Employ training-and-test regimen for model validation - Conference Categories

my_model3 = str('TotalComp ~ Conference + capacity + OverallPct + YearsCurrentSchool + latitude + RankedPost')

# fit the model to the training set
train_model_fit3 = smf.ols(my_model3, data = coaches5_train).fit()
# summary of model fit to the training set
print(train_model_fit3.summary())
# training set predictions from the model fit to the training set
coaches5_train['predict_totalcomp3'] = train_model_fit3.fittedvalues

# test set predictions from the model fit to the training set
coaches5_test['predict_totalcomp3'] = train_model_fit3.predict(coaches5_test)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              TotalComp   R-squared:                       0.809
Model:                            OLS   Adj. R-squared:                  0.769
Method:                 Least Squares   F-statistic:                     19.83
Date:                Sat, 18 Apr 2020   Prob (F-statistic):           1.86e-19
Time:                        17:02:39   Log-Likelihood:                -1303.2
No. Observations:                  86   AIC:                             2638.
Df Residuals:                      70   BIC:                             2678.
Df Model:                          15
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                    -1.376e+06   1.43e+06     -0.962      0.339   -4.23e+06    1.48e+06
Conference[T.ACC]             1.958e+06   5.53e+05      3.540      0.001    8.55e+05    3.06e+06
Conference[T.Big 12]          1.546e+06    6.5e+05      2.377      0.020    2.49e+05    2.84e+06
Conference[T.Big Ten]         1.958e+06   6.66e+05      2.939      0.004    6.29e+05    3.29e+06
Conference[T.C-USA]          -1.345e+04   6.13e+05     -0.022      0.983   -1.24e+06    1.21e+06
Conference[T.Ind.]            2.508e+05   8.74e+05      0.287      0.775   -1.49e+06    1.99e+06
Conference[T.MAC]            -5.648e+04   6.54e+05     -0.086      0.931   -1.36e+06    1.25e+06
Conference[T.Mt. West]        1.088e+05   5.67e+05      0.192      0.848   -1.02e+06    1.24e+06
Conference[T.Pac-12]          1.116e+06   6.02e+05      1.854      0.068   -8.46e+04    2.32e+06
Conference[T.SEC]             2.423e+06   5.48e+05      4.421      0.000    1.33e+06    3.52e+06
Conference[T.Sun Belt]        7.057e+04   6.11e+05      0.116      0.908   -1.15e+06    1.29e+06
RankedPost[T.1]               1.096e+06   4.56e+05      2.403      0.019    1.86e+05    2.01e+06
capacity                        33.6671      8.284      4.064      0.000       17.145      50.189
OverallPct                    2.222e+05   8.24e+05      0.270      0.788   -1.42e+06    1.87e+06
YearsCurrentSchool            3.319e+04   2.82e+04      1.175      0.244   -2.31e+04    8.95e+04
latitude                      2.102e+04   3.36e+04      0.626      0.533   -4.59e+04    8.79e+04
==============================================================================
Omnibus:                        0.594   Durbin-Watson:                   1.932
Prob(Omnibus):                  0.743   Jarque-Bera (JB):                0.722
Skew:                           0.100   Prob(JB):                        0.697
Kurtosis:                       2.599   Cond. No.                     8.40e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.4e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: SettingWithCopyWarning:


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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: SettingWithCopyWarning:


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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
```

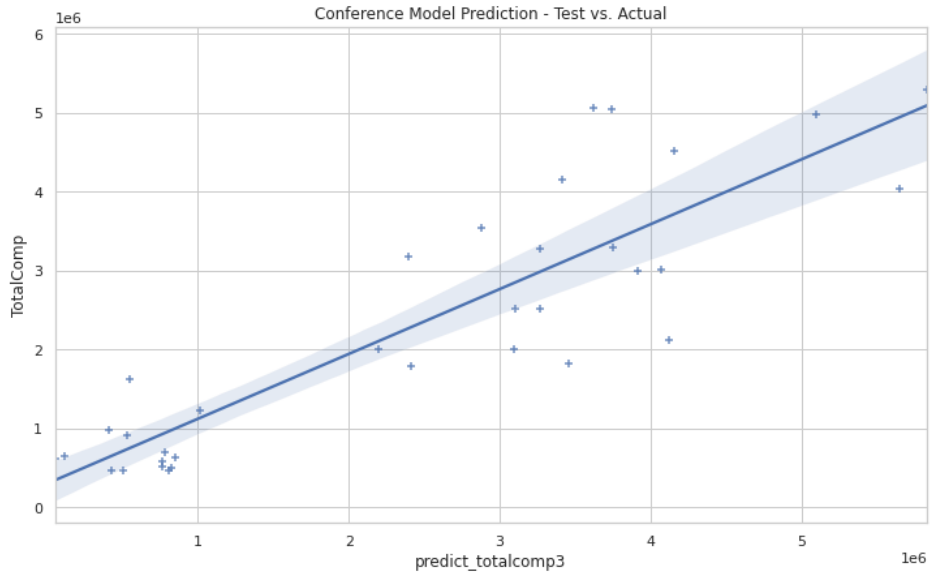
```
# compute the proportion of response variance
# accounted for when predicting out-of-sample
print('\nProportion of Test Set Variance Accounted for: ',\
    round(np.power(coaches5_test['TotalComp'].corr(coaches5_test['predict_totalcomp3']),2),3))

# use the full data set to obtain an estimate of the increase in
# pay due to being ranked at the end of the season, controlling for other factors
my_model_fit3 = smf.ols(my_model3, data = coaches5).fit()
print(my_model_fit3.summary())
```

```
Proportion of Test Set Variance Accounted for:  0.778
                            OLS Regression Results
==============================================================================
Dep. Variable:              TotalComp   R-squared:                       0.808
Model:                            OLS   Adj. R-squared:                  0.780
Method:                 Least Squares   F-statistic:                     29.10
Date:                Sat, 18 Apr 2020   Prob (F-statistic):           1.84e-30
Time:                        17:02:39   Log-Likelihood:                -1811.4
No. Observations:                 120   AIC:                             3655.
Df Residuals:                     104   BIC:                             3699.
Df Model:                          15
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                    -6.468e+05   1.08e+06     -0.599      0.550   -2.79e+06    1.49e+06
Conference[T.ACC]             1.74e+06   4.22e+05      4.120      0.000    9.02e+05    2.58e+06
Conference[T.Big 12]         1.822e+06   4.69e+05      3.887      0.000    8.93e+05    2.75e+06
Conference[T.Big Ten]        1.829e+06   4.71e+05      3.885      0.000    8.95e+05    2.76e+06
Conference[T.C-USA]          -1.622e+05   4.38e+05     -0.370      0.712   -1.03e+06    7.07e+05
Conference[T.Ind.]           -5.297e+05   6.33e+05     -0.837      0.404   -1.78e+06    7.25e+05
Conference[T.MAC]            -2.926e+05   4.61e+05     -0.635      0.527   -1.21e+06    6.21e+05
Conference[T.Mt. West]       -1.465e+05   4.26e+05     -0.344      0.732   -9.92e+05    6.99e+05
Conference[T.Pac-12]          1.01e+06   4.48e+05      2.255      0.026    1.22e+05     1.9e+06
Conference[T.SEC]             2.36e+06   4.39e+05      5.382      0.000    1.49e+06    3.23e+06
Conference[T.Sun Belt]       -1.642e+05   4.64e+05     -0.354      0.724   -1.08e+06    7.56e+05
RankedPost[T.1]              9.182e+05   3.44e+05      2.666      0.009    2.35e+05     1.6e+06
capacity                       29.1663      6.343      4.598      0.000      16.587      41.745
OverallPct                   3.322e+05   5.96e+05      0.558      0.578   -8.49e+05    1.51e+06
YearsCurrentSchool           3.391e+04   2.09e+04      1.621      0.108   -7578.357    7.54e+04
latitude                     9642.5817   2.53e+04      0.381      0.704   -4.05e+04    5.98e+04
==============================================================================
Omnibus:                        2.660   Durbin-Watson:                   2.064
Prob(Omnibus):                  0.264   Jarque-Bera (JB):                2.477
Skew:                           0.351   Prob(JB):                        0.290
Kurtosis:                       2.963   Cond. No.                     7.67e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.67e+05. This might indicate that there are
strong multicollinearity or other numerical problems
```

```
ax = sns.regplot('predict_totalcomp3', 'TotalComp', marker="+", data=coaches5_train).set_title('Conference Model Prediction - Train v
```




```
ax = sns.regplot('predict_totalcomp3', 'TotalComp', marker="+", data=coaches5_test).set_title('Conference Model Prediction - Test vs.
```



Predictions

Predict Total Compensation

```
# Prediction for Syracuse head coach total compensation in ACC (without conference model)
coaches5['predict_totalcomp'] = my_model_fit.predict(coaches5)
syracusepredict = coaches5.loc[coaches5['School']== 'Syracuse']
SyrACC = round(syracusepredict.predict_totalcomp,0)
SyrACC
```

```
81    2565451.0
Name: predict_totalcomp, dtype: float64
```

```
# Prediction for Syracuse head coach total compensation in ACC (sub-conference model)
coaches5['predict_totalcomp2'] = my_model_fit2.predict(coaches5)
syracusepredict2 = coaches5.loc[coaches5['School']== 'Syracuse']
SyrACC2 = round(syracusepredict2.predict_totalcomp2,0)
SyrACC2
```

```
81    3237476.0
Name: predict_totalcomp2, dtype: float64
```

```
my_model_fit.params
```

```
Intercept            -2.732960e+06
RankedPost[T.1]       5.819396e+05
capacity              6.220095e+01
YearsCurrentSchool    4.639596e+04
latitude              4.654311e+04
dtype: float64
```

```
# Prediction for Syracuse head coach total compensation in Former Big East (Now American - East) using sub-conference model

ACCCoef = round(my_model_fit2.params['Conf[T.ACC (Coastal)]'],0)
AmericanCoef = (round(my_model_fit2.params[2],0))
SyrACC2-ACCCoef+AmericanCoef
```

```
81    1376396.0
Name: predict_totalcomp2, dtype: float64
```

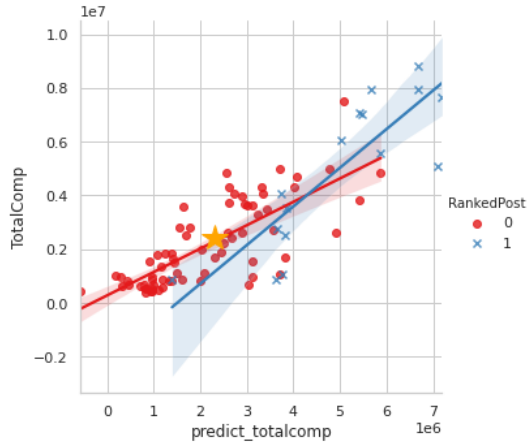
```
# Prediction for Syracuse head coach total compensation in Big Ten - East using conference model
BigTenCoef = (round(my_model_fit2.params[5],0))
SyrACC2-ACCCoef+BigTenCoef
```

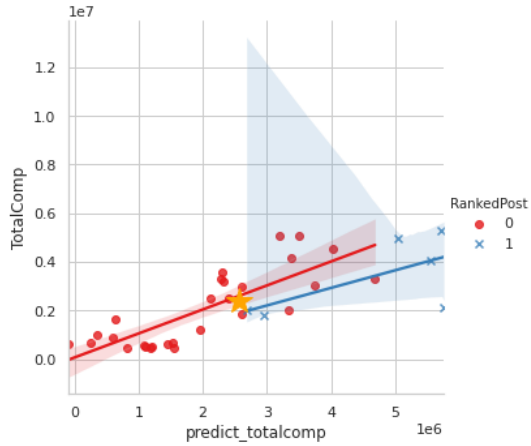
```
81    3286131.0
Name: predict_totalcomp2, dtype: float64
```

Predict Impact of Team Being Ranked End of Season

```
# Plots of the predictions vs actual for train and test in simple model
# Need to substitute numbers with variables
sns.lmplot('predict_totalcomp','TotalComp', hue = 'RankedPost', markers=["o","x"], palette='Set1', data=coaches5_train)
plt.plot(2295596,2401206,'*',color="orange",markersize=20)
sns.lmplot('predict_totalcomp','TotalComp', hue = 'RankedPost', markers=["o","x"], palette='Set1', data=coaches5_test)
plt.plot(syracusepredict.predict_totalcomp,syracusepredict.TotalComp,'*',color="orange",markersize=20)
```

```
[<matplotlib.lines.Line2D at 0x7f7252e408d0>]
```






```
#Predict the value of being ranked in the post season polls with simple model
print('\nEstimated Value of Being Ranked in the Post Season Polls: ',\
    round(my_model_fit.params["RankedPost[T.1]"],0))
```

```
Estimated Value of Being Ranked in the Post Season Polls:  581940.0
```