IST 718

LAB 1 ASSIGNMENT

Matthew L. Pergolski

Professor Jillian Lando

Overview

In the IST 718 Lab 1 assignment, the O-S-E-M-IN method will be conducted to recommend a specific salary of Syracuse's next Head Football Coach. In order to provide the analysis, a brief explanation of the data science method will be required.

- O
    - Obtain: In the obtaining section, Data Acquisition will be discussed and referenced.
- S
    - Scrub: In the scrubbing section, Data Cleaning will be discussed and referenced.
- E
    - Explore: In the exploring section, Data Exploration will be discussed and referenced.
- M
    - Model: In the modeling section, Data Modeling techniques will be discussed – the workings of our linear model will be introduced and referenced.
- IN
    - Interpret: In the interpreting section, we will summarize the results and provide the overall recommendation to the stakeholder.

To achieve the recommendation, a modeling technique of 'Ordinary Least Squares' (OLS) will be conducted. All in all, OLS is an optimization strategy that aids in finding a straight line as close as possible to data points in a linear regression model. The main predictor that will be used as the dependent variable in the analysis will be 'TotalPay.' The independent variables will be discussed and introduced in the following sections.

Data Acquisition

Multiple data sets were considered in the analysis. They are the following:

- Coaches
    - https://raw.githubusercontent.com/2SUBDA/IST_718/68273222b88e35e1d7ef4f2c874e667ee64dca60/Coaches9.csv
- Stadiums
    - https://www.collegegridirons.com/comparisons-by-capacity/https://www.teamrankings.com/ncf/trends/win_trends/
- Teamrecord
    - https://www.teamrankings.com/ncf/trends/win_trends/

- Grad
    - /content/sample_data/gradyear.xlsx
    - https://www.ncaa.org/sports/2016/12/14/shared-ncaa-research-data.aspx

Most datasets were able to be loaded with the pandas library directly from the web. For the 'Grad' datasets, this was downloaded directly from the 'Shared NCAA Research Data' site in excel format.

Data Cleaning

In the 'Data Cleaning' phase, each dataset was inspected and transformed as needed.

Coaches

In looking at the 'Coaches' dataset, the .info() method was invoked and provided the following results:

```
# Overview of coaches_df dataset

coaches_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129 entries, 0 to 128
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   School        129 non-null    object
 1   Conference    129 non-null    object
 2   Coach         129 non-null    object
 3   SchoolPay     129 non-null    object
 4   TotalPay      129 non-null    object
 5   Bonus         129 non-null    object
 6   BonusPaid     129 non-null    object
 7   AssistantPay  129 non-null    object
 8   Buyout        129 non-null    object
dtypes: object(9)
memory usage: 9.2+ KB
```

This method allowed me to inspect datatypes as well as any null-values; it also provides an overview of the shape of the dataframe – calling out the number of rows and columns, respectively.

Although no nulls were technically found, the .head() method gave insight into a null-like value: the "--".

```
coaches_df.head(20)
```

	School	Conference	Coach	SchoolPay	TotalPay	Bonus	BonusPaid	AssistantPay	Buyout
0	Air Force	Mt. West	Troy Calhoun	885000	885000	247000	--	$0	--
1	Akron	MAC	Terry Bowden	$411,000	$412,500	$225,000	$50,000	$0	$688,500
2	Alabama	SEC	Nick Saban	$8,307,000	$8,307,000	$1,100,000	$500,000	$0	$33,600,000
3	Alabama at Birmingham	C-USA	Bill Clark	$900,000	$900,000	$950,000	$165,471	$0	$3,847,500
4	Appalachian State	Sun Belt	Scott Satterfield	$712,500	$712,500	$295,000	$145,000	$0	$2,160,417
5	Arizona	Pac-12	Kevin Sumlin	$1,600,000	$2,000,000	$2,025,000	--	$0	$10,000,000
6	Arizona State	Pac-12	Herm Edwards	$2,000,000	$2,000,000	$3,010,000	--	$0	$8,166,667
7	Arkansas	SEC	Chad Morris	$3,500,000	$3,500,000	$1,000,000	--	$0	$12,500,000

As a result, the "--" values were converted to null values; the null values were then replaced by zeros, and zeros were replaced with the mean of the respective numerical column.

The datatypes were reset for all numerical columns, converting from 'object' to 'float.'

The ending result looked as follows:

```
# Clean coaches_df
coaches_df.replace('Pac-12', 'Pac 12', inplace=True)
coaches_df.replace('--', 0, inplace=True)

replace_dict = {col: str for col in ['School', 'Conference', 'Coach']}
coaches_df = coaches_df.astype(replace_dict).replace(',', '', regex=True)

money_columns = ['SchoolPay', 'TotalPay', 'Bonus', 'BonusPaid', 'AssistantPay', 'Buyout']
for col in money_columns:
    coaches_df[col] = coaches_df[col].str.replace('[^0-9.]', '', regex=True).astype(float)

coaches_df.fillna(coaches_df.mean(), inplace=True)
coaches_df.head(20)
```

```
<ipython-input-170-8df8af9d99cb>:12: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a
  coaches_df.fillna(coaches_df.mean(), inplace=True)
```

	School	Conference	Coach	SchoolPay	TotalPay	Bonus	BonusPaid	AssistantPay	Buyout
0	Air Force	Mt. West	Troy Calhoun	885000.000	885000.00	2.470000e+05	149524.295455	0.0	8.119107e+06
1	Akron	MAC	Terry Bowden	411000.000	412500.00	2.250000e+05	50000.000000	0.0	6.885000e+05
2	Alabama	SEC	Nick Saban	8307000.000	8307000.00	1.100000e+06	500000.000000	0.0	3.360000e+07
3	Alabama at Birmingham	C-USA	Bill Clark	900000.000	900000.00	9.500000e+05	165471.000000	0.0	3.847500e+06
4	Appalachian State	Sun Belt	Scott Satterfield	712500.000	712500.00	2.950000e+05	145000.000000	0.0	2.160417e+06
5	Arizona	Pac 12	Kevin Sumlin	1600000.000	2000000.00	2.025000e+06	149524.295455	0.0	1.000000e+07
6	Arizona State	Pac 12	Herm Edwards	2000000.000	2000000.00	3.010000e+06	149524.295455	0.0	8.166667e+06
7	Arkansas	SEC	Chad Morris	3500000.000	3500000.00	1.000000e+06	149524.295455	0.0	1.250000e+07
8	Arkansas State	Sun Belt	Blake Anderson	825000.000	825000.00	1.850000e+05	25000.000000	0.0	3.000000e+05
9	Army	Ind.	Jeff Monken	932521.000	932521.00	8.741782e+05	149524.295455	0.0	8.119107e+06

Stadiums

The Stadiums dataset was then inspected. Similarly, the .info() method was invoked to see an overview:

Inspect Stadiums dataset

```
stadiums = stadiums[0]
stadiums.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 131 entries, 0 to 130
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Stadium     131 non-null    object
 1   College     131 non-null    object
 2   Conference  131 non-null    object
 3   Capacity    131 non-null    int64
 4   Opened      131 non-null    int64
dtypes: int64(2), object(3)
memory usage: 5.2+ KB
```

No null or null-like values were observed within this dataset. The datatypes also appeared to be satisfactory. In hopes to eventually merge all datasets together, the 'College' column was renamed to 'School' to conform to the 'Coaches' dataset. The number of columns were then reduced, as the actual Stadium names (as well as year Opened) were not needed for the analysis. The final dataset appeared as this:

```
stadiums.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 131 entries, 0 to 130
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   School      131 non-null    object
 1   Conference  131 non-null    object
 2   Capacity    131 non-null    int64
dtypes: int64(1), object(2)
memory usage: 3.2+ KB
```

An import sidenote to introduce: School names appeared to have variations in spelling or abbreviation. This was not immediately transformed manually. Instead, a 'fuzzy match' that attempts to merge school names based on similarity score was used later in the analysis.

Team Record

As for the team record dataset, no observed null or null-like values were detected; however, the datatypes for winning percentage appeared to be incorrect. In addition to this, more columns were determined to not be required for the analysis, and were thus dropped. A before and after view can be seen in the figures below. In brief, the ATS, MOV, and Win-Loss Record were dropped, as 'Win %' was chosen as the main statistic for potential use as an independent variable in the model.

Before:

Inspect teamrecord dataset

```
[ ]  teamrecord = teamrecord[0]
     teamrecord.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 131 entries, 0 to 130
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Team             131 non-null    object
 1   Win-Loss Record  131 non-null    object
 2   Win %            131 non-null    object
 3   MOV              131 non-null    float64
 4   ATS +/-          131 non-null    float64
dtypes: float64(2), object(3)
memory usage: 5.2+ KB
```

After:

```
teamrecord['Win %'] = teamrecord['Win %'].str.replace('%', '').astype(float)
teamrecord = teamrecord[['Team', 'Win %']]
teamrecord.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 131 entries, 0 to 130
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Team    131 non-null    object
 1   Win %   131 non-null    float64
dtypes: float64(1), object(1)
memory usage: 2.2+ KB
```

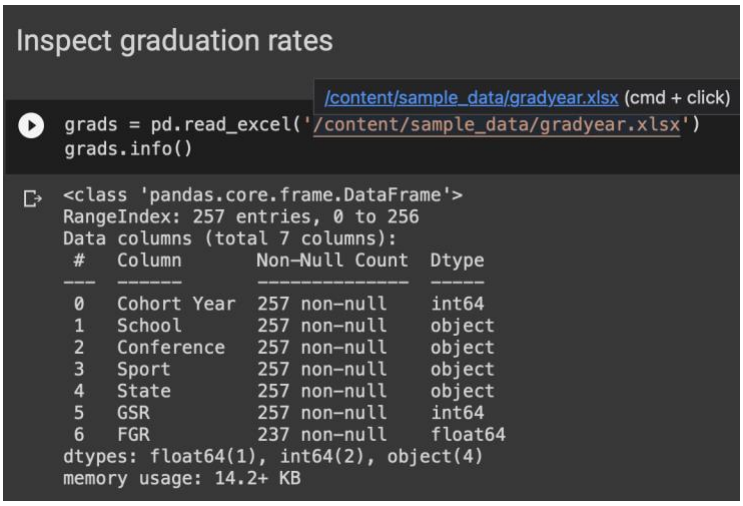
```
teamrecord['Team'].value_counts().to_csv('teamrecord_schools.csv')
```

```
teamrecord = teamrecord.rename(columns = {'Team' : 'School'})
```

An import sidenote to introduce: School names appeared to have variations in spelling or abbreviation. This was not immediately transformed manually. Instead, a 'fuzzy match' that attempts to merge school names based on similarity score was used later in the analysis.

<u>Grades</u>

The final dataset used in the analysis included data on graduation rates. Similarly to the previous datasets, the .info(), .head(), and other methods were used to gather an overview and preview the data.



This time, null-values were detected in the FGR column name. These values were dropped and the datatypes were not modified. The ending results looks as follows:

```
grads = grads.dropna()
grads.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 237 entries, 0 to 256
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Cohort Year  237 non-null    int64
 1   School       237 non-null    object
 2   Conference   237 non-null    object
 3   Sport        237 non-null    object
 4   State        237 non-null    object
 5   GSR          237 non-null    int64
 6   FGR          237 non-null    float64
dtypes: float64(1), int64(2), object(4)
memory usage: 14.8+ KB
```

Merging into One, Overall Dataset

Since the goal of the datasets is to use everything in one analysis, a merging of the various data frames will be required for the next phase in the analysis. Rather than manually transforming each school abbreviation through code, a python library called 'fuzzywuzzy' was used to use a 'partial match' based on similarity. This approach is not always 100% accurate, but was chosen as an exploratory step in this analysis lab assignment.

The process for the partial matching is as follows:

- Derive a 'best match' determined by what's known as the Levenshtein Distance, which is a measure of the difference between two strings
    - Each string would be the 'School' attribute instance in each data frame

- If a match is found, the school name found in the ‘Coaches’ dataset is replaced by the school names found in the Stadiums, Team Record, and Grads datasets

The end results is the following:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 82 entries, 0 to 441
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   School        82 non-null     object
 1   Coach         82 non-null     object
 2   SchoolPay     82 non-null     float64
 3   TotalPay      82 non-null     float64
 4   Bonus         82 non-null     float64
 5   BonusPaid     82 non-null     float64
 6   AssistantPay  82 non-null     float64
 7   Buyout        82 non-null     float64
 8   Capacity      82 non-null     float64
 9   Win %         82 non-null     float64
 10  Conference    82 non-null     object
 11  Sport         82 non-null     object
 12  State         82 non-null     object
 13  GSR           82 non-null     int64
 14  FGR           82 non-null     float64
dtypes: float64(9), int64(1), object(5)
memory usage: 10.2+ KB
```

The partial matching via the ‘fuzzywuzzy’ python library was not completed with high data retention, as the amount of observations was reduced from ~130 to just ~80. Although this is not an ideal occurrence, it is believed that a reliable model can still be derived based on the final results of this data frame.

All data types in the final data frame are either object, float, or integer.

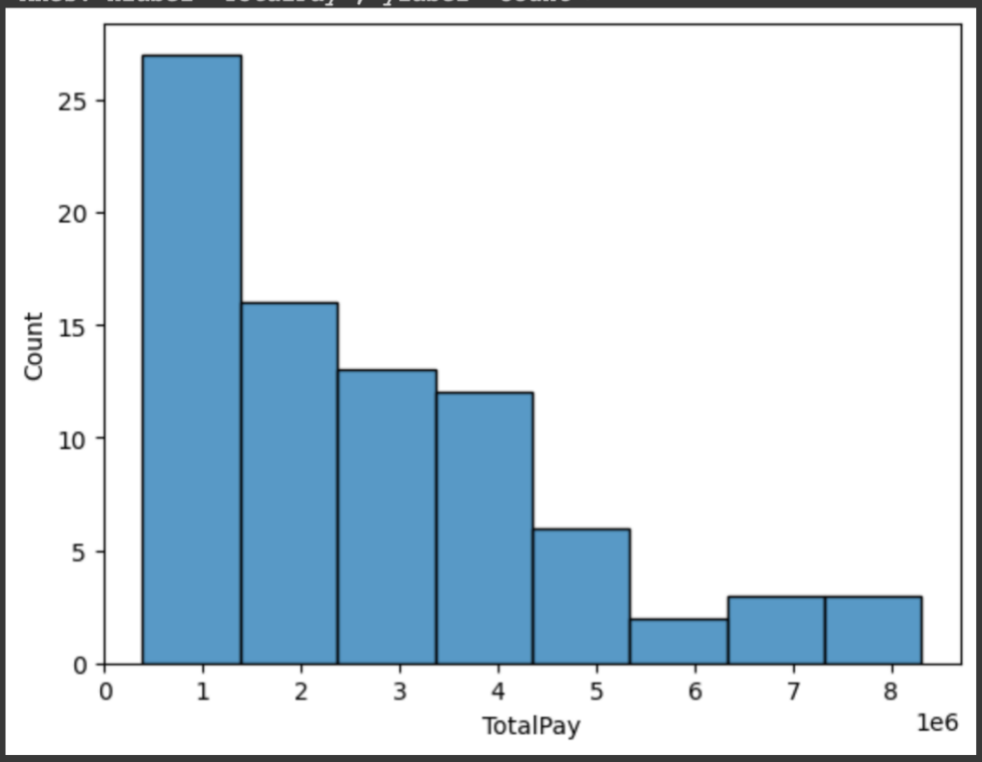
Data Exploration

Now that a final data frame was derived based on the various data sources, exploration can begin.

The exploration is started by checking the distribution of the dependent variable, ‘TotalPay’:

```
# View Distribution of TotalPay
sns.histplot(df['TotalPay'])
```

```
<Axes: xlabel='TotalPay', ylabel='Count'>
```



This looks to be positively skewed, since many of the observations are closer to zero, but a proverbial 'tail' is viewed trailing towards larger values. Some would argue taking the logarithm of this variable to convert the distribution type to 'normal'; however, for the purposes of this exploratory lab, we will see if this negatively affects the results of our model.

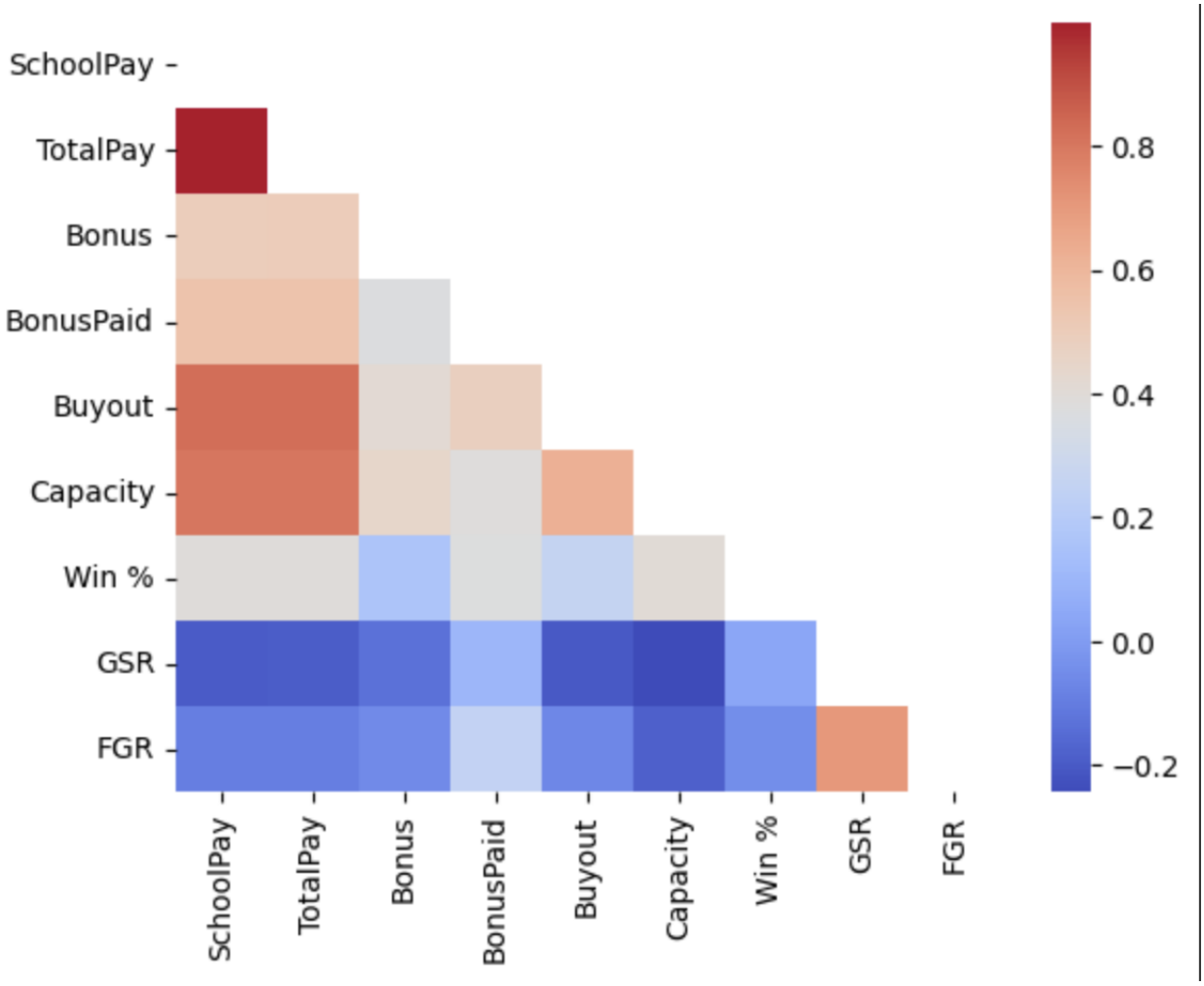
Next, before checking the distributions of independent variables, we will gather correlation insights by deriving a matrix along with a heatmap (provided via the seaborn python library).

```
import numpy as np

# Explore the data
correlations = df.corr()

# Remove The AssistantPay column from the correlations DataFrame
correlations = correlations.drop('AssistantPay', axis=0).drop('AssistantPay', axis=1)

# Zero out the lower triangle of the matrix
mask = np.triu(np.ones_like(correlations, dtype=bool))
correlations = correlations.mask(mask)

correlations
```

```
<ipython-input-196-86c7bff2e180>:4: FutureWarning: The default value of numeric_only in DataFra
  correlations = df.corr()
```

	SchoolPay	TotalPay	Bonus	BonusPaid	Buyout	Capacity	Win %	GSR	FGR
SchoolPay	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
TotalPay	0.999566	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Bonus	0.496905	0.501814	NaN	NaN	NaN	NaN	NaN	NaN	NaN
BonusPaid	0.545730	0.547483	0.370240	NaN	NaN	NaN	NaN	NaN	NaN
Buyout	0.826087	0.828953	0.412215	0.483947	NaN	NaN	NaN	NaN	NaN
Capacity	0.802511	0.802756	0.444124	0.385656	0.622141	NaN	NaN	NaN	NaN
Win %	0.390897	0.390198	0.162289	0.373447	0.257871	0.406655	NaN	NaN	NaN
GSR	-0.197069	-0.193242	-0.133699	0.097298	-0.204554	-0.243810	0.033822	NaN	NaN
FGR	-0.094980	-0.094176	-0.057573	0.250936	-0.065918	-0.184339	-0.044788	0.708121	NaN

Although this is helpful, a visualization would aid in a quicker analysis:

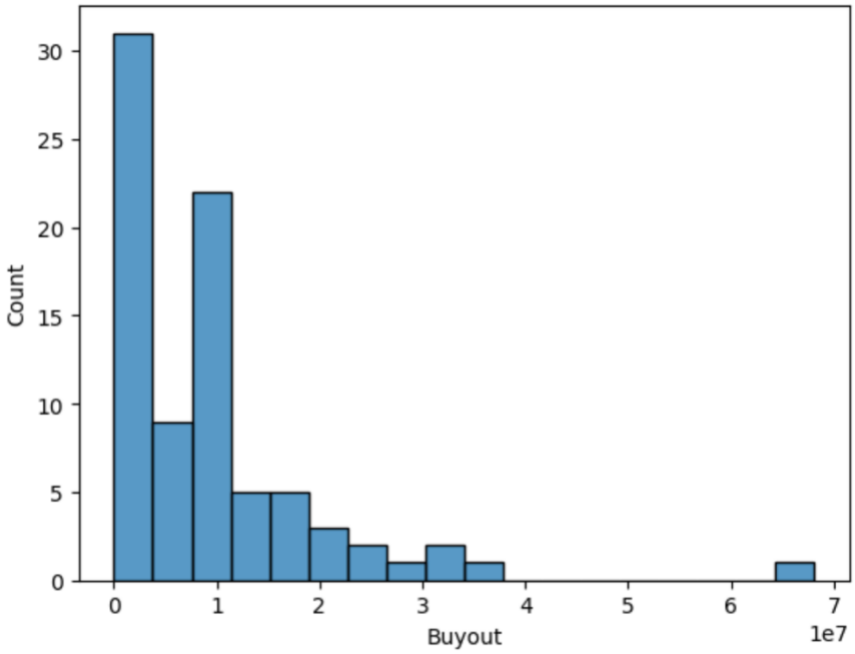


Based on the correlation matrix visualization, the following independent variables will be trialed for the modeling portion:

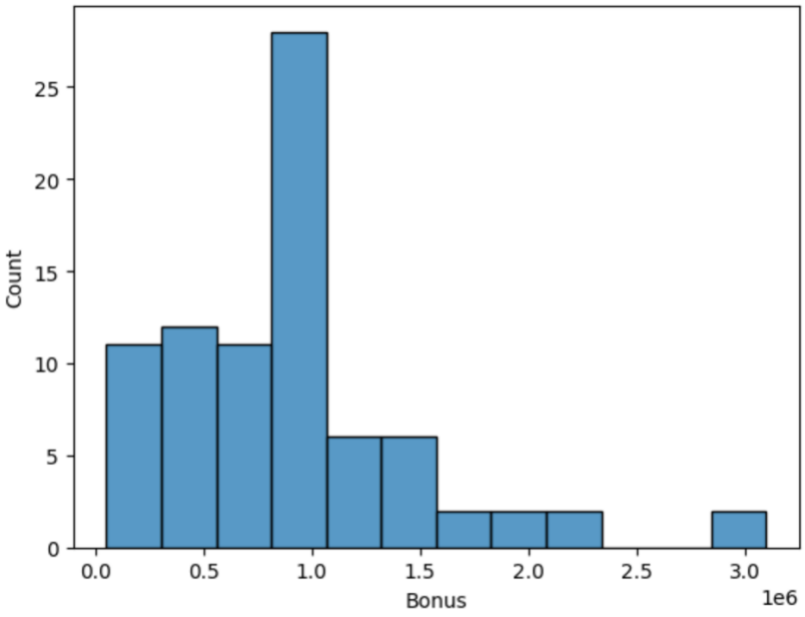
- Buyout
- Capacity
- Bonus
- BonusPaid
- Win %
- GSR
    - Chosen based on lab question set
- FGR
    - Chosen based on lab question set

Histograms were then derived for each independent variable chosen for the modeling of this data:

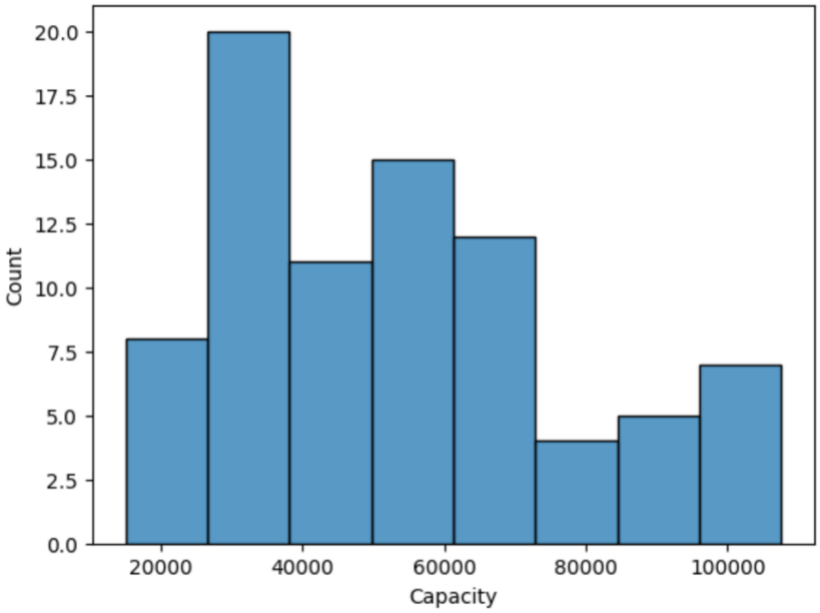
Buyout:



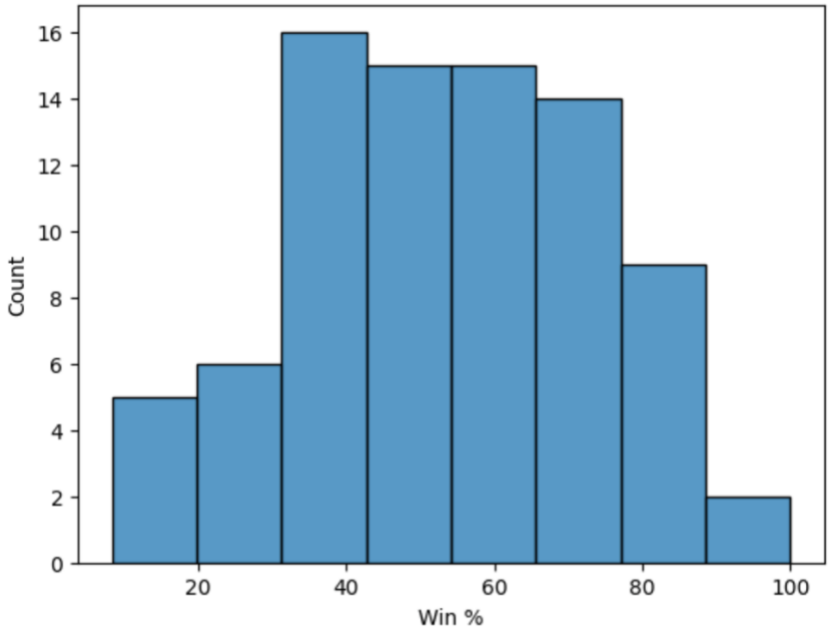
BonusPaid



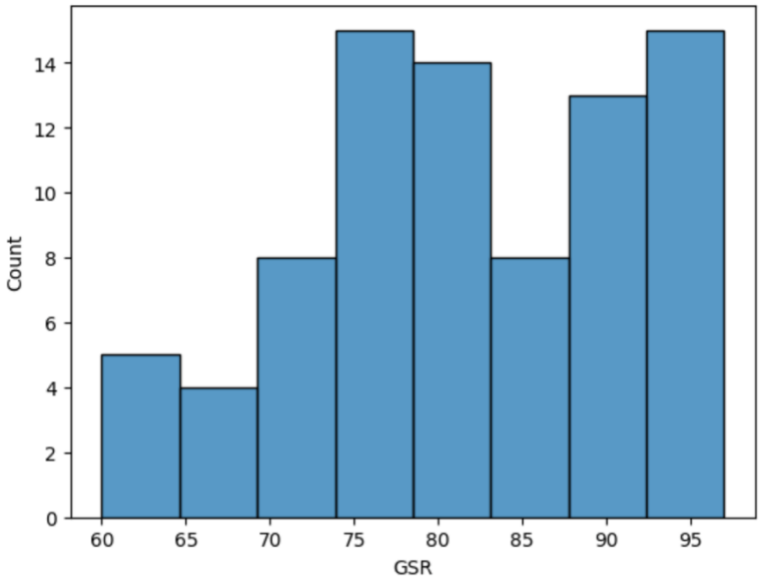
Capacity



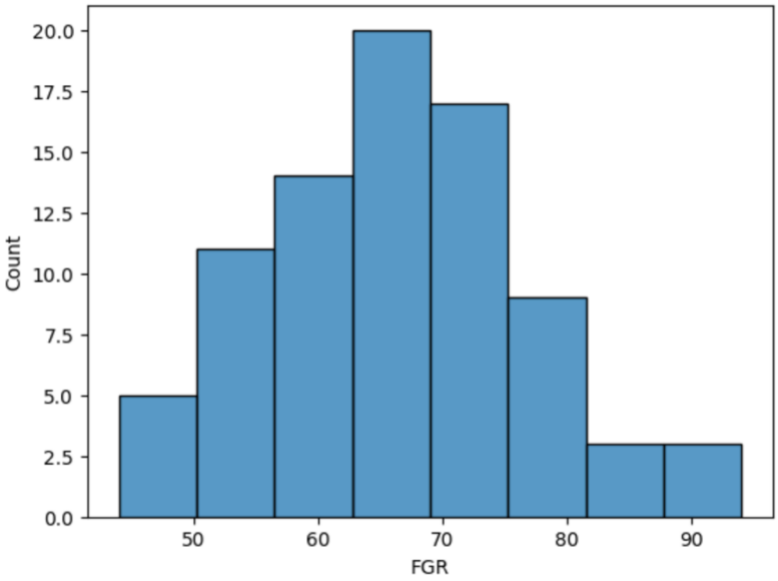
Win %



GSR

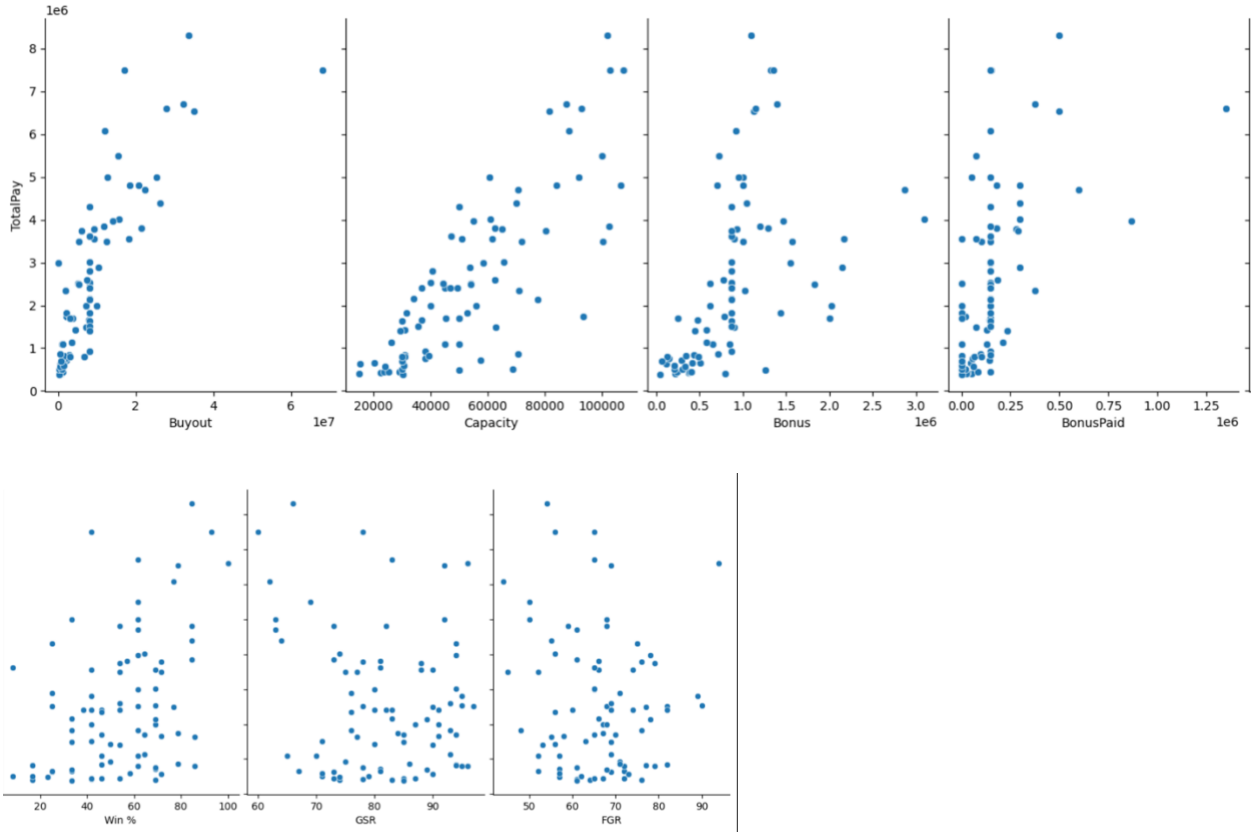


FGR



To summarize, we generally see positively skewed or normal distributions for the independent variables chosen – this could affect the quality of our results in the modeling phase. Modifying some of these distributions with the logarithmic equivalent could improve accuracy.

Last but not least, a pairplot was conducted, also with the seaborn package, to see the relationship between the Y and X variables:



Visually speaking, the GSR and FGR data columns from the 'Grads' dataset do not appear as though they have a strong relationship with the Y variable; however, since a question related to these data points is specifically in the lab 1 questions, they will be kept in the modeling analysis.

Data Modeling

As mentioned in the Overview section of this document, the Ordinary Least Squares (OLS) is a method for estimating unknown parameters in a linear regression model (i.e., prediction data). This modeling technique will be used to ultimately determine the salary Syracuse should offer their next football coach.

```python
# Model without log variables

X = df[['Buyout', 'Capacity','Bonus','BonusPaid','Win %','GSR','FGR']]
y = df['TotalPay']

# Add a constant term to the predictor variables (X)
X1 = sm.add_constant(X)

# Create the linear model and fit it to the data
model = sm.OLS(y, X).fit()

# Print the model summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               TotalPay   R-squared (uncentered):                   0.942
Model:                           OLS   Adj. R-squared (uncentered):              0.936
Method:                Least Squares   F-statistic:                              173.4
Date:               Mon, 17 Apr 2023   Prob (F-statistic):                    1.22e-43
Time:                       22:48:04   Log-Likelihood:                         -1228.2
No. Observations:                 82   AIC:                                      2470.
Df Residuals:                     75   BIC:                                      2487.
Df Model:                          7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Buyout         0.0876      0.012      7.443      0.000       0.064       0.111
Capacity      31.6866      5.009      6.325      0.000      21.707      41.666
Bonus          0.2265      0.178      1.269      0.208      -0.129       0.582
BonusPaid      1.3174      0.573      2.298      0.024       0.176       2.459
Win %       3479.3999   5010.850      0.694      0.490   -6502.725    1.35e+04
GSR        -1664.6599   1.14e+04     -0.146      0.884   -2.44e+04    2.1e+04
FGR        -6119.9941   1.32e+04     -0.464      0.644   -3.24e+04    2.02e+04
==============================================================================
Omnibus:                        9.985   Durbin-Watson:                   2.470
Prob(Omnibus):                  0.007   Jarque-Bera (JB):               10.743
Skew:                           0.658   Prob(JB):                      0.00465
Kurtosis:                       4.189   Cond. No.                     2.69e+06
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 2.69e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Conclusions and insights from this modeling technique will be discussed in the next section.

Conclusions

Insights from this model can be derived by reviewing the following:

First, the probability of the F-statistic should be reviewed. Since this value is well below the threshold of 0.05, the model can be interpreted and is statistically significant.

Next, the R-squared value can be reviewed. The value of 0.942 indicates that the ~94% of the change in Y, our TotalPay variable, can be explained by the changes in our X variables – Buyout, Capacity, Bonus, BonusPaid, Win%, FGR, and GSR. This is generally a good sign and shows that we may be able to make a 'good' prediction with this model.

Following this, checking the validity of the variable coefficients is important. From our model output, it can be viewed that many of the p-value coefficients are above the 0.05 threshold, meaning that they are not statistically significant to the model. More specifically, the Bonus, Win %, GSR, and FGR all have p-values above this threshold. We cannot conclude that these specific variables have any significance to our model. Buyout, Capacity, and BonusPaid appear to be statistically significant, with p-values below the .05 threshold.

The largest statistically significant coefficient (and coefficient in general) is the capacity variable, which indicates a coaches pay increases ~$31 for each unit increase in a stadium's capacity.

Lab Questions

Lab Question 1

What is the recommended salary for the Syracuse football coach?

To answer this question, a prediction will need to be made with the model developed in the 'Data Modeling' section.

```
[ ]  # Create a dictionary of data for Syracuse
     syracuse_data = {
         #'const': 1,
         'Buyout': np.mean(df['Buyout']),
         'Capacity': np.mean(df['Capacity']),
         'Bonus': np.mean(df['Bonus']),
         'BonusPaid': np.mean(df['BonusPaid']),
         'Win %': np.mean(df['Win %']),
         'GSR': np.mean(df['GSR']),
         'FGR': np.mean(df['FGR'])
     }

     # Convert the dictionary to a DataFrame
     syracuse_df = pd.DataFrame(syracuse_data, index=[0])

     # Predict the salary
     predicted_salary = model.predict(syracuse_df)
     formatted_salary = "${:,.2f}".format(round(predicted_salary[0], 2))
     print(f"Predicted salary for Syracuse football coach: {formatted_salary}")

     Predicted salary for Syracuse football coach: $2,559,590.59
```

Using the model, the predicted salary for the new Syracuse football coach is ~$2.5 million.

Lab Question 2

What would his salary be if we were still in the Big East? What if we went to the Big Ten?

A prediction for each conference in the dataset was made into a table:

```
# Convert the dictionary to a DataFrame
predicted_salaries_df = pd.DataFrame.from_dict(predicted_salaries, orient='index', columns=['Predicted Salary'])

predicted_salaries_df
```

	Predicted Salary
Mid-American Conference	$752,670.93
Southwestern Athletic Conf.	$5,492,733.15
Sun Belt Conference	$2,176,338.17
Pac-12 Conference	$2,708,739.59
Colonial Athletic Association	$1,448,125.21
Southeastern Conference	$3,638,213.16
Big 12 Conference	$2,870,144.41
Mountain West Conference	$1,673,217.05
Atlantic Coast Conference	$3,076,676.05
Independent	$2,310,981.73
Big Sky Conference	$3,058,507.16
American Athletic Conference	$1,901,983.18
Conference USA	$869,343.94
Northeast Conference	$3,407,039.85
Patriot League	$2,891,690.83
Southland Conference	$2,588,256.42
Ohio Valley Conference	$2,995,872.54
Missouri Valley Football Conference	$1,755,513.39
Mid-Eastern Athletic Conf.	$5,314,384.73
ASUN Conference	$4,082,832.13
Big South Conference	$3,501,382.20
Big Ten Conference	$3,662,141.69
Southern Conference	$2,655,541.60

Although the Big East does not seem to be included in the dataset, the Big 10 predicted salary appears to be ~$3.6 million, which is significantly higher.

Lab Question 3

What schools did we drop from our data and why?

As mentioned earlier in the report, some school names were dropped due to the ‘fuzzywuzzy’ python library partial match function. Any school that was not able to be detected by the similarity partial match functionality of this package was dropped.

All in all, 82 schools/observations were kept in the dataset. The schools that were dropped were the following:

```
print(len(dropped_schools))
dropped_schools
```

```
47
{'Air Force',
 'Alabama at Birmingham',
 'Arizona State',
 'Arkansas State',
 'Colorado State',
 'Eastern Michigan',
 'Florida Atlantic',
 'Florida International',
 'Florida State',
 'Fresno State',
 'Georgia Southern',
 'Georgia State',
 'Georgia Tech',
 'Iowa State',
 'Kansas State',
 'Liberty',
 'Louisiana Tech',
 'Miami (Fla.)',
 'Miami (Ohio)',
 'Michigan State',
 'Mississippi State',
 'Nevada-Las Vegas',
 'New Mexico State',
 'North Carolina',
 'North Carolina State',
 'Northern Illinois',
 'Ohio State',
 'Oklahoma State',
 'Oregon State',
 'Rutgers',
 'South Alabama',
 'South Carolina',
 'South Florida',
 'Southern California',
 'Southern Methodist',
 'Southern Mississippi',
 'Texas Christian',
 'Texas Tech',
 'Texas-El Paso',
 'Texas-San Antonio',
 'UCLA',
 'Utah State',
 'Virginia Tech',
 'Washington State',
 'West Virginia',
 'Western Kentucky',
 'Western Michigan'}
```

More details can be found in the supporting .ipynb (Jupyter Notebook) file on the 2SU submission page.

Lab Question 4

What effect does graduation rate have on the projected salary?

Based on the previously mentioned p-values for the graduation rate data (GSR, FGR), we cannot conclude these variables are statistically significant, since they are well above the 0.05 threshold – coming in at ~0.8 and ~0.6, respectively. We cannot confirm whether this data influences projected salary.

Lab Question 5

How good is our model?

As previously mentioned, the R-squared value for this model is ~0.94. This means that the ~94% of the change in Y, our TotalPay variable, can be explained by the changes in our X variables – Buyout, Capacity, Bonus, BonusPaid, Win%, FGR, and GSR.

It is a relatively high score, and indicates the variation in the data is explained (mostly) by these variables.

Lab Question 6

What is the single biggest impact on salary size?

The largest impact on salary size is the Capacity variable. As mentioned in the conclusion section, the variable's p-value is statistically significant and the coefficient value is ~31, meaning a coaches pay increases ~$31 for each unit increase in a stadium's capacity.