DAY 3

LEARNING VOYAGE

5: Data Cleaning and Exploratory Data Analysis

LEARNING VOYAGE

Data Cleaning and Data Wrangling

- Data cleaning or data cleansing is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record of set, table or database.

- The process on the raw data that makes it “clean” enough to input to our analytical algorithm is called data wrangling

“Understand the Data“ (Review the Data and the Data Dictionary)

Imputing Missing Values

- we need to subset the categorical and numerical variables for the imputation process.

```
numeric_subset = data.select_dtypes('number')
categorical_subset = data.select_dtypes('object')
```

Imputing Missing Values

```
from sklearn.impute import SimpleImputer
```

```
# Create an imputer object with a median filling strategy
num_imputer = SimpleImputer(strategy='median')

num_imputer.fit(numeric_subset)

num_data = num_imputer.transform(numeric_subset)
```

```
# Create an imputer object with a mode filling strategy
cat_imputer = SimpleImputer(strategy='most_frequent')

cat_imputer.fit(categorical_subset)

cat_data = cat_imputer.transform(categorical_subset)
```

Imputing Missing Values

```
num_df = pd.DataFrame(num_data, columns =  numeric_subset.columns)

cat_df = pd.DataFrame(cat_data, columns = categorical_subset.columns)

mod_data = pd.concat([num_df, cat_df], axis=1)
mod_data.head()
```

We concatenated the data and saved it as modified data set.

Now let's check the presence of any null values in the data.

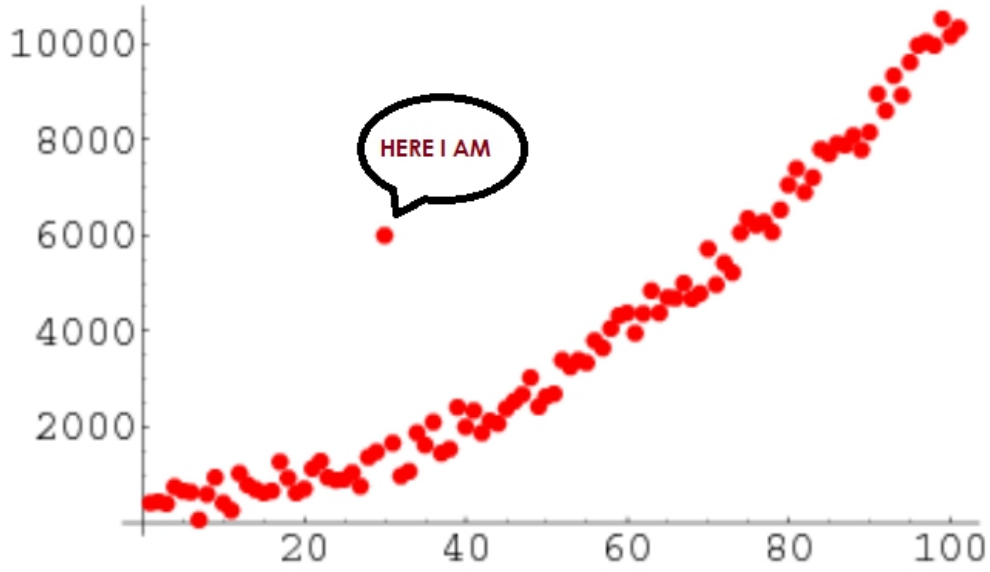
```
mod_data.isnull().sum().sum()
```

```
0
```

We can see that there are no null values present in the data set.

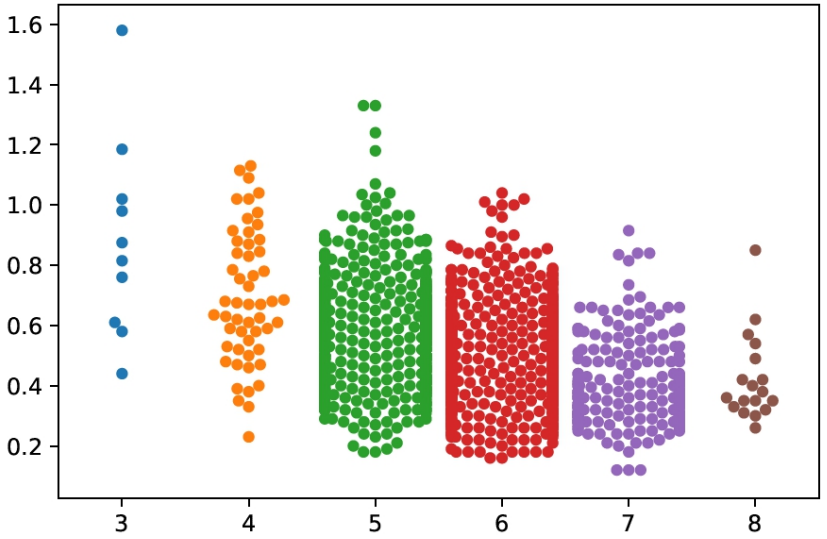
Outliers

- Outliers are the observations that lie at an abnormal distance from the other values in a random sample taken from a population.



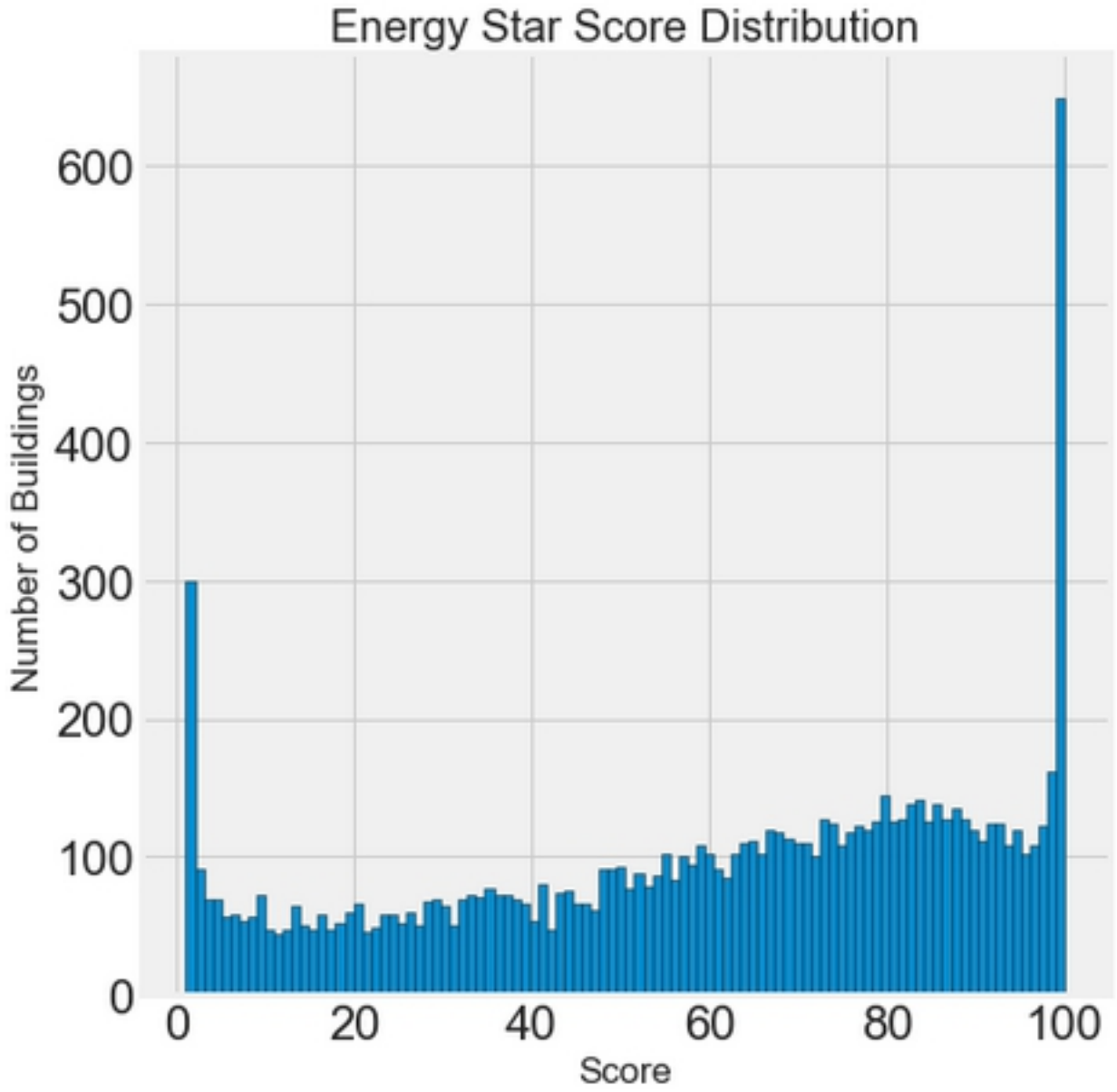
Exploratory Data Analysis (EDA)

- Exploratory data analysis is an open-ended process, where we perform statistics and make figures to find trends, anomalies, patterns or relationships within the given data.



Univariate analysis (Single variable plots)

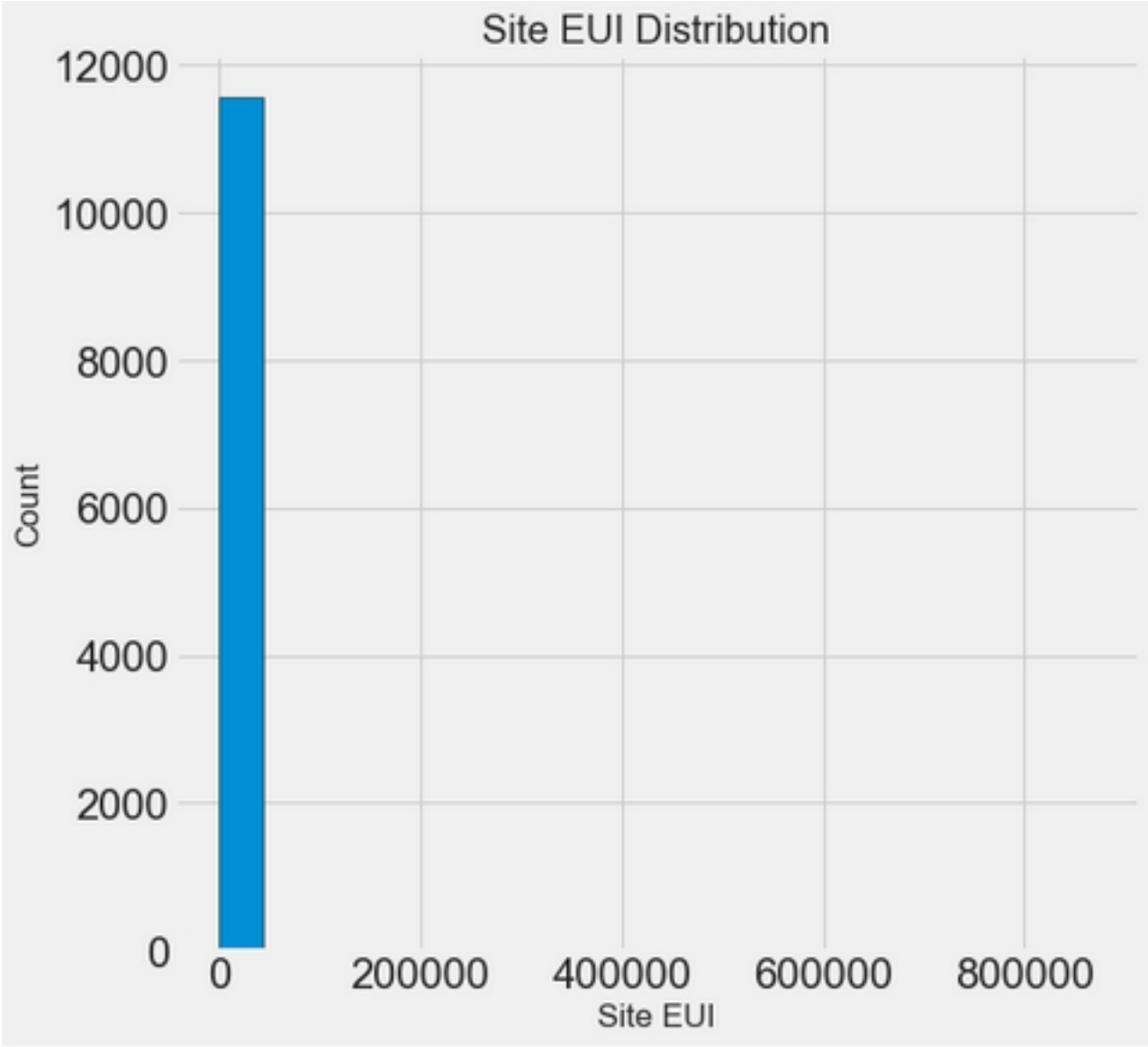
```
plt.figure(figsize=(8,8))

# Rename the socre
data = data.rename(columns = {'ENERGY STAR Score' : 'Score'})

# Histogram of the Energy Star Score
plt.style.use('fivethirtyeight')
plt.hist(data['Score'].dropna(), bins = 100, edgecolor = 'k')
plt.xlabel('Score')
plt.ylabel('Number of Buildings')
plt.title('Energy Star Score Distribution')
```


Energy Star Score Distribution
Number of Buildings
0
100
200
300
400
500
600
0
20
40
60
80
100
Score

Univariate analysis (Single variable plots)

- Let us look at the distribution of the ‘site EUI’ variable.

```
# Histogram plot of site EUI
plt.figure(figsize=(8,8))
plt.hist(data['Site EUI (kBtu/ft²)'].dropna(),bins=20,edgecolor='black')
plt.xlabel('Site EUI')
plt.ylabel('Count')
plt.title('Site EUI Distribution')
```


Site EUI Distribution
12000
10000
8000
6000
4000
2000
0
Count
0
200000
400000
600000
800000
Site EUI

```
data['Site EUI (kBtu/ft²)'].describe()
```

```
count     11583.000000
mean        280.071484
std        8607.178877
min           0.000000
25%          61.800000
50%          78.500000
75%          97.600000
max      869265.000000
Name: Site EUI (kBtu/ft²), dtype: float64
```

```
data['Site EUI (kBtu/ft²)'].dropna().sort_values(ascending = False).head(10)
```

```
8068    869265.0
7       143974.4
3898    126307.4
8174    112173.6
8268    103562.7
3263     95560.2
8269     84969.6
3383     78360.1
3170     51831.2
3173     51328.8
Name: Site EUI (kBtu/ft²), dtype: float64
```

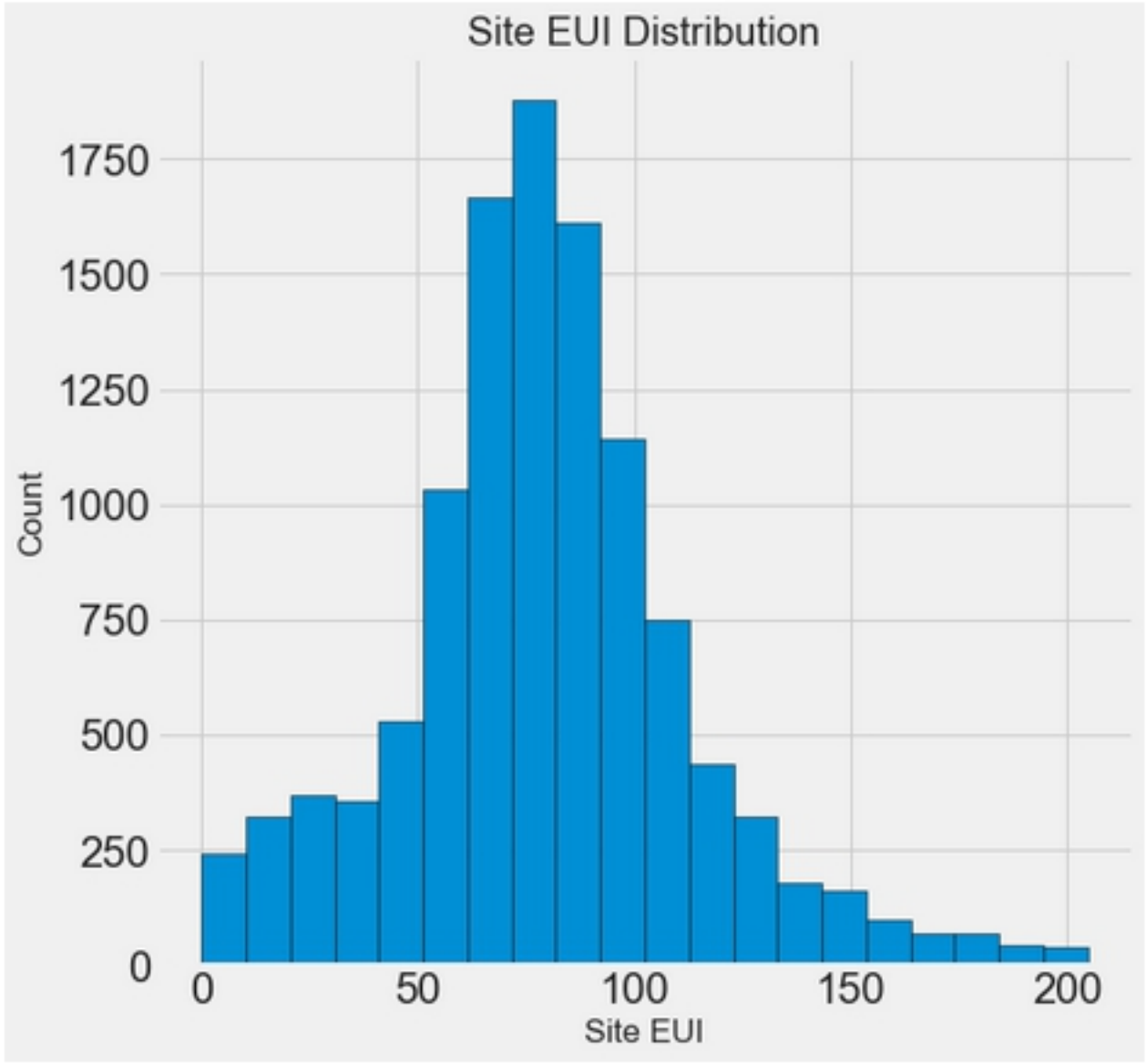
Univariate analysis (Single variable plots)

- Wow! One building is clearly far above the rest

Street Name	Borough	DOF Gross Floor Area	Primary Property Type - Self Selected	List of All Property Use Types at Property	Largest Property Use Type	Largest Property Use Type - Gross Floor Area (ft²)	Year Built	Number of Buildings - Self-reported	Occupancy	Metered Areas (Energy)	Metered Areas (Water)	Score	Site EUI (kBtu/ft²)
SKILLMAN AVENUE	Brooklyn	61811.0	Multifamily Housing	Multifamily Housing	Multifamily Housing	56900.0	2004	1	90	Whole Building	NaN	1.0	869265.0

Removing Outliers

```
# Calculate first and third quartile
first_quantile = data['Site EUI (kBtu/ft²)'].describe()['25%']
third_quantile = data['Site EUI (kBtu/ft²)'].describe()['75%']

# Interquartile range
iqr = third_quantile - first_quantile

#Remove outliers
data = data[(data['Site EUI (kBtu/ft²)'] > (first_quantile - 3 * iqr)) &
            (data['Site EUI (kBtu/ft²)'] < (third_quantile + 3 * iqr))]
```


Site EUI Distribution
Count
1750
1500
1250
1000
750
500
250
0
0
50
100
150
200
Site EUI

Removing Outliers

```
data['Site EUI (kBtu/ft²)'].describe()
```

```
count    11319.000000
mean        79.086377
std         33.317277
min          0.000000
25%         61.200000
50%         77.800000
75%         95.800000
max        204.800000
Name: Site EUI (kBtu/ft²), dtype: float64
```

Bivariate Analysis

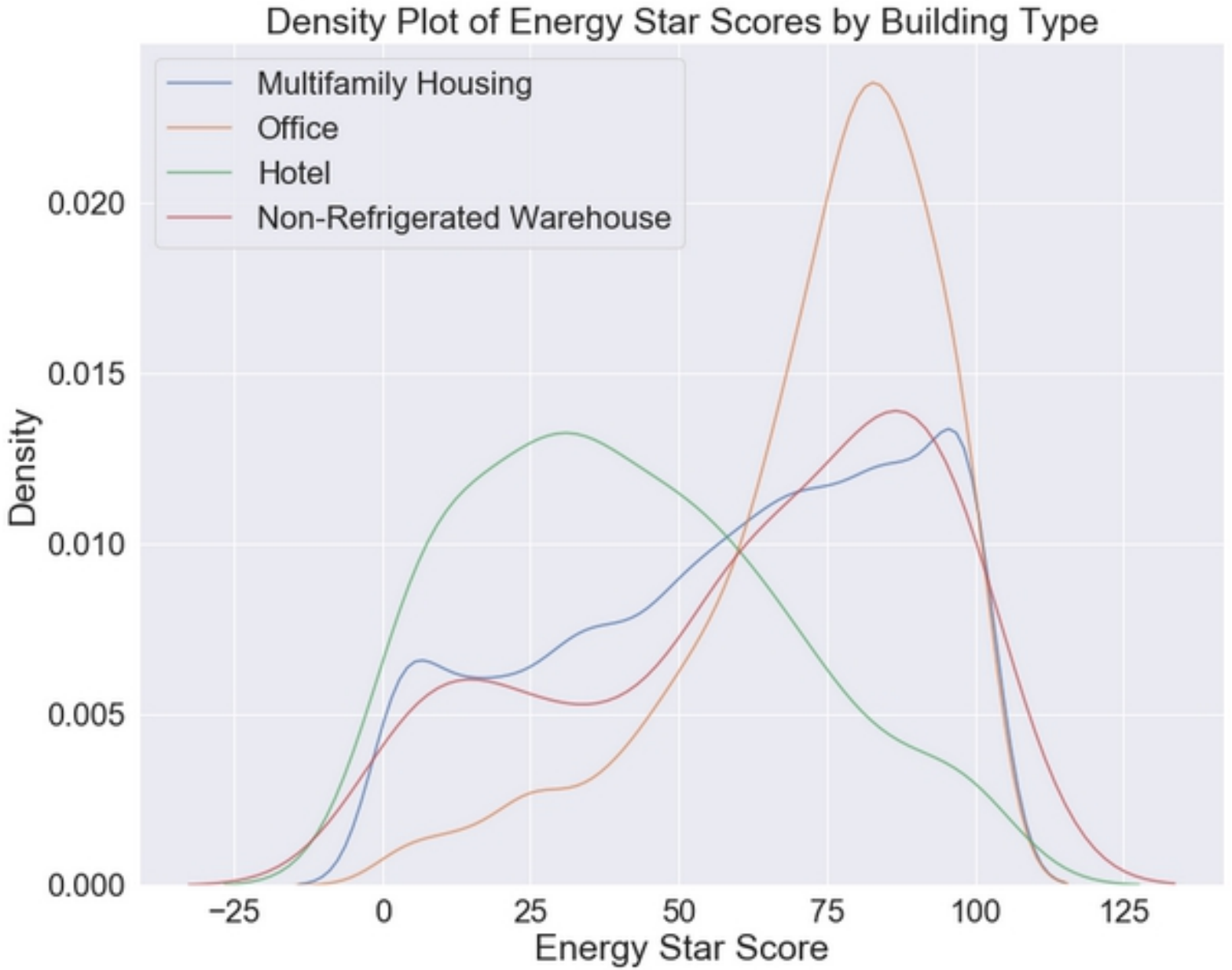
- The first plot we will make shows the distribution of scores by the property type.
- In order to not clutter the plot with large data, we will limit the graph to building types that have more than 100 observations in the dataset.

```
types = data.dropna(subset=['Score'])
types = types['Largest Property Use Type'].value_counts()
types = list(types[types.values > 100].index)
types
```

```
['Multifamily Housing', 'Office', 'Hotel', 'Non-Refrigerated Warehouse']
```

Bivariate Analysis

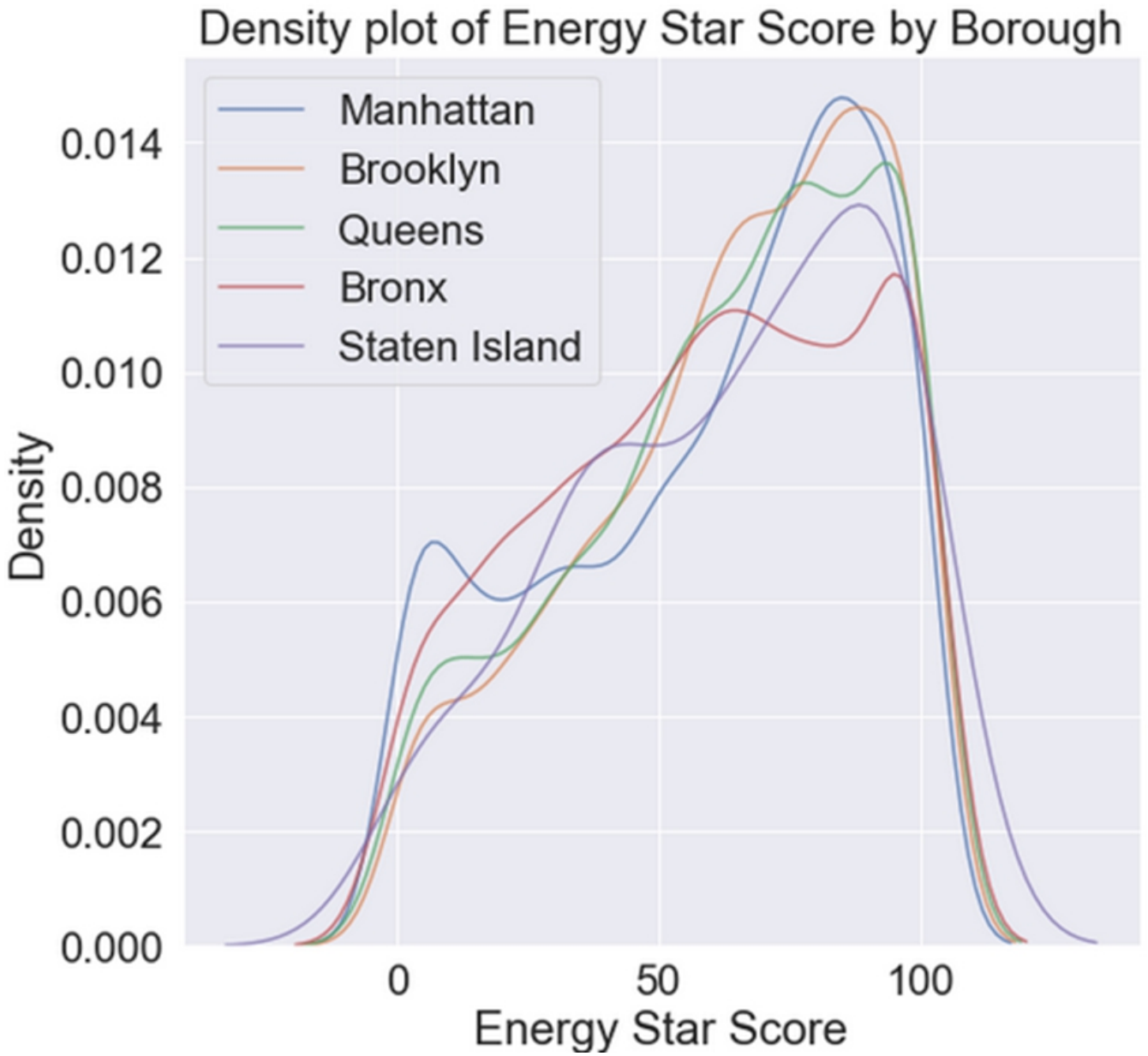
```
# Plot of distribution of scores for building categories

plt.figure(figsize=(12,10))

# plot each building
for b_type in types:
    #select the building type
    subset = data[data['Largest Property Use Type'] == b_type]
    # Density plot of Energy Star Scores
    sns.kdeplot(subset['Score'].dropna(),
                label = b_type, shade = False,
                alpha = 0.8)

# label the plot
plt.xlabel('Energy Star Score', size = 25)
plt.ylabel('Density', size = 25);
plt.title('Density Plot of Energy Star Scores by Building Type', size = 25);
```


Density Plot of Energy Star Scores by Building Type
Multifamily Housing
Office
Hotel
Non-Refrigerated Warehouse
Density
0.020
0.015
0.010
0.005
0.000
−25
0
25
50
75
100
125
Energy Star Score

```
# Create a list of boroughs with more than 100 observations
boroughs = data.dropna(subset=['Score'])
boroughs = boroughs['Borough'].value_counts()
boroughs = list(boroughs[boroughs.values > 100].index)
boroughs
```

```
['Manhattan', 'Brooklyn', 'Queens', 'Bronx', 'Staten Island']
```

```
# Plot of distribution scores of boroughs

plt.figure(figsize=(8,8))

# Plot each borough
for b_borough in boroughs:
    subset = data[data['Borough'] == b_borough]
    sns.kdeplot(subset['Score'].dropna(),
                label = b_borough, shade = False,
                alpha = 0.8)

plt.xlabel("Energy Star Score", size = 5)
plt.ylabel("Density")
plt.title("Density plot of Energy Star Score by Borough")
```


Density plot of Energy Star Score by Borough
Manhattan
Brooklyn
Queens
Bronx
Staten Island
Density
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
Energy Star Score

Correlations between Features and Target

```
# Find all correlations and sort
correlations_data = data.corr()['Score'].sort_values()

# Print the most negative correlations
print(correlations_data.head(15), '\n')

# Print the most positive correlations
print(correlations_data.tail(15))
```

Correlations between Features and Target

Site EUI (kBtu/ft²)	-0.723864
Weather Normalized Site EUI (kBtu/ft²)	-0.713993
Weather Normalized Source EUI (kBtu/ft²)	-0.645542
Source EUI (kBtu/ft²)	-0.641037
Weather Normalized Site Electricity Intensity (kWh/ft²)	-0.358394
Weather Normalized Site Natural Gas Intensity (therms/ft²)	-0.346046
Direct GHG Emissions (Metric Tons CO2e)	-0.147792
Weather Normalized Site Natural Gas Use (therms)	-0.135211
Natural Gas Use (kBtu)	-0.133648
Year Built	-0.121249
Total GHG Emissions (Metric Tons CO2e)	-0.113136
Electricity Use - Grid Purchase (kBtu)	-0.050639
Weather Normalized Site Electricity (kWh)	-0.048207
Latitude	-0.048196
Property Id	-0.046605

Name: Score, dtype: float64

Correlations between Features and Target

```
Property Id                                                -0.046605
Indirect GHG Emissions (Metric Tons CO2e)                  -0.043982
Longitude                                                  -0.037455
Occupancy                                                  -0.033215
Number of Buildings - Self-reported                        -0.022407
Water Use (All Water Sources) (kgal)                       -0.013681
Water Intensity (All Water Sources) (gal/ft²)              -0.012148
Census Tract                                               -0.002299
DOF Gross Floor Area                                        0.013001
Property GFA - Self-Reported (ft²)                          0.017360
Largest Property Use Type - Gross Floor Area (ft²)          0.018330
Order                                                       0.036827
Community Board                                             0.056612
Council District                                            0.061639
Score                                                       1.000000
Name: Score, dtype: float64
```

```
numeric_subset = data.select_dtypes('number')

# Create columns with square root and log of numeric columns
for col in numeric_subset.columns:
    # Skip the Energy Star Score column
    if col == 'Score':
        next
    else:
        numeric_subset['sqrt_' + col] = np.sqrt(numeric_subset[col])
        numeric_subset['log_' + col] = np.log(numeric_subset[col])

# Select the categorical columns
categorical_subset = data[['Borough', 'Largest Property Use Type']]

# One hot encode
categorical_subset = pd.get_dummies(categorical_subset)

# Join the two dataframes using concat
# Make sure to use axis = 1 to perform a column bind
features = pd.concat([numeric_subset, categorical_subset], axis = 1)

# Drop buildings without an energy star score
features = features.dropna(subset = ['Score'])

# Find correlations with the score
correlations = features.corr()['Score'].dropna().sort_values()
```

Correlations between Features and Target

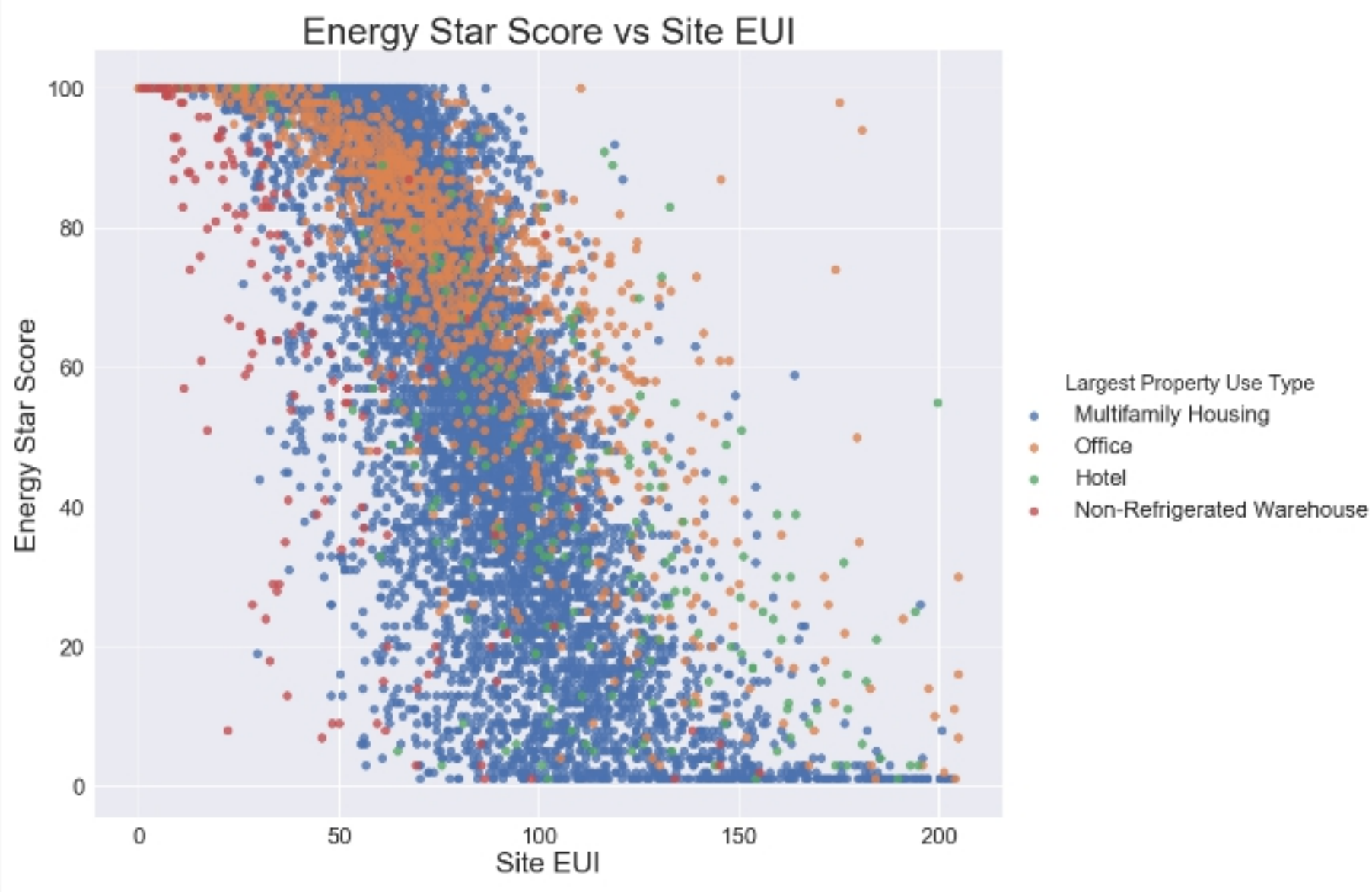
```
# Display most negative correlations
correlations.head(15)
```

```
Site EUI (kBtu/ft²)                                          -0.723864
Weather Normalized Site EUI (kBtu/ft²)                       -0.713993
sqrt_Site EUI (kBtu/ft²)                                     -0.699817
sqrt_Weather Normalized Site EUI (kBtu/ft²)                  -0.689019
sqrt_Weather Normalized Source EUI (kBtu/ft²)                -0.671044
sqrt_Source EUI (kBtu/ft²)                                   -0.669396
Weather Normalized Source EUI (kBtu/ft²)                     -0.645542
Source EUI (kBtu/ft²)                                        -0.641037
log_Source EUI (kBtu/ft²)                                    -0.622892
log_Weather Normalized Source EUI (kBtu/ft²)                 -0.620329
log_Site EUI (kBtu/ft²)                                      -0.612039
log_Weather Normalized Site EUI (kBtu/ft²)                   -0.601332
log_Weather Normalized Site Electricity Intensity (kWh/ft²)  -0.424246
sqrt_Weather Normalized Site Electricity Intensity (kWh/ft²) -0.406669
Weather Normalized Site Electricity Intensity (kWh/ft²)      -0.358394
Name: Score, dtype: float64
```

Correlations between Features and Target

```
# Display most positive correlations
correlations.tail(15)
```

```
sqrt_Order                                                        0.028662
Borough_Queens                                                    0.029545
Largest Property Use Type_Supermarket/Grocery Store               0.030038
Largest Property Use Type_Residence Hall/Dormitory                0.035407
Order                                                             0.036827
Largest Property Use Type_Hospital (General Medical & Surgical)   0.048410
Borough_Brooklyn                                                  0.050486
log_Community Board                                               0.055495
Community Board                                                   0.056612
sqrt_Community Board                                              0.058029
sqrt_Council District                                             0.060623
log_Council District                                              0.061101
Council District                                                  0.061639
Largest Property Use Type_Office                                  0.158484
Score                                                             1.000000
Name: Score, dtype: float64
```

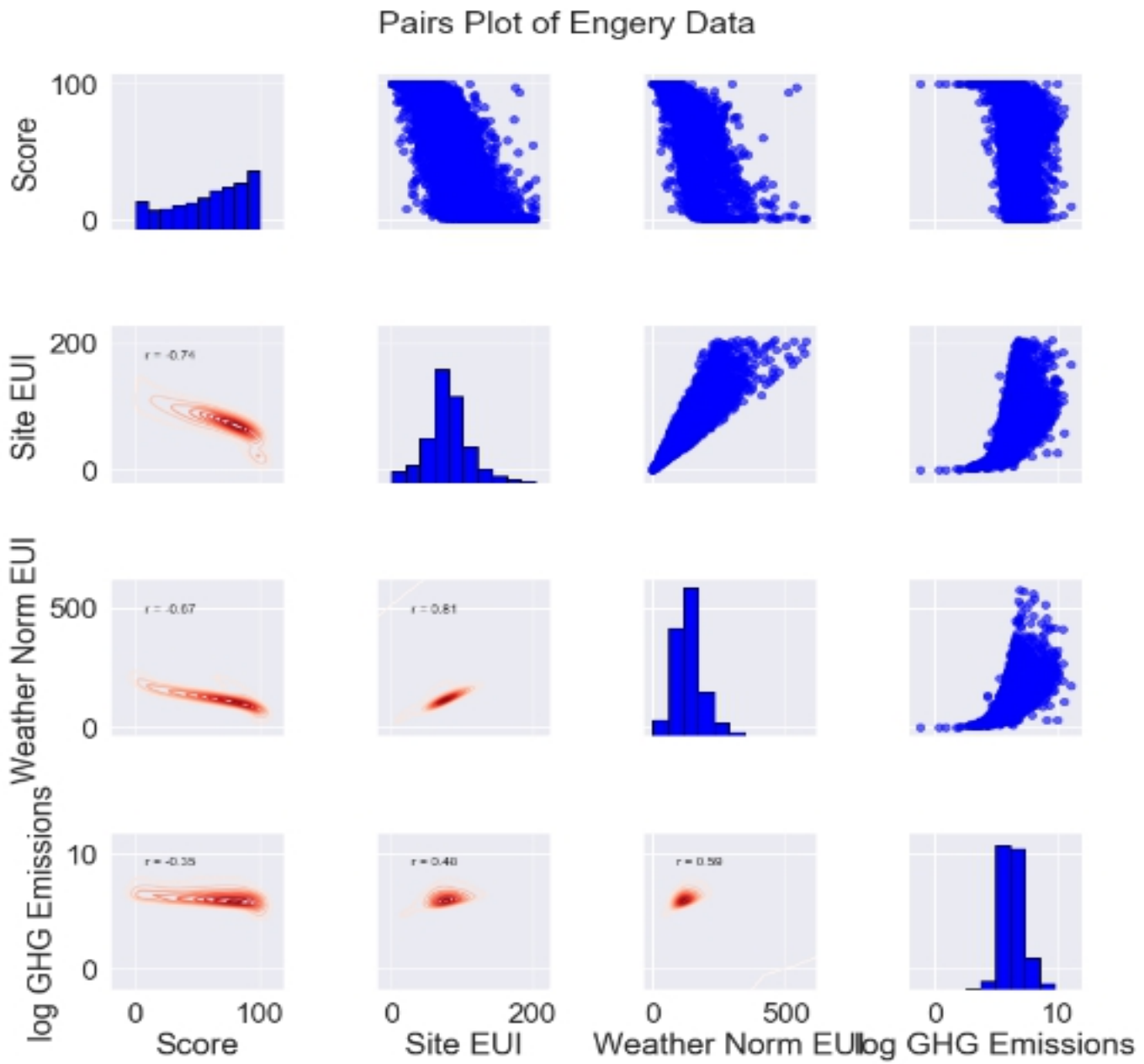

Energy Star Score vs Site EUI
Energy Star Score
100
80
60
40
20
0
0
50
100
150
200
Site EUI
Largest Property Use Type
Multifamily Housing
Office
Hotel
Non-Refrigerated Warehouse

Multivariate Analysis

```
# Extract the columns to  plot
plot_data = features[['Score', 'Site EUI (kBtu/ft²)',
                      'Weather Normalized Source EUI (kBtu/ft²)',
                      'log_Total GHG Emissions (Metric Tons CO2e)']]

# Replace the inf with nan
plot_data = plot_data.replace({np.inf: np.nan, -np.inf: np.nan})

# Rename columns
plot_data = plot_data.rename(columns = {'Site EUI (kBtu/ft²)': 'Site EUI',
                                        'Weather Normalized Source EUI (kBtu/ft²)': 'Weather Norm EUI',
                                        'log_Total GHG Emissions (Metric Tons CO2e)': 'log GHG Emissions'})

# Drop na values
plot_data = plot_data.dropna()

# Function to calculate correlation coefficient between two columns
def corr_func(x, y, **kwargs):
    r = np.corrcoef(x, y)[0][1]
    ax = plt.gca()
    ax.annotate("r = {:.2f}".format(r),
                xy=(.2, .8), xycoords = ax.transAxes,
                size = 10)
```

Multivariate Analysis

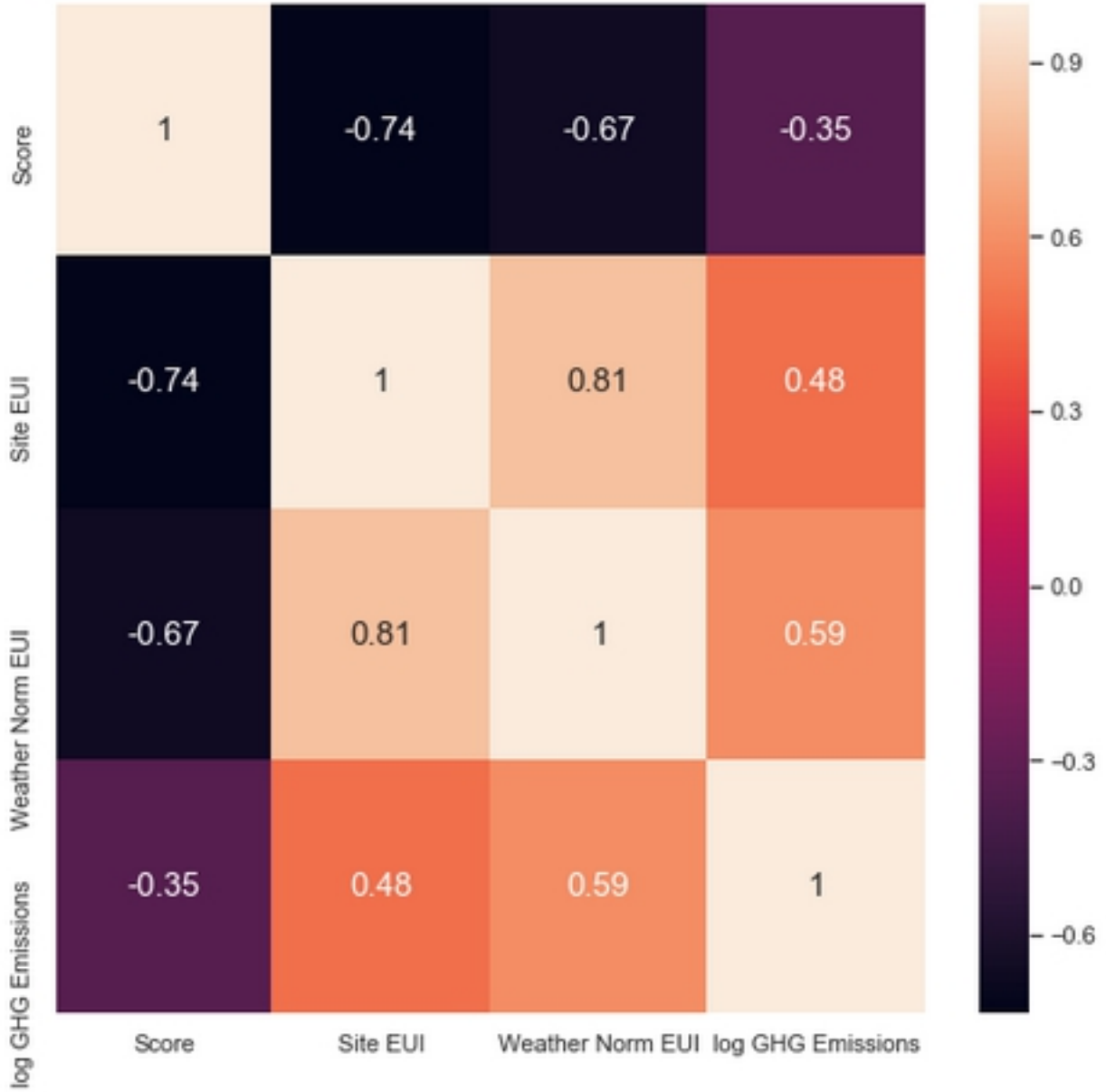
```
# Create the pairgrid object
grid = sns.PairGrid(data = plot_data, size = 3)

# Upper is a scatter plot
grid.map_upper(plt.scatter, color = 'blue', alpha = 0.6)

# Diagonal is a histogram
grid.map_diag(plt.hist, color = 'blue', edgecolor = 'black')

# Bottom is correlation and density plot
grid.map_lower(corr_func)
grid.map_lower(sns.kdeplot,cmap = plt.cm.Reds)

#Title for entire plot
plt.suptitle('Pairs Plot of Engery Data', size = 25, y = 1.02)
```


Pairs Plot of Engery Data
Score
Site EUI
Weather Norm EUI
log GHG Emissions
100
0
200
0
500
0
10
0
r = -0.74
r = -0.67
r = 0.81
r = -0.35
r = 0.48
r = 0.59
0
100
Score
0
200
Site EUI
0
500
Weather Norm EUI
0
10
log GHG Emissions

Multivariate Analysis

- We also utilize the heat map to visualize the correlation between the continuous variables.

```
corr = plot_data.corr()
f, ax = plt.subplots(figsize=(8, 8))
sns.heatmap(corr, vmax=1,annot_kws={'size': 15}, annot=True);
```


Score
Site EUI
Weather Norm EUI
log GHG Emissions
1
-0.74
-0.67
-0.35
-0.74
1
0.81
0.48
-0.67
0.81
1
0.59
-0.35
0.48
0.59
1
Score
Site EUI
Weather Norm EUI
log GHG Emissions
0.9
0.6
0.3
0.0
−0.3
−0.6

Multivariate Analysis

```
# we will limit the graph to building types that have
# more than 100 observations in the dataset.
building_types = data.dropna(subset=['Score'])
building_types = building_types['Largest Property Use Type'].value_counts()
building_types = list(building_types[building_types.values > 100].index)
print("Buidling types with more than 100 observations ",building_types)

# Create a list of boroughs with more than 100 observations
boroughs = data.dropna(subset=['Score'])
boroughs = boroughs['Borough'].value_counts()
boroughs = list(boroughs[boroughs.values > 100].index)
print("Boroughs with more than 100 observations ",boroughs)
```

```
Buidling types with more than 100 observations  ['Multifamily Housing', 'Office', 'Hotel', 'Non-Refrigerated Warehouse']
Boroughs with more than 100 observations  ['Manhattan', 'Brooklyn', 'Queens', 'Bronx', 'Staten Island']
```

Multivariate Analysis

- We filter the dataset based on the building types and boroughs.

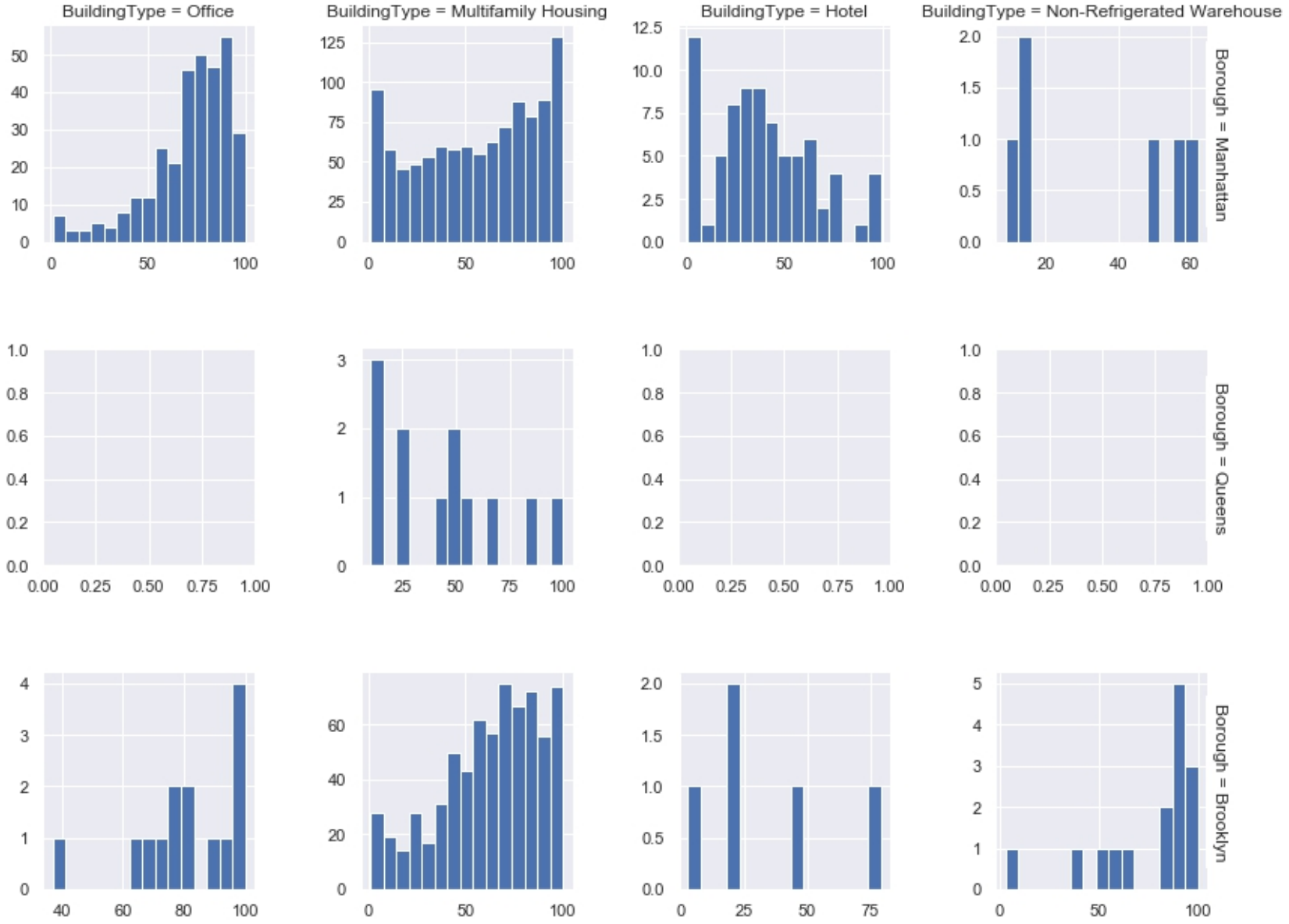
```
multivari_data = data[data['Largest Property Use Type'].isin(building_types) &
                      data['Borough'].isin(boroughs)].dropna()
multivari_data.rename(columns = {'Largest Property Use Type':"BuildingType"},
                      inplace = True)
multivari_data.head()
```

Multivariate Analysis

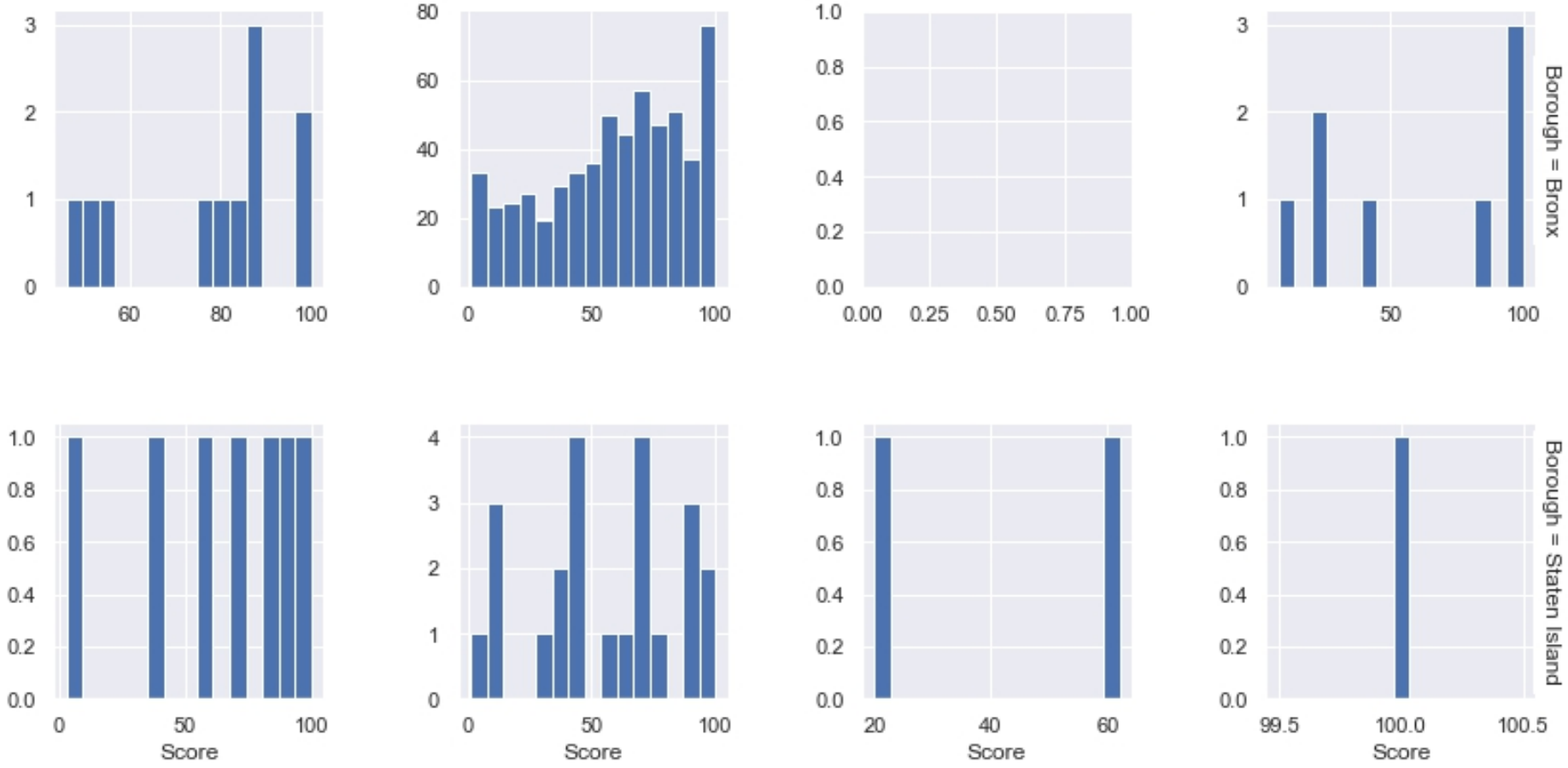
	Order	Property Id	Property Name	Parent Property Id	Parent Property Name	BBL - 10 digits	NYC Borough, Block and Lot (BBL) self-reported	NYC Building Identification Number (BIN)	Address 1 (self-reported)	Postal Code	Street Number
99	102	2605684	Hammer Health Sciences Center	3614737	Columbia University Medical Center	1021390051	1021390051	1063402	1 Haven Ave; 701 W 168 Street	10032	1
103	106	2741656	154 Haven Dormitory	3614737	Columbia University Medical Center	1021390275	1021390275	1063430	154 Haven Avenue	10032	154
161	165	2809891	434 West 120th Street	3618216	435 W 119 and 434 W 120	1019620070	1019620070	1059514	434 West 120th Street	10027	1211
323	332	4414870	Dayton Towers: 76-00 Shore Front Parkway	4994297	1-50/76-00 Dayton Towers	4161280001	4-16128-0001	4457805	76-00 Shore Front Parkway	11692	7600
327	336	4994375	Riverbend 2301-2311 (WW)	4994371	Riverbend (WW)	1017640001	1-01764-0001	1054345	2289-2311 5th Avenue	10037	2301

Multivariate Analysis

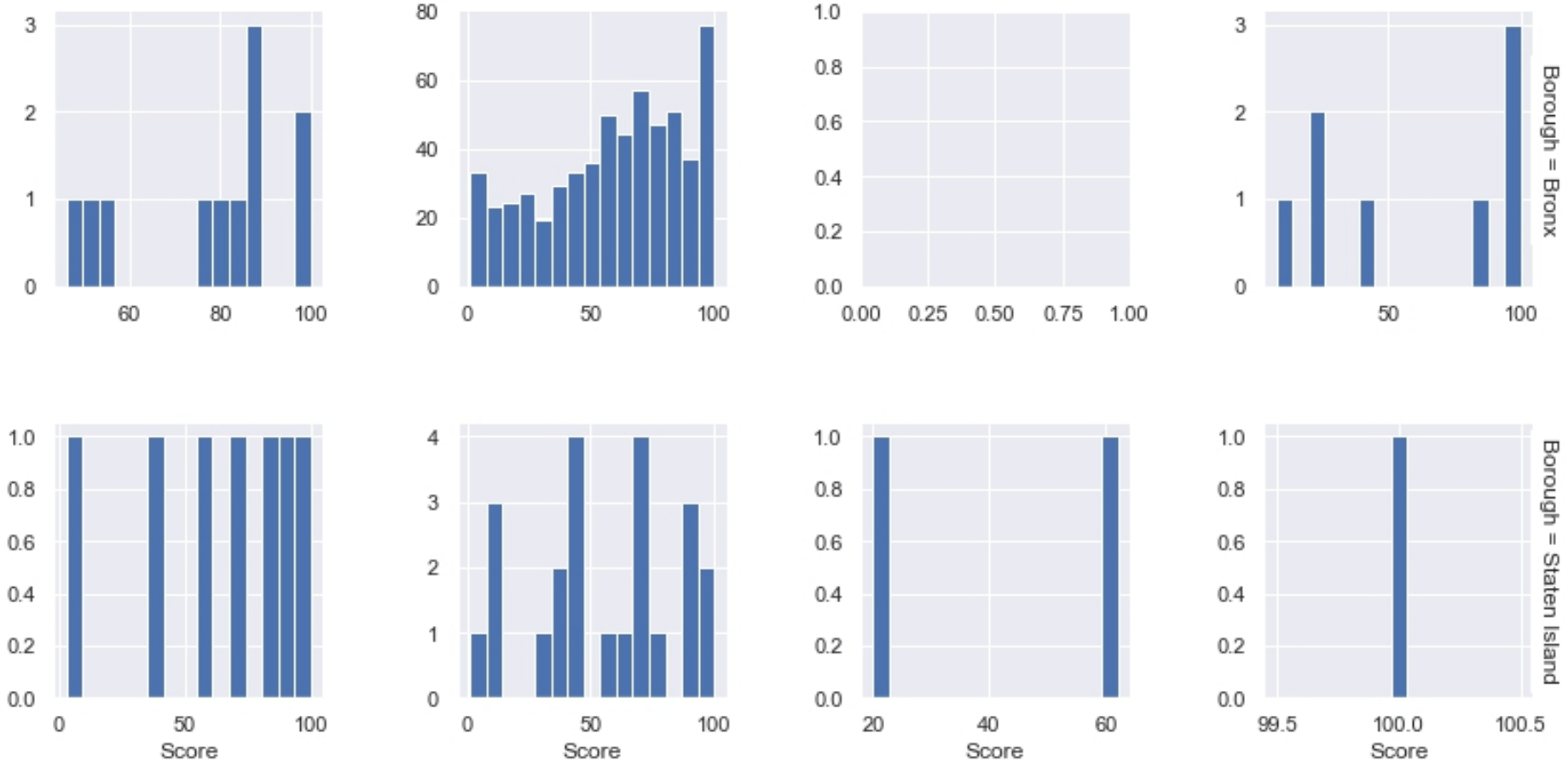
```
plt.figure(figsize=(20,12))
x=sns.FacetGrid(multivari_data, row='Borough',col = 'BuildingType',
                palette='husl',sharex=False,sharey=False, margin_titles=True)
x=x.map(plt.hist, 'Score', bins=15)
x=x.fig.subplots_adjust(wspace=0.5, hspace=0.5)
```


BuildingType = Office
BuildingType = Multifamily Housing
BuildingType = Hotel
BuildingType = Non-Refrigerated Warehouse
Borough = Manhattan
Borough = Queens
Borough = Brooklyn

Multivariate Analysis



Multivariate Analysis



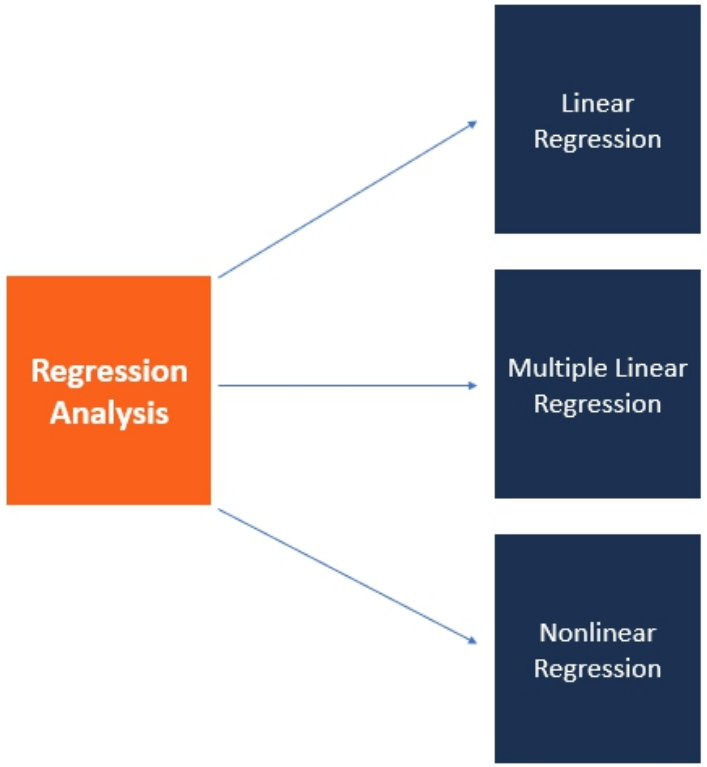
"Complete Assessment "

"Complete Lab 5 Programming Assignment"

6: Regression Analysis

Regression Analysis

- Regression analysis is a set of statistical procedures for estimating the relationship between a dependent variable (often called the ‘outcome variable’ or ‘target variable’)
- One or more independent variables (often called ‘predictors’ or ‘features’).



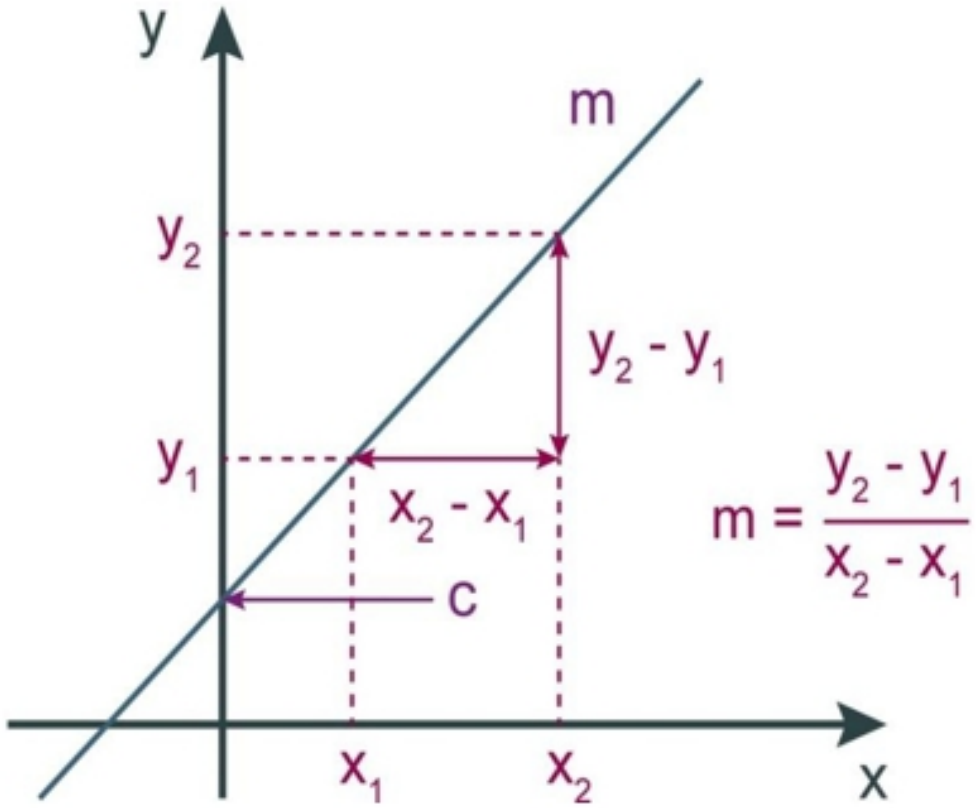
The use of Regression Analysis

The benefits of using Regression analysis for calculating these chances are as follows:

- It provides the significant relationships between the label (dependent variable) and the feature (independent variable).
- It shows the extent of the impact of multiple independent variables on the dependent variable.
- It can also quantify these effects even if the variables are on a different scale

Linear Regression

$$Y = MX + C$$



$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

Problem Statement

- This data is about the amount spent on advertising certain products through various media channels like TV, radio, and newspaper.

- The goal is to predict how the expense on each channel affects the sales and is there a way to optimize the sales?

Importing the necessary packages.

```
# necessary Imports
import pandas as pd
import matplotlib.pyplot as plt
import pickle
% matpllotlib inline
```

Loading and exploring the data.

```
# loading the data

data = pd.read_csv(r'D:\Data Sets\Advertising.csv',
                   index_col='Index')

# Checking the first five rows from the datatset
data.head()
```

Index	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

Problem Statement

- Dimensions of the data

```
data.shape
```

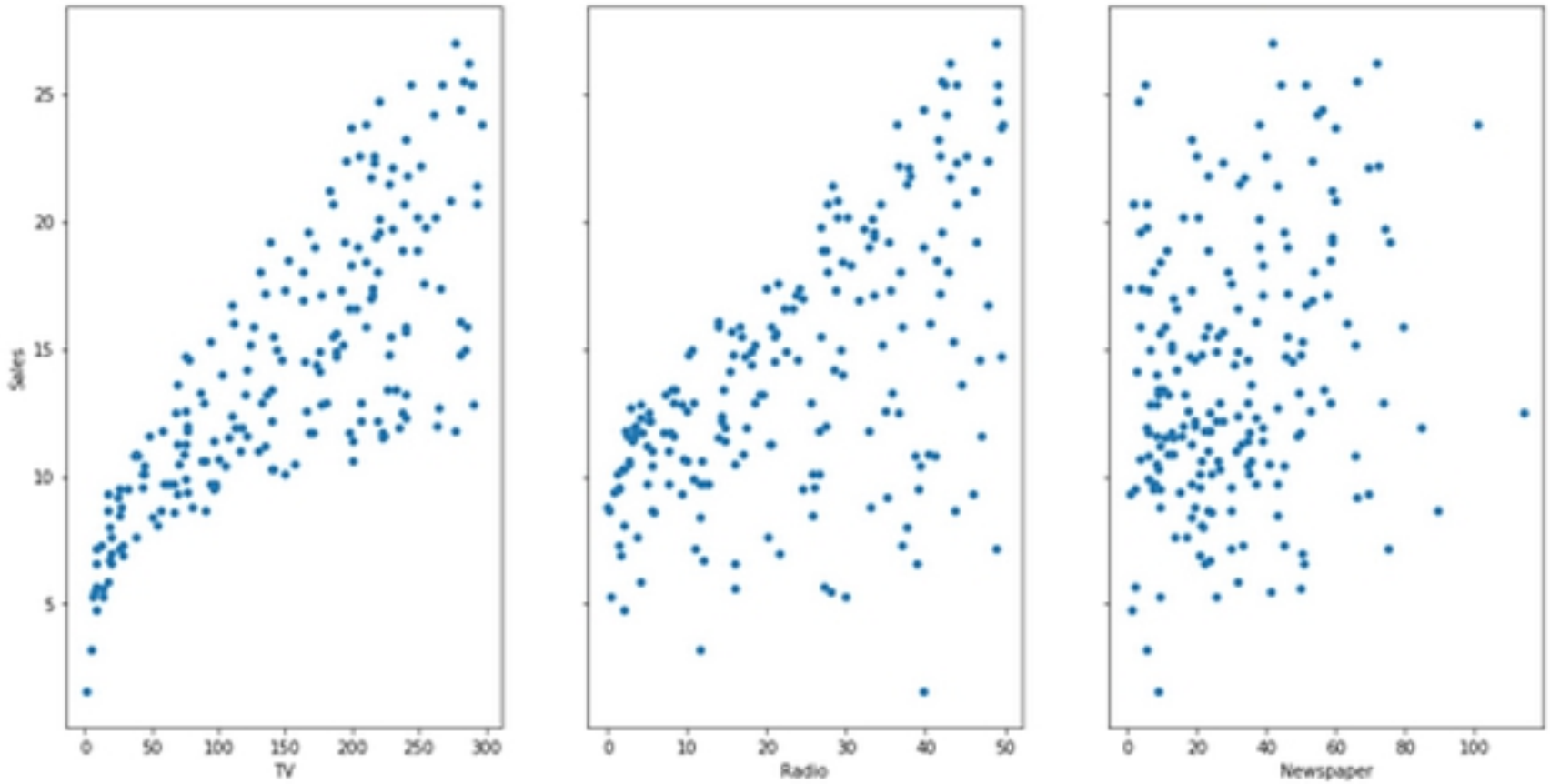
```
(200, 4)
```

- Find the missing values from different columns:

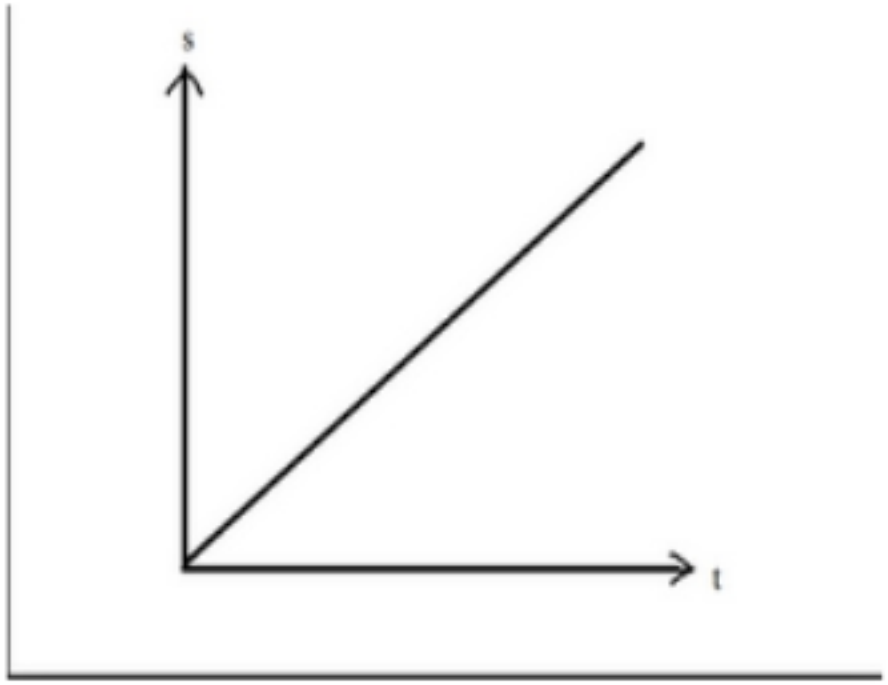
```
data.isna().sum()
```

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

```
# visualize the relationship between the features and the response using scatterplots
fig, axs = plt.subplots(1, 3, sharey=True)
data.plot(kind='scatter', x='TV', y='Sales', ax=axs[0], figsize=(16, 8))
data.plot(kind='scatter', x='Radio', y='Sales', ax=axs[1])
data.plot(kind='scatter', x='Newspaper', y='Sales', ax=axs[2])
```



Linear Regression



- Hence, we can build a model using the Linear Regression Algorithm.

Simple Linear Regression

- A simple linear regression is a method for predicting a quantitative response using a single feature ("input variable").

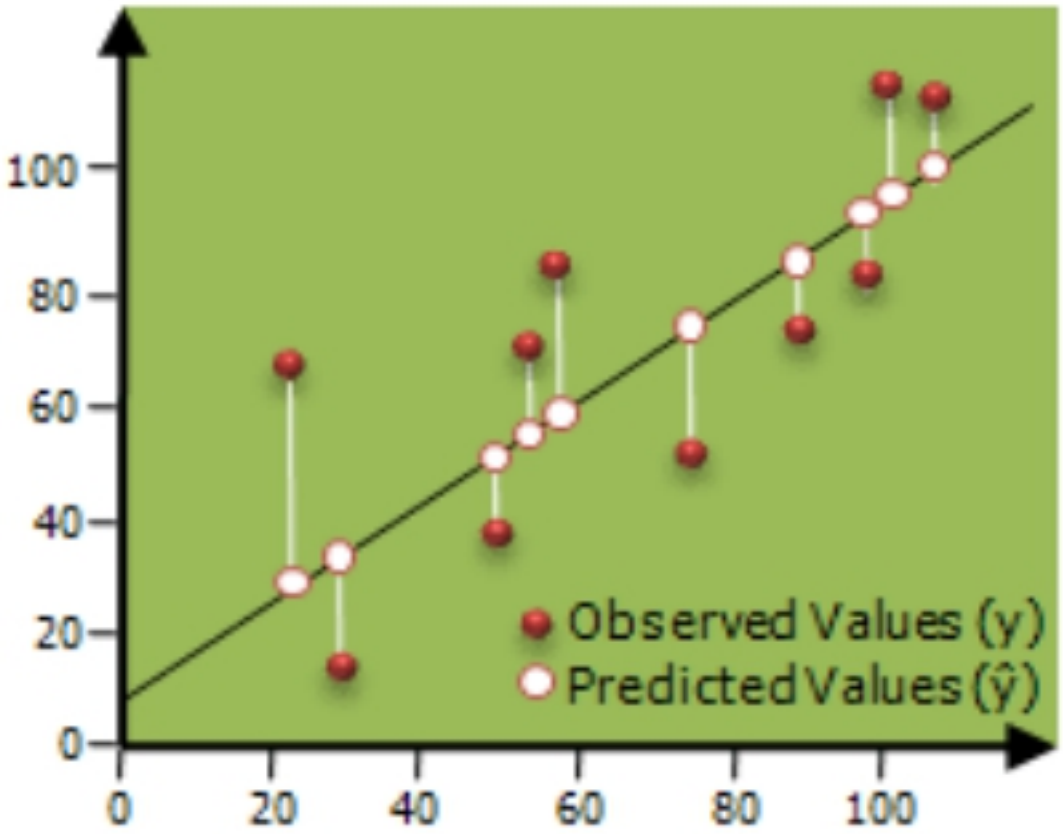
- The mathematical equation is given as:

$$y = \beta 0 + \beta 1 x$$

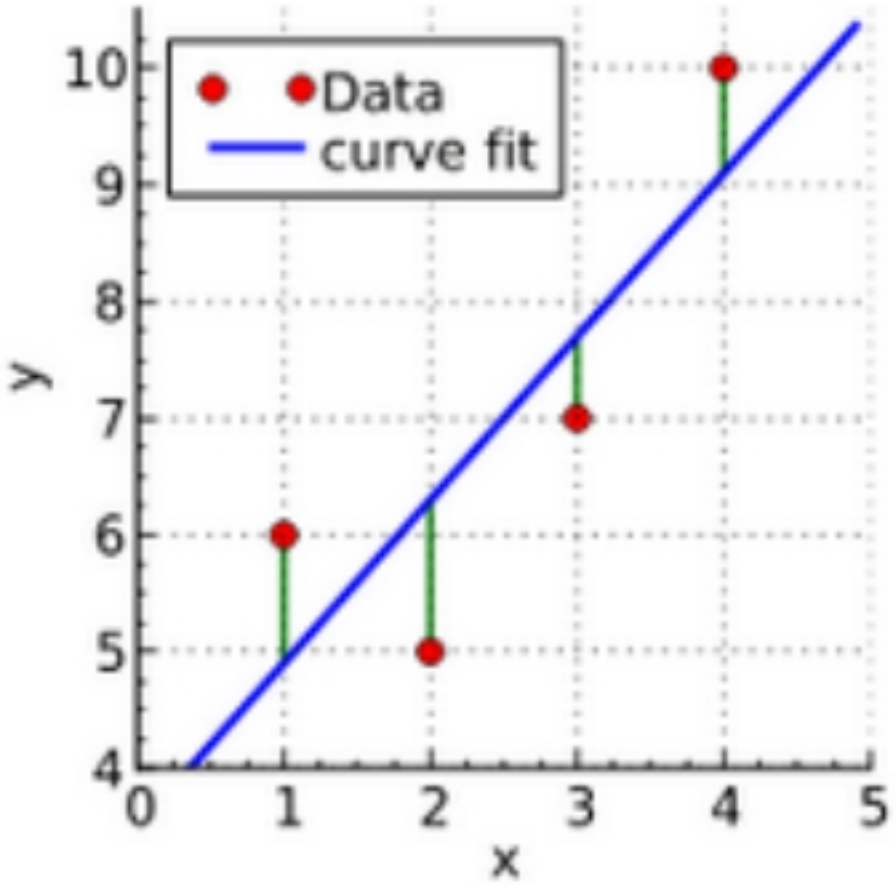
Estimating ("Learning") Model Coefficients

- The coefficients are estimated using the Ordinary Least Squares estimates criterion (OLS],
- i.e., a best fit line that minimizes the sum of the squared residuals (or "Sum of Squared Error").

Ordinary Least Squared Estimation



The Mathematics involved



Derivation of OLS by Minimizing Errors

- Minimize the sum of squared error term by substituting as below:

$$\text{Minimize} \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (y_i - \hat{y})^2; \text{ where } \hat{y} = \beta_0 + \beta_1 x_i$$

$$\text{Therefore,} \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

Derivation of OLS by Minimizing Errors

- Solving the above equation by calculus – partial differencing by $\beta 0$ and $\beta 1$ respectively and solving for the two variables, we get the following equations,

- $\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{\overline{xy} - \bar{x}\,\bar{y}}{\overline{x^2} - \bar{x}^2}$
- $\beta_0 = \bar{y} - \beta_1 \bar{x}$

Derivation of OLS by Minimizing Errors

- Building Simple Linear Regression Model to predict the sales based on TV ads,

```
# create X and y
feature_cols = ['TV']
X = data[feature_cols]
y = data.Sales

# follow the usual sklearn pattern: import, instantiate, fit
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X, y)

# print intercept and coefficients
print(lm.intercept_)
print(lm.coef_)
```

```
7.032593549127695
[0.04753664]
```

Prediction using the model

If the expense on a TV ad is $50000, what will be the sales prediction for that market?

$$y = \beta_0 + \beta_1 x$$

Y = 7.032594 + 0.047537 * (50)

```
#calculate the prediction
7.032594 + 0.047537*50
```

```
9.409444
```

Prediction using the model

Let's do the same calculation using the code.

```
#  Let's create a DataFrame since the model expects it
X_new = pd.DataFrame({'TV': [50]})
X_new.head()
```

	TV
0	50

```
# use the model to make predictions on a new value
lm.predict(X_new)
```

```
array([9.40942557])
```

Plotting the least Squares Line:

```
# create a DataFrame with the minimum and maximum values of TV
X_new = pd.DataFrame({'TV': [data.TV.min(), data.TV.max()]})
X_new.head()
```

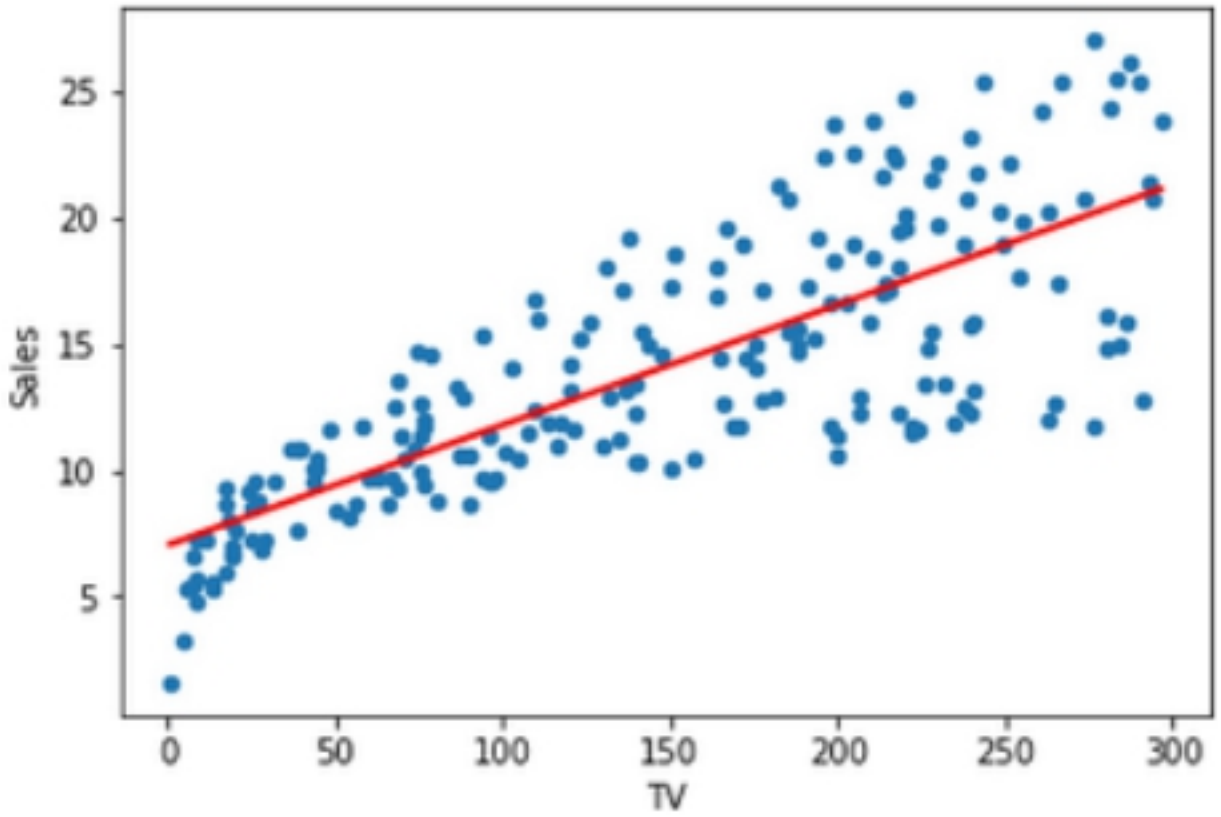
	TV
0	0.7
1	296.4

```
# make predictions for those x values and store them
preds = lm.predict(X_new)
preds
```

```
array([ 7.0658692 , 21.12245377])
```

```
# first, plot the observed data
data.plot(kind='scatter', x='TV', y='Sales')

# then, plot the least squares line
plt.plot(X_new, preds, c='red', linewidth=2)
```

Plotting the least Squares Line:



Plotting the least Squares Line:

```
import statsmodels.formula.api as smf
lm = smf.ols(formula='sales ~ TV', data=data).fit()
lm.conf_int()
```

	0	1
Intercept	6.129719	7.935468
TV	0.042231	0.052843

Hypothesis Testing and p-values

```
import statsmodels.formula.api as smf
lm = smf.ols(formula='sales ~ TV', data=data).fit()
lm.conf_int()
```

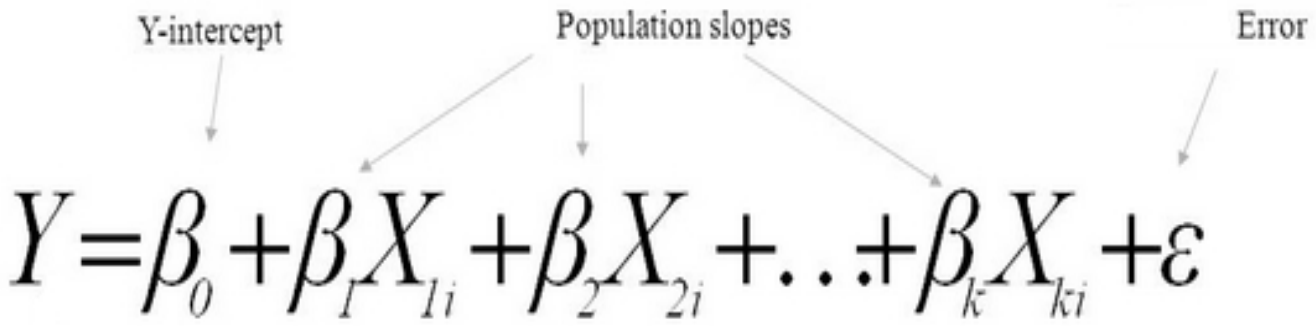
	0	1
Intercept	6.129719	7.935468
TV	0.042231	0.052843

```
# print the p-values for the model coefficients
lm.pvalues
```

```
Intercept    1.406300e-35
TV           1.467390e-42
dtype: float64
```

Multiple Linear Regression

- A multiple linear regression is a method for predicting a quantitative response using a multiple feature ("input variable").



Estimation of model parameters

- Consider the model where,

$$Y = X\beta + \epsilon$$

$$Y = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix} \quad X = \begin{pmatrix} 1 & X_{11} & X_{12} & \dots & X_{1p} \\ 1 & X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{n1} & X_{n2} & \dots & X_{np} \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \quad \epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

Estimation of model parameters

- Based on this model, we get the following expansion for the first subject:

$$Y_1 = \beta_0 + \beta_1 X_{11} + \beta_2 X_{12} + \ldots + \beta_p X_{1p} + \epsilon_1$$

- Then using the matrix calculus, we can find that the least square estimate for β is given by,

$$\hat{\beta} = (X^T X)^{-1} X^T Y \qquad \mathrm{Var}(\hat{\beta}) = \sigma^2 (X^\top X)^{-1}$$

Estimation of model parameters

Hence, the least squares regression line is:

$$\hat{Y} = X\hat{\beta}$$

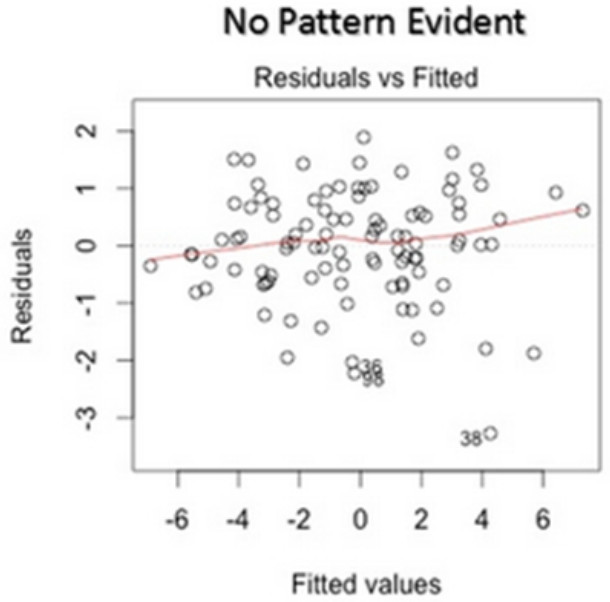
The beta values are obtained by calculating the below equation:

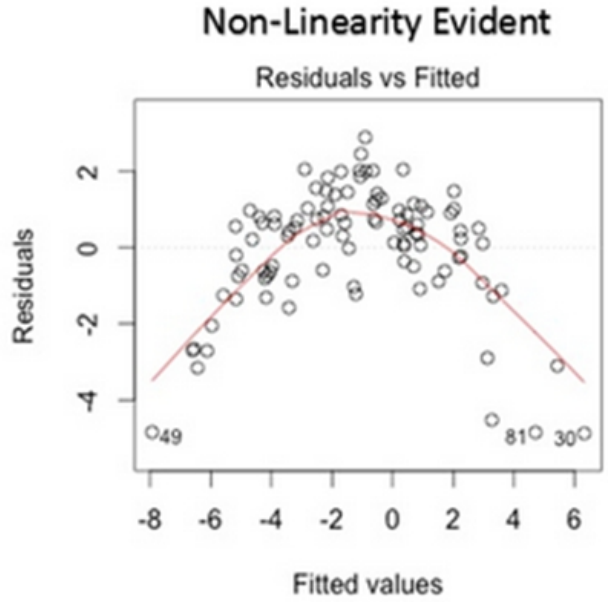
$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

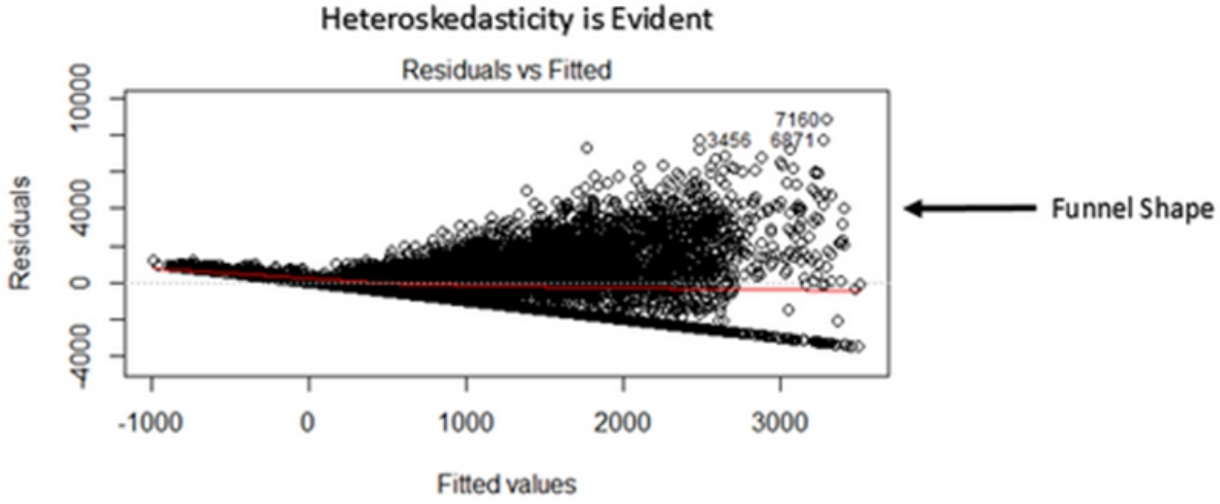
We know that,

$$\begin{aligned} \hat{Y} &= X\hat{\beta} \\ &= X(X^T X)^{-1} X^T Y \\ &= HY \end{aligned}$$

Interpretation of Regression Plots

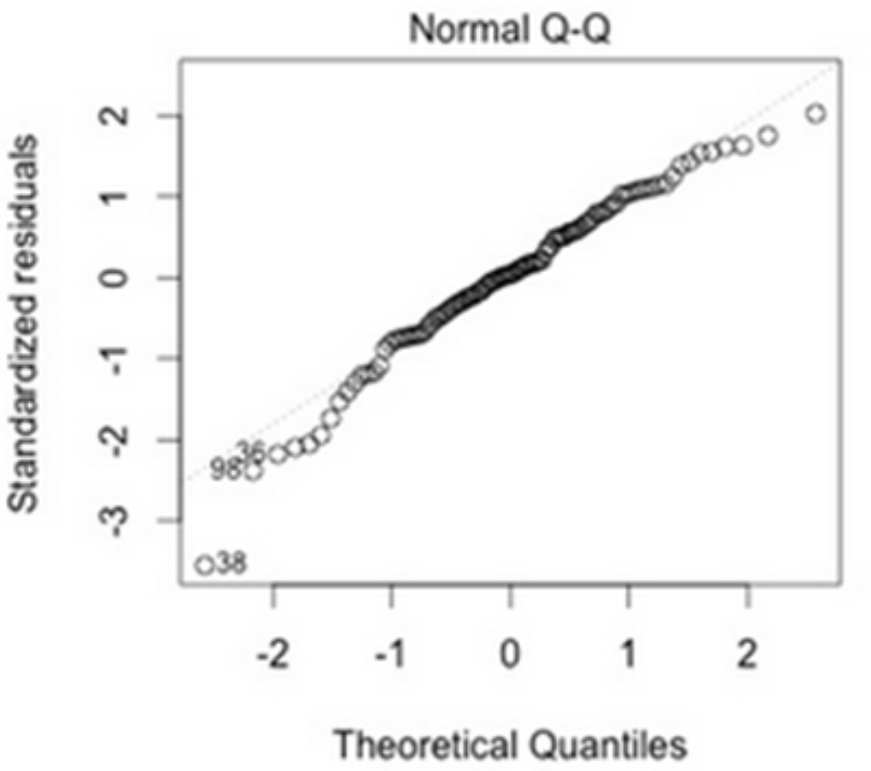




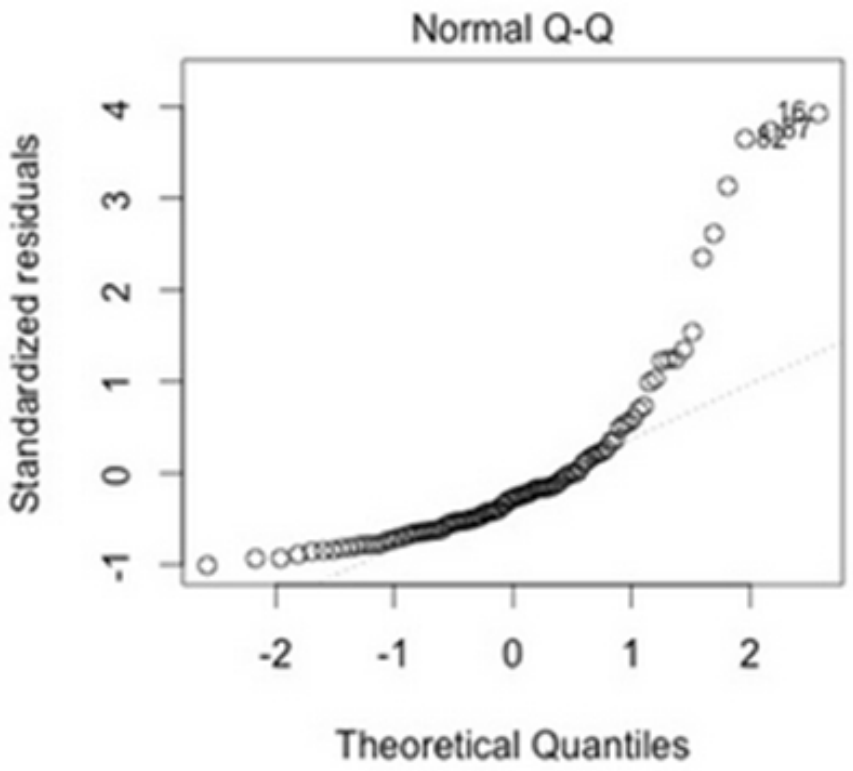


Normal Q-Q Plot

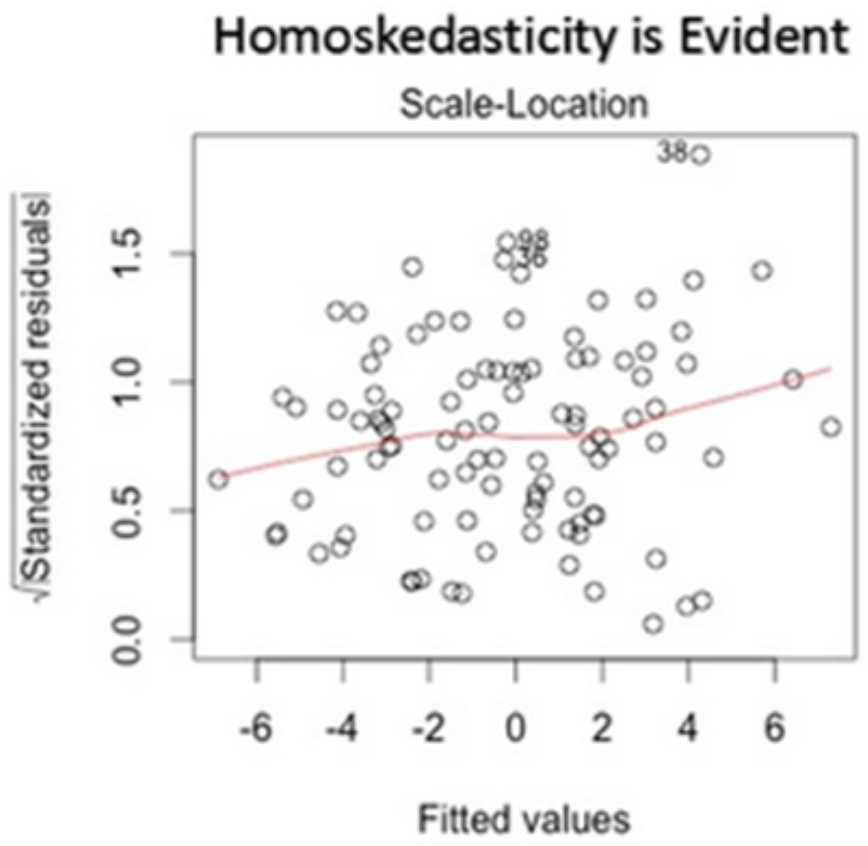
Normal Distribution Evident

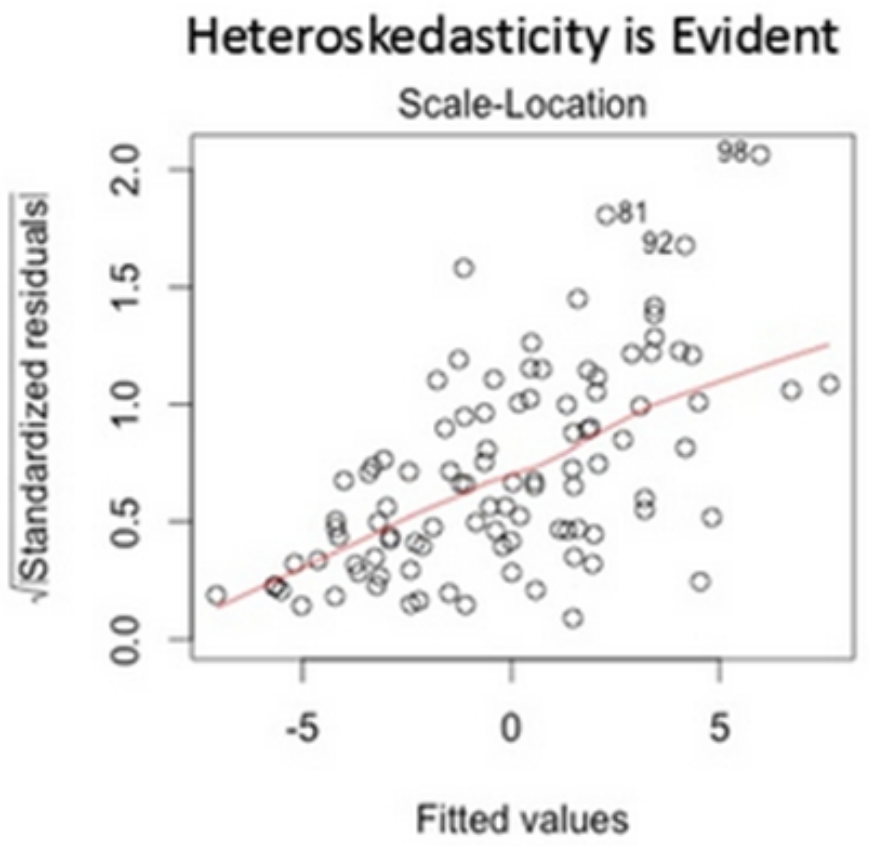


Not Normally Distributed

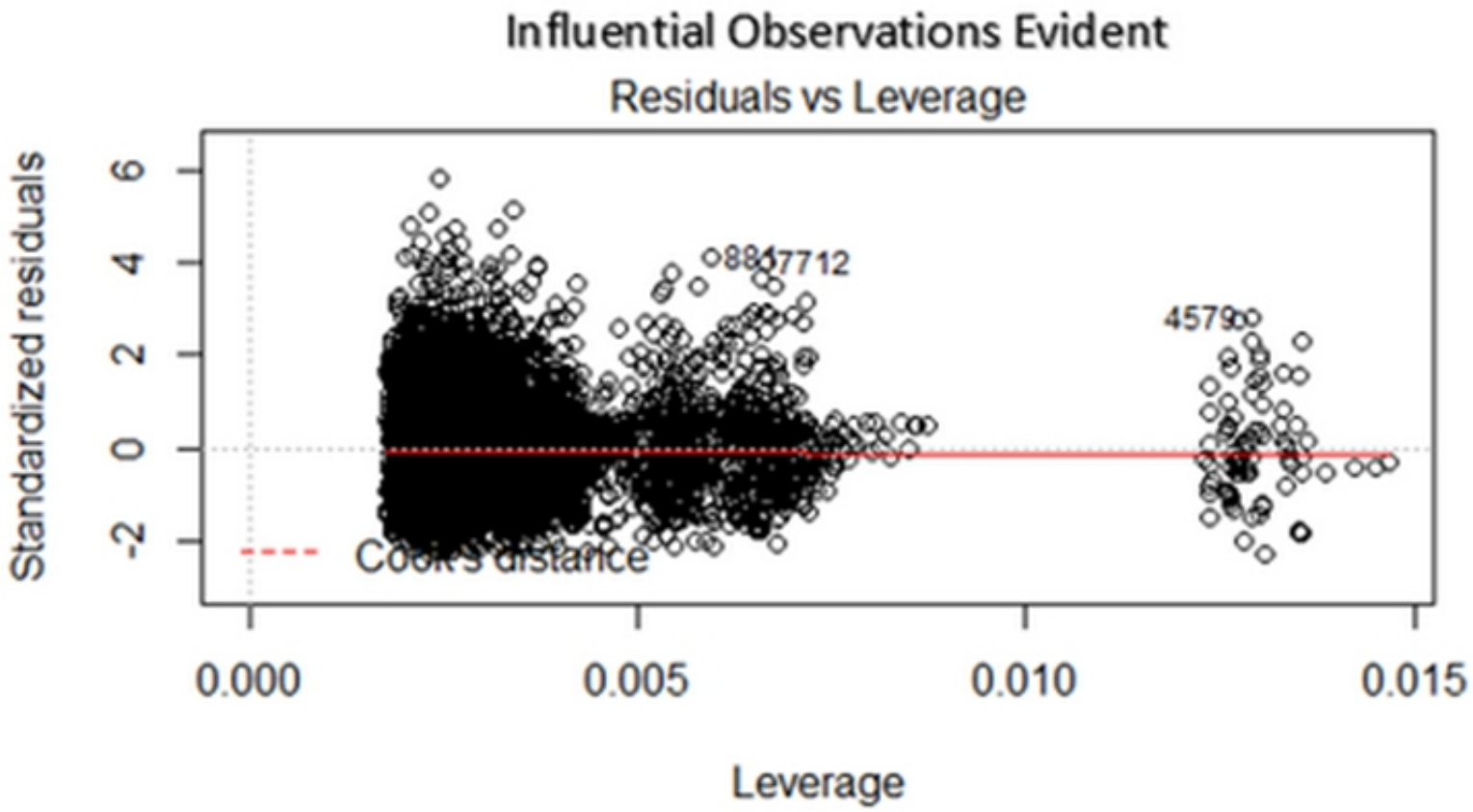


Scale Location Plot





Residuals vs Leverage Plot



Model Evaluation Metrics

There are 5 major metrics for model evaluation in regression analysis:

1. R Square
2. Adjusted R Square
3. Mean Square Error (MSE)
4. Root Mean Squared Error (RMSE)
5. Mean Absolute Error

R Square/Adjusted R Square

Mathematically, $\mathbf{R^2}$ statistic is calculated as:

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

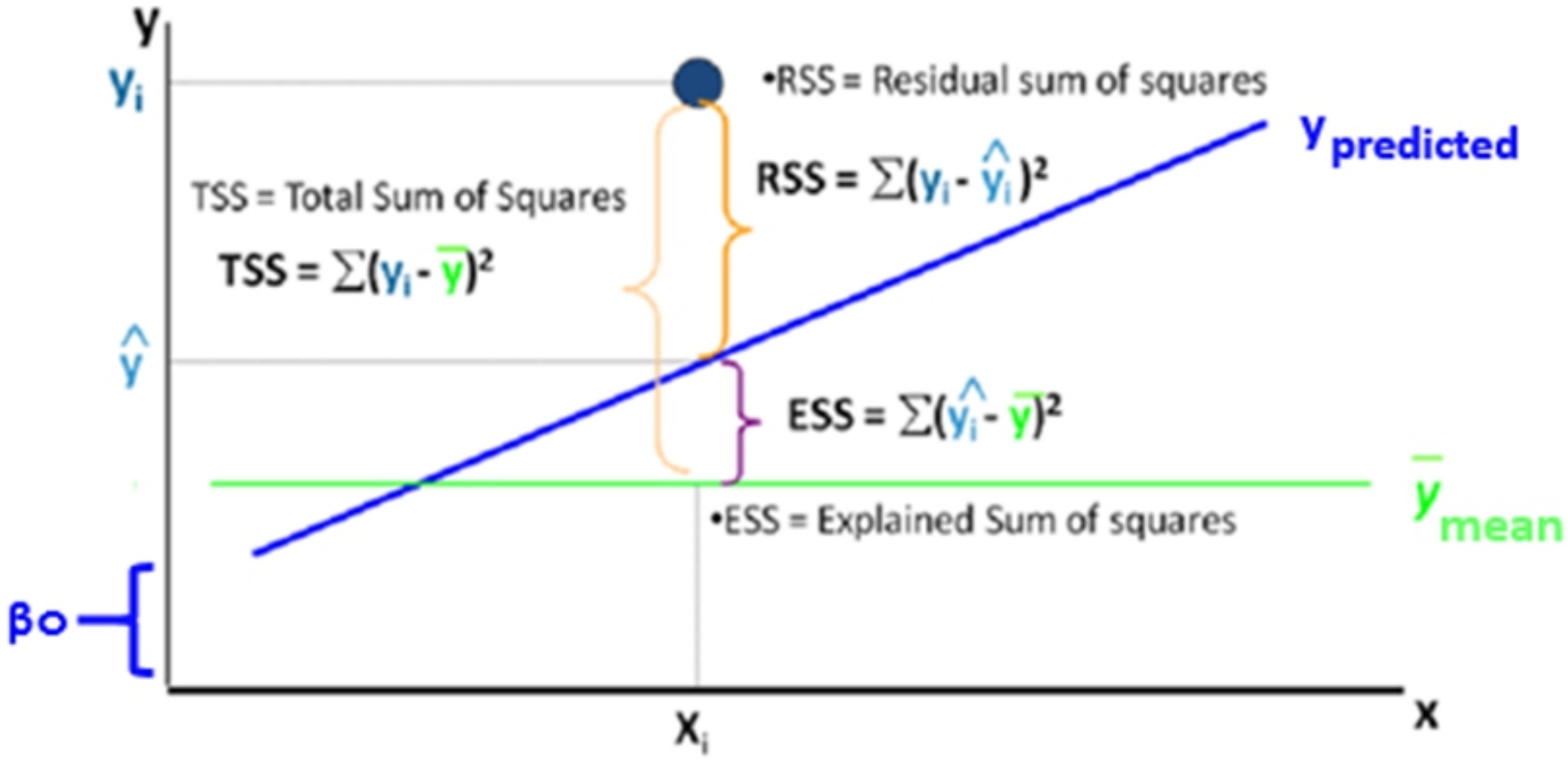
Where **RSS** is the Residual Sum of Squares and is given as:

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

RSS is the residual (error) term, whereas, TSS is the Total Sum of Squares and given as:

$$\text{TSS} = \sum (y_i - \bar{y})^2$$

R Square/Adjusted R Square



Adjusted R Square

- Adjusted R Square is introduced to overcome this problem because it penalizes additional independent variables added to the model and adjust the metric to prevent the overfitting issues.

- Mathematically, it is calculated as:

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

Mean Square Error (MSE)

- While R Square is a relative measure of how well the model fits the dependent variables, Mean Square Error (MSE) is an absolute measure of the goodness of fit.

- Mathematically, it is calculated as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Root Mean Square Error (RMSE)

- Root Mean Square Error (RMSE) is the square root of MSE.
- It is used more commonly as compared to MSE, because sometimes MSE values can be too big for easy comparisons.
- Mathematically it is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2}$$

Mean Absolute Error

- Mean Absolute Error (MAE) is similar to Mean Square Error (MSE).
- However, instead of the sum of square error in MSE, MAE is taking the Sum of Absolute value of error.
- Mathematically, MAE is calculated as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

"Complete Lab 6"

Summary

- Regression models describe the relationship between variables by fitting a line to the observed dataset.
- Generally, the linear regression models use a straight line.
- Regression allows us to estimate how a dependent variable changes with the change in the independent variables.

"Complete Assessment"