Data Analytics

Professor Ernesto Lee

Wide versus Long Data

LONG FORMAT DATA

Product	Attribute	Value
A	Height	10
A	Width	5
A	Weight	2
B	Height	20
B	Width	10

WIDE FORMAT DATA

Prod	Hght	Wght	Weight
A	10	5	2
B	20	10	NA

Wide or Long?

- Often, people will record and present data in wide format, but there are certain visualizations that require the data to be in long format:

variables

observations

	date	TMAX	TMIN	TOBS
0	2018-10-01	21.1	8.9	13.9
1	2018-10-02	23.9	13.9	17.2
2	2018-10-03	25.0	15.6	16.1
3	2018-10-04	22.8	11.7	11.7
4	2018-10-05	23.3	11.7	18.9
5	2018-10-06	20.0	13.3	16.1

variable names

variable values

repeated values for **date** column

	date	datatype	value
0	2018-10-01	TMAX	21.1
1	2018-10-01	TMIN	8.9
2	2018-10-01	TOBS	13.9
3	2018-10-02	TMAX	23.9
4	2018-10-02	TMIN	13.9
5	2018-10-02	TOBS	17.2

Wide or Long

- read in the CSV files containing wide and long format data:
  - https://github.com/fenago/machine-learning-essentials-module1/blob/master/lab_08/data/wide_data.csv

date	TMAX	TMIN	TOBS
2018-10-01	21.1	8.9	13.9
2018-10-02	23.9	13.9	17.2
2018-10-03	25.0	15.6	16.1
2018-10-04	22.8	11.7	11.7
2018-10-05	23.3	11.7	18.9
2018-10-06	20.0	13.3	16.1
2018-10-07	20.0	16.1	20.0
2018-10-08	26.7	17.8	17.8
2018-10-09	18.9	17.2	17.8
2018-10-10	24.4	17.2	18.3
2018-10-11	26.1	17.8	21.7
2018-10-12	22.8	14.4	15.6
2018-10-13	15.6	7.2	8.3
2018-10-14	13.3	5.6	6.7
2018-10-15	13.3	6.7	10.0
2018-10-16	18.9	7.8	7.8
2018-10-17	13.3	3.3	5.0
2018-10-18	16.1	4.4	5.0
2018-10-19	10.0	-1.1	0.0
2018-10-20	15.0	-0.6	10.6
2018-10-21	16.7	7.8	7.8

Wide or Long

- Each column contains the top six observations of a specific class of temperature data in degrees Celsius—maximum temperature (TMAX), minimum temperature (TMIN), and temperature at the time of observation (TOBS)—at a daily frequency:

	date	TMAX	TMIN	TOBS
0	2018-10-01	21.1	8.9	13.9
1	2018-10-02	23.9	13.9	17.2
2	2018-10-03	25.0	15.6	16.1
3	2018-10-04	22.8	11.7	11.7
4	2018-10-05	23.3	11.7	18.9
5	2018-10-06	20.0	13.3	16.1

Data Insight…

- When working with wide format data, we can easily grab summary statistics on this data!!!

Wide or Long

- With hardly any effort on our part, we get summary statistics for the dates, maximum temperature, minimum temperature, and temperature at the time of observation:

	date	TMAX	TMIN	TOBS
count	31	31.000000	31.000000	31.000000
mean	2018-10-16 00:00:00	16.829032	7.561290	10.022581
min	2018-10-01 00:00:00	7.800000	-1.100000	-1.100000
25%	2018-10-08 12:00:00	12.750000	2.500000	5.550000
50%	2018-10-16 00:00:00	16.100000	6.700000	8.300000
75%	2018-10-23 12:00:00	21.950000	13.600000	16.100000
max	2018-10-31 00:00:00	26.700000	17.800000	21.700000
std	NaN	5.714962	6.513252	6.596550

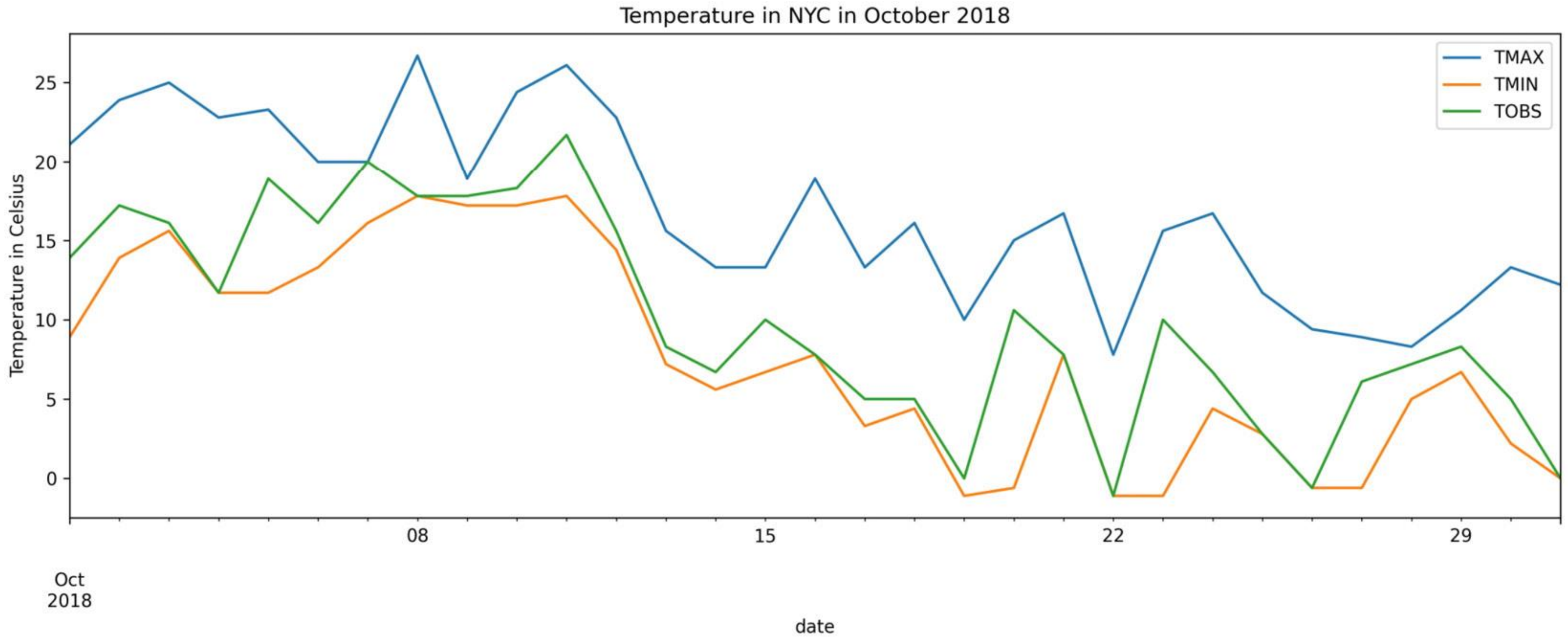
Wide Data is Easy to Visualize in PowerBI

- The summary data in the preceding table is easy to obtain and is informative.

- This format can easily be plotted with PowerBI

Data transformation

- PowerBI plots the daily maximum temperature, minimum temperature, and temperature at the time of observation as their own lines on a single line plot:



Long Format Data

- We can look at the top six rows of the long format data in long_df to see the difference between wide format and long format data:

https://github.com/fenago/machine-learning-essentials-module1/blob/master/lab_08/data/long_data.csv

attributes	datatype	date	station	value
,,H,0700	TMAX	2018-10-01T00:00:00	GHCND:USC00280907	21.1
,,H,0700	TMIN	2018-10-01T00:00:00	GHCND:USC00280907	8.9
,,H,0700	TOBS	2018-10-01T00:00:00	GHCND:USC00280907	13.9
,,H,0700	TMAX	2018-10-02T00:00:00	GHCND:USC00280907	23.9
,,H,0700	TMIN	2018-10-02T00:00:00	GHCND:USC00280907	13.9
,,H,0700	TOBS	2018-10-02T00:00:00	GHCND:USC00280907	17.2
,,H,0700	TMAX	2018-10-03T00:00:00	GHCND:USC00280907	25.0
,,H,0700	TMIN	2018-10-03T00:00:00	GHCND:USC00280907	15.6
,,H,0700	TOBS	2018-10-03T00:00:00	GHCND:USC00280907	16.1
,,H,0700	TMAX	2018-10-04T00:00:00	GHCND:USC00280907	22.8
,,H,0700	TMIN	2018-10-04T00:00:00	GHCND:USC00280907	11.7
,,H,0700	TOBS	2018-10-04T00:00:00	GHCND:USC00280907	11.7
,,H,0700	TMAX	2018-10-05T00:00:00	GHCND:USC00280907	23.3
,,H,0700	TMIN	2018-10-05T00:00:00	GHCND:USC00280907	11.7
,,H,0700	TOBS	2018-10-05T00:00:00	GHCND:USC00280907	18.9
,,H,0700	TMAX	2018-10-06T00:00:00	GHCND:USC00280907	20.0
,,H,0700	TMIN	2018-10-06T00:00:00	GHCND:USC00280907	13.3
,,H,0700	TOBS	2018-10-06T00:00:00	GHCND:USC00280907	16.1

Long Data

- Notice that we now have three entries for each date, and the datatype column tells us what the data in the value column is for that row:

	date	datatype	value
0	2018-10-01	TMAX	21.1
1	2018-10-01	TMIN	8.9
2	2018-10-01	TOBS	13.9
3	2018-10-02	TMAX	23.9
4	2018-10-02	TMIN	13.9
5	2018-10-02	TOBS	17.2

Long Data

- If we try to get summary statistics, like we did with the wide format data, the result isn't as helpful.

Long Data

- This means that this summary data is not very helpful:

	date	datatype	value
count	93	93	93.000000
unique	NaN	3	NaN
top	NaN	TOBS	NaN
freq	NaN	31	NaN
mean	2018-10-16 00:00:00	NaN	11.470968
min	2018-10-01 00:00:00	NaN	-1.100000
25%	2018-10-08 00:00:00	NaN	6.700000
50%	2018-10-16 00:00:00	NaN	11.700000
75%	2018-10-24 00:00:00	NaN	17.200000
max	2018-10-31 00:00:00	NaN	26.700000
std	NaN	NaN	7.362354

Summary

- PowerBI often expects its data for plotting to be in wide format.
- So, you should get comfortable converting from Long to Wide format data and vice versa.

The Importance of Data Types

- Know the type and scale for every columns
- Go to Kaggle.com and FIND a dataset to use as we go through this lesson

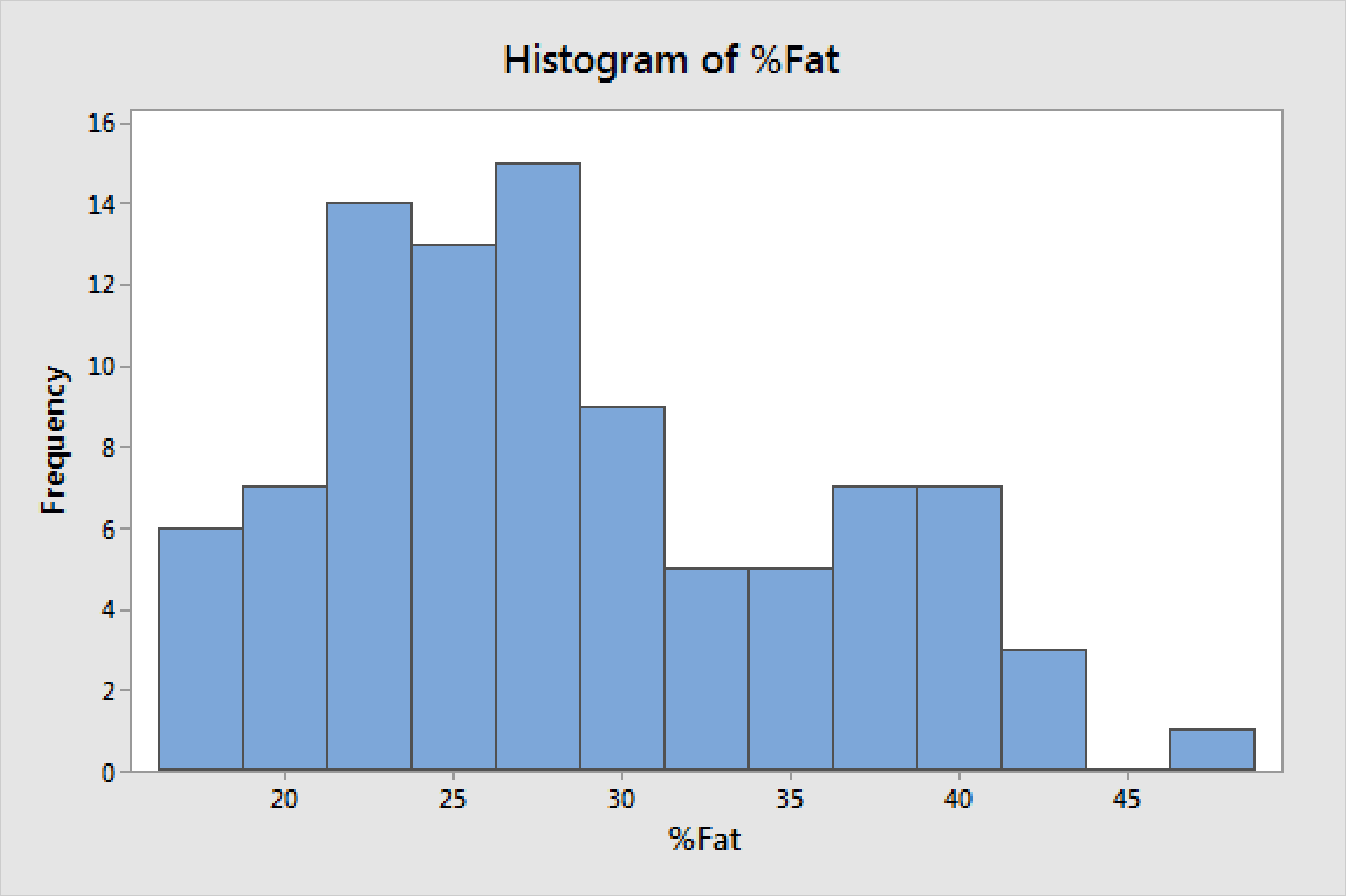
Quantitative or Qualitative

- The difference between these two types of data is the most fundamental way to divide them.
    - ***So, are characteristics something you measure with numbers or not?***

Continuous Numeric Data (Quantitative)

- This type of data takes on any numerical value, and can be divided into smaller increments — this includes decimal and fractional values, which means there is an infinite number of potential values between any two values.
- The differences between the two values are always meaningful and are typically measured on a scale.

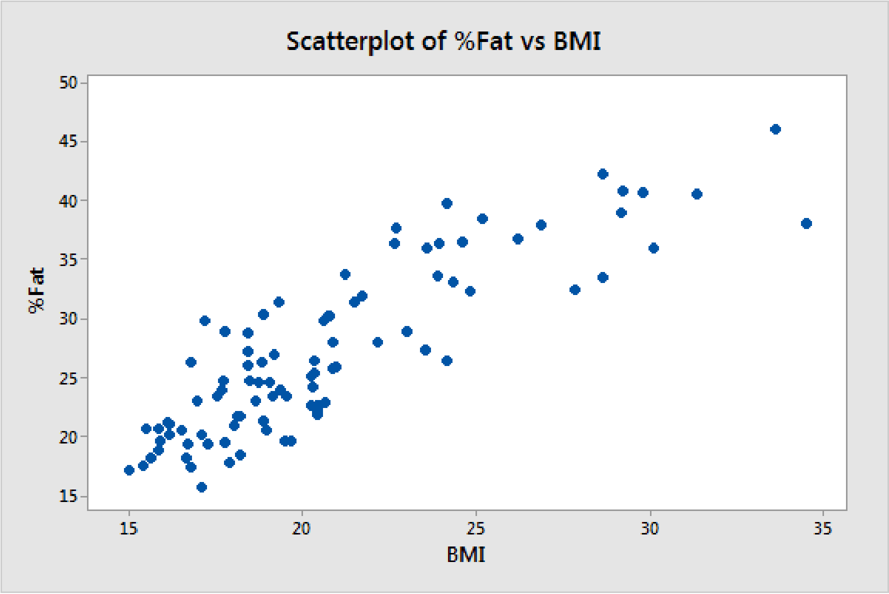
Histograms are good for CONTINUOUS Data



Scatter Plots

- Scatterplots are used to record two continuous variables on a graph; they are perfect for showing the relationship between those two variables clearly since every dot on the graph has an X and Y coordinate which corresponds to a pair of values.

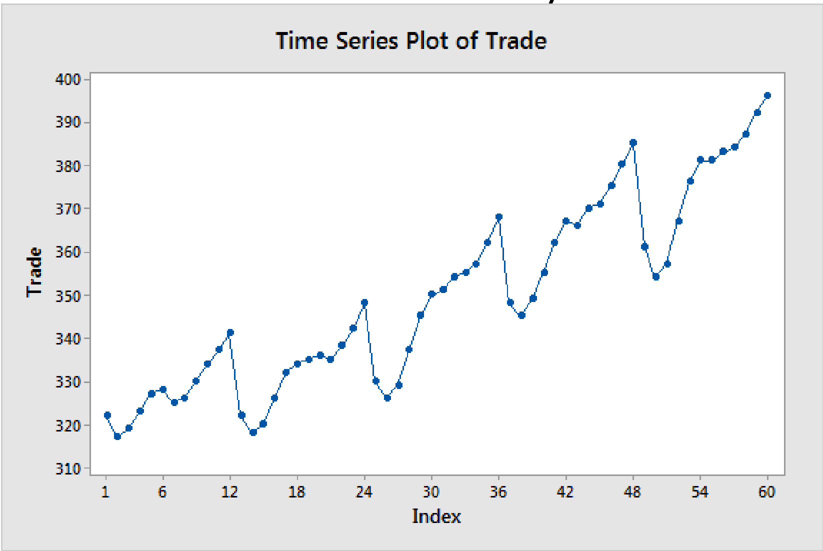
BMI (X)	%Fat (Y)
19.3	23.9
23.0	28.8
27.8	32.4
20.9	25.8
20.4	22.5
20.4	22.1



Time Series Plots

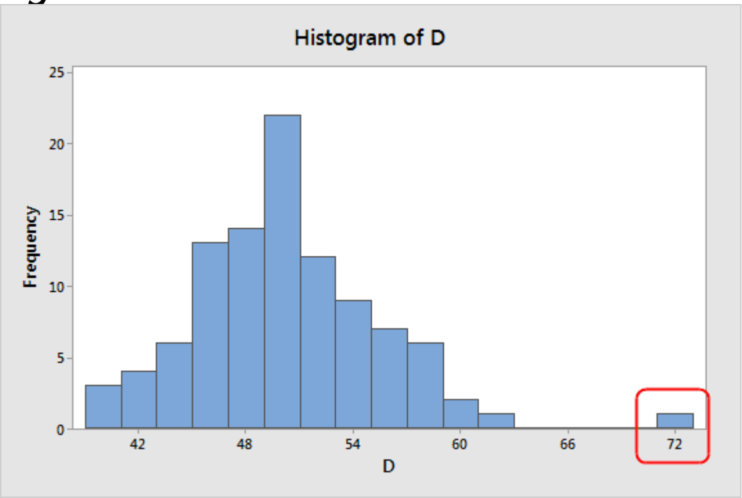
- So, we have established that scatterplots displayed the relationship between two variables.
- Well, time series plots do the same, though one of the continuous variables every time is always time.

Trade
322
317
319
323
327
328



Outliers

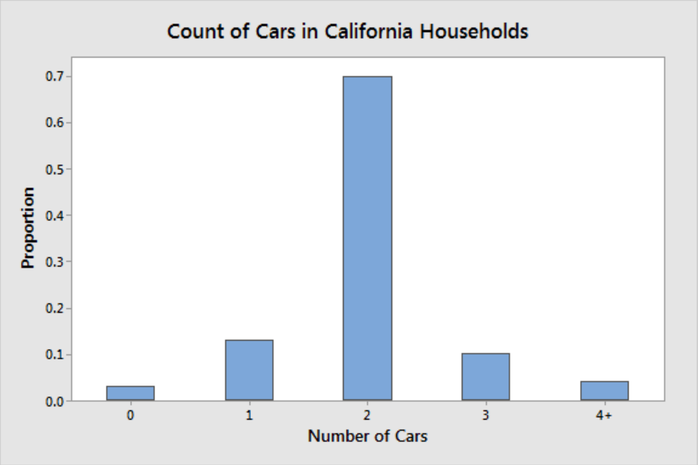
- Outliers are the unusual values that you find in your data, and they can be identified very easily using histograms.
- This is because you will be able to spot any outliers immediately.
- After all, they will be classed as extreme values.



Discrete Numeric Data

- Discrete quantitative data are non-negative integers that can't be divided.
- So, as an example, a single household can have one or two cars, but they can't have 1.6, it's just not possible as there are a finite number of possible values.

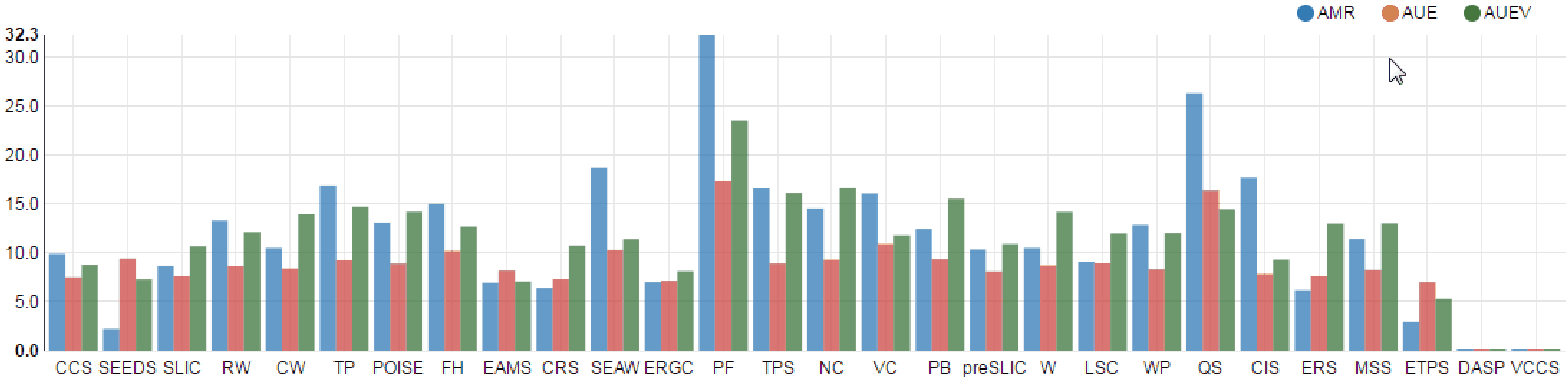
Car Count
2
1
2
2
3



Bar Charts

- These charts are a standard way of presenting data.
- While histograms and bar charts look quite similar, the bars on the former touch, but they are separate on the latter.
- Each bar on a histogram represents a range of values of continuous measurements, while on a bar chart, the bar is one set of discrete values.

Qualitative Data: Categorical, Binary, and Ordinal

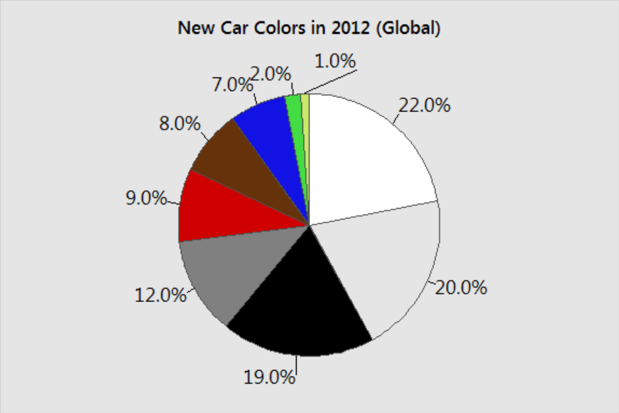


Cardinality

- You tell me… what is it?
- Why is it important?

Categorical Data

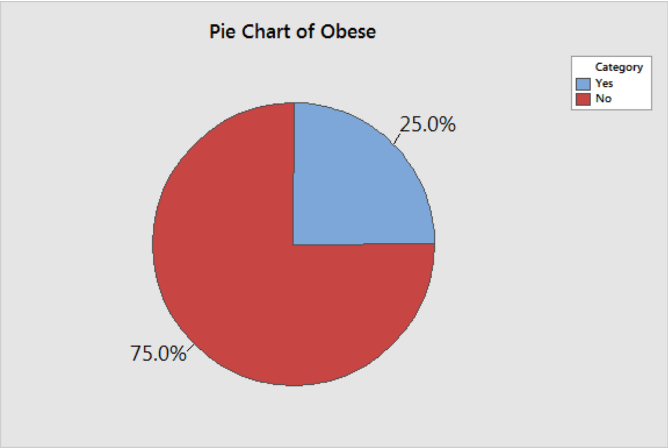
- This type of data is a value that can be placed into several groups based on a characteristic.
- So, you can assign categories to it though these categories do not have a natural order they go in.



Color
White
Silver
Black
Gray
Red

Binary Data

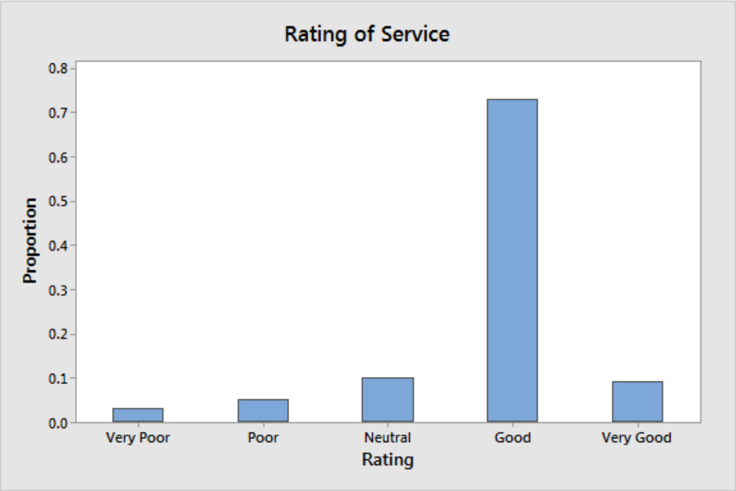
- As it states with the name, binary data can only have two values, so if you can put an observation into only two of your categories, it is classed as binary data.
- It is also referred to as both indicator variables and dichotomous data.



Obese
Yes
No
No
Yes
No

Ordinal Data

- In this type of data there are also three categories; this time, they do have a natural order.
- Ordinal variables can be the overall status (poor to excellent), agreement (strongly disagree to strongly agree), and rank (sporting teams, for example).



Rating
Very Poor
Poor
Neutral
Good
Very Good

Correlation

- The next step in this phase is to build a heatmap…

OR

Build a table of correlations so you can see what variables are related to each other.

Next Steps

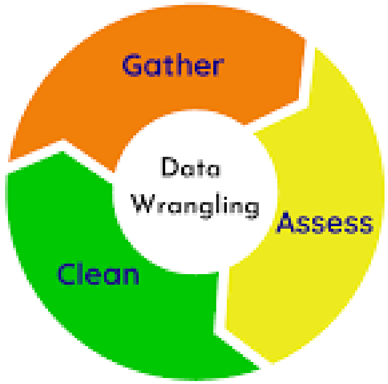
- Is your data wide or long?
- Identify all columns of your data as Qualitative or Quantitative.
- For all Quantitative Wide Continuous Data – do Descriptive statistics for every column.
- Identify and possibly remove outliers
- Graph (use the right graph) every relevant column. (Numeric X-axis and Count Y-Axis)
- Do a correlation analysis
- Find 2 columns that are correlated graph them on the X and Y axis.

5: Data Cleaning and Exploratory Data Analysis

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



"Complete Case Study"

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.

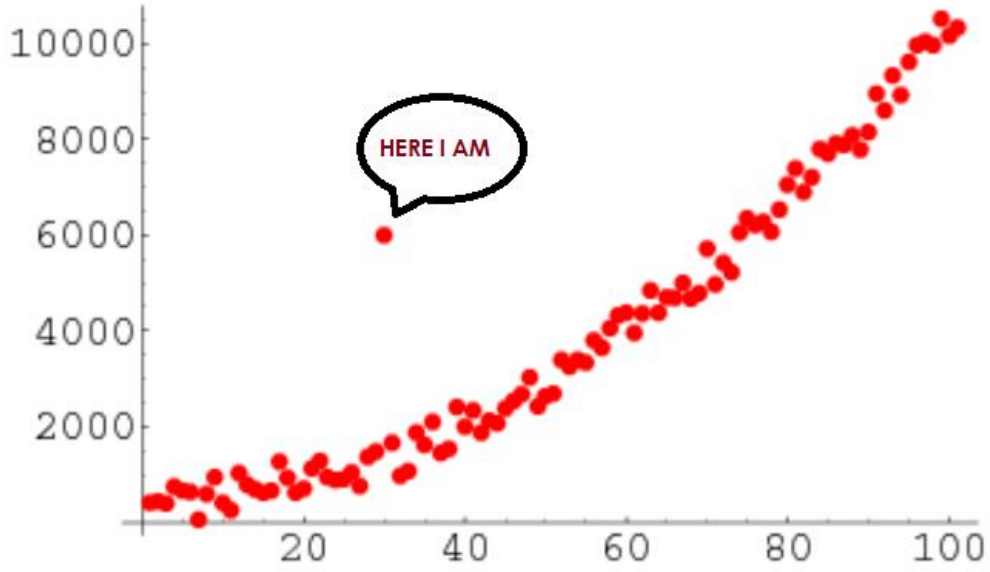
```
mod_data.isnull().sum().sum()
```

```
0
```

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

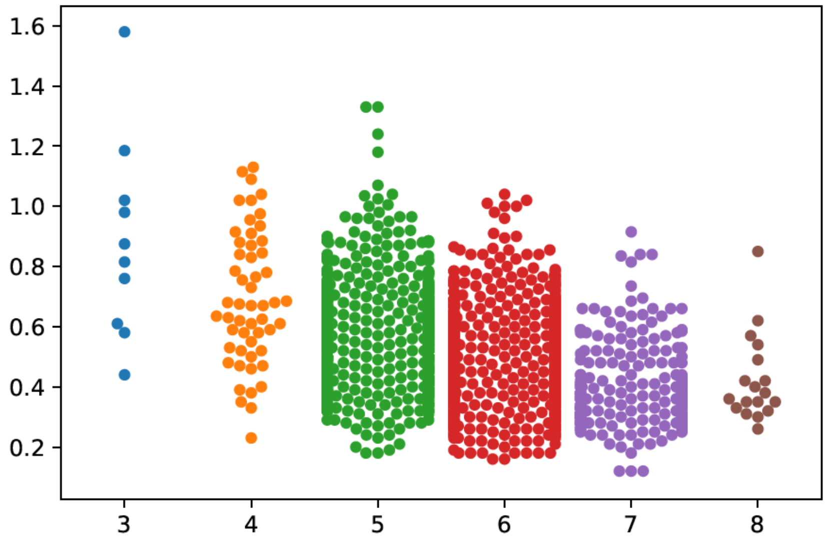
Outliers

- 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)

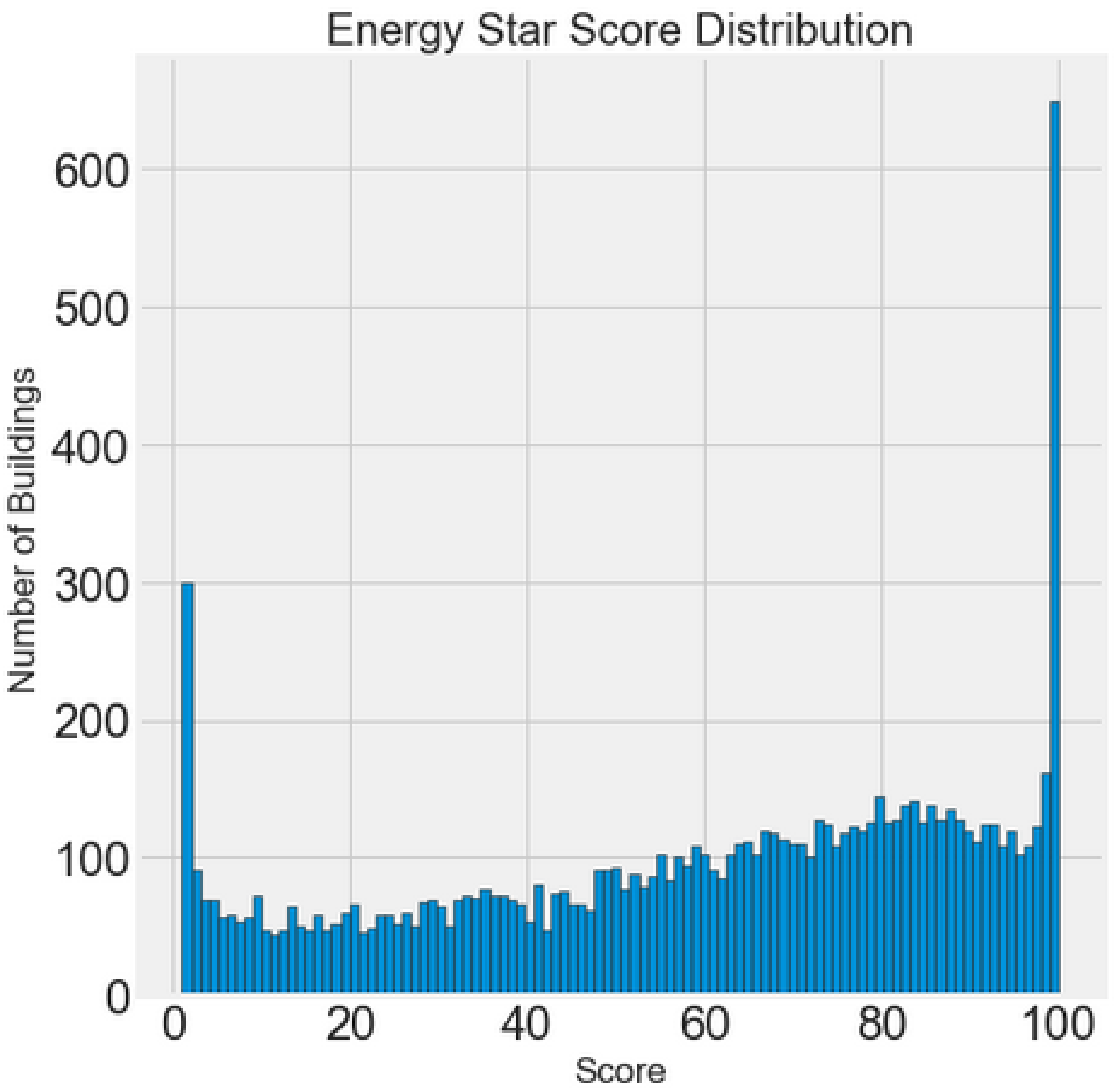
- Exploratory data analysis is an open-ended process, where we perform statistics and make figures to find trends, anomalies, patterns or relationships within the given data.



LEARNING VOYAGE

Univariate analysis (Single variable plots)

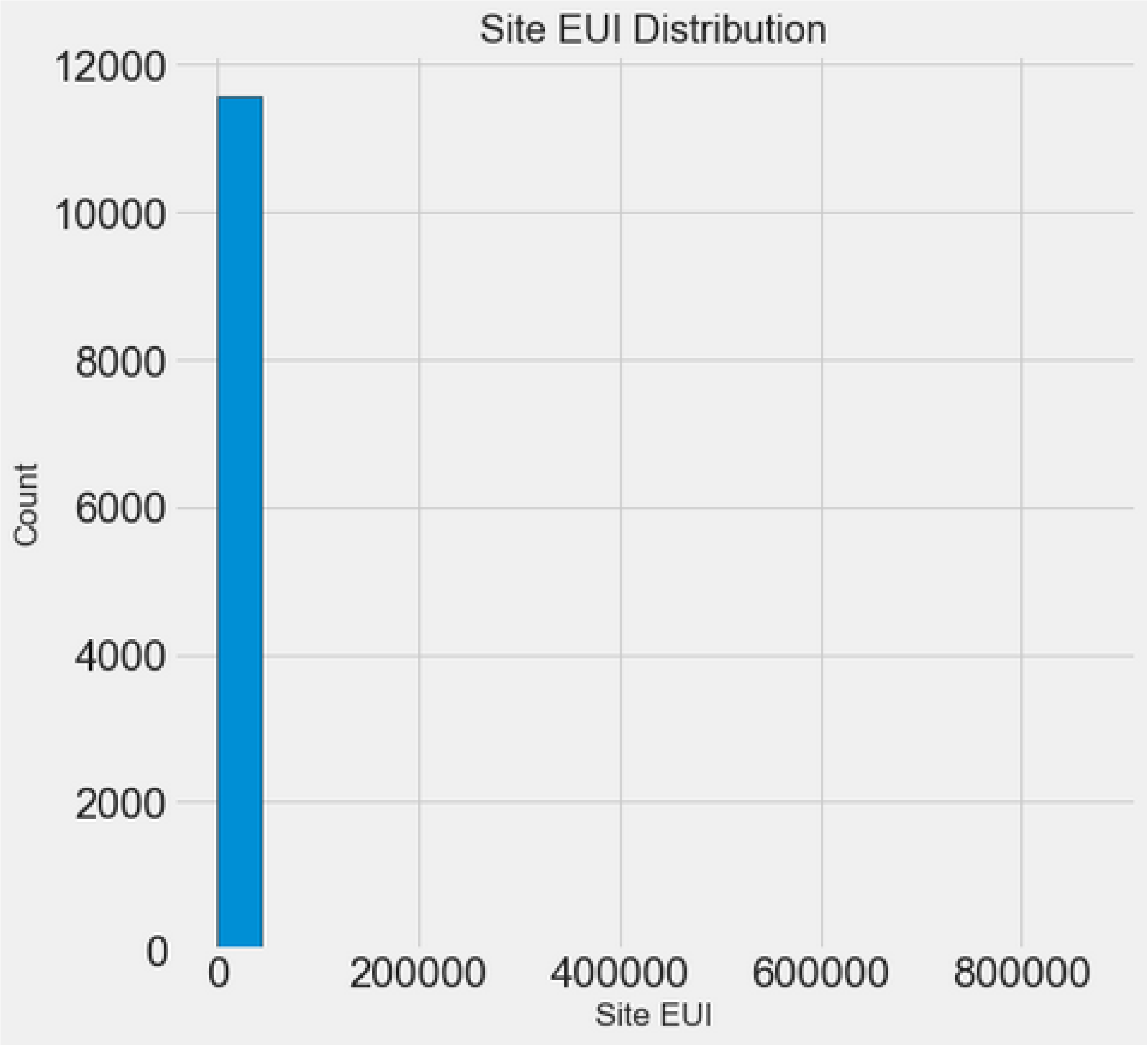
```
plt.figure(figsize=(8,8))

# 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
Score
0
100
200
300
400
500
600
0
20
40
60
80
100

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

```python
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
```

```python
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
```

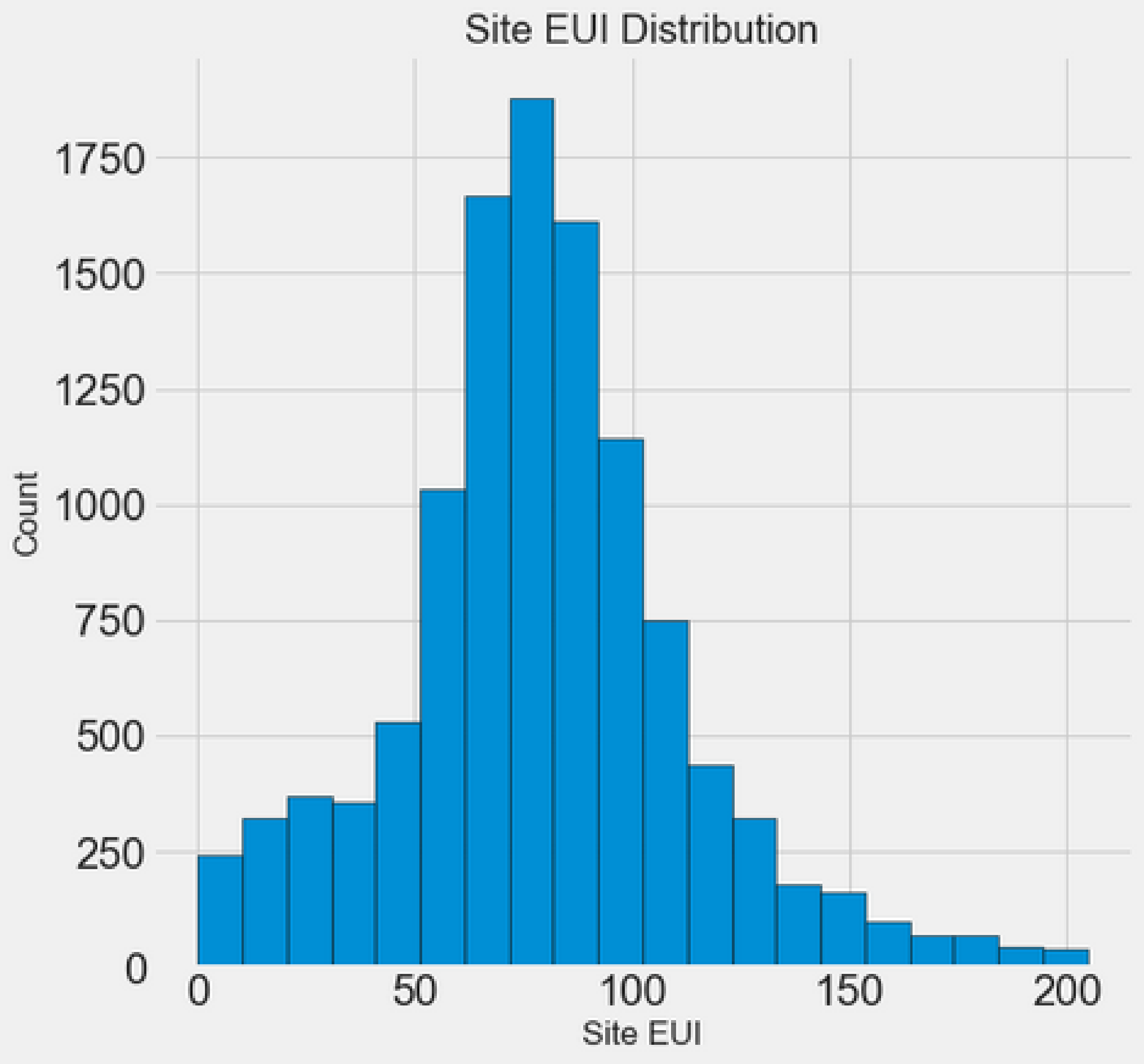
Univariate analysis (Single variable plots)

- 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
1750
1500
1250
1000
750
500
250
0
Count
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

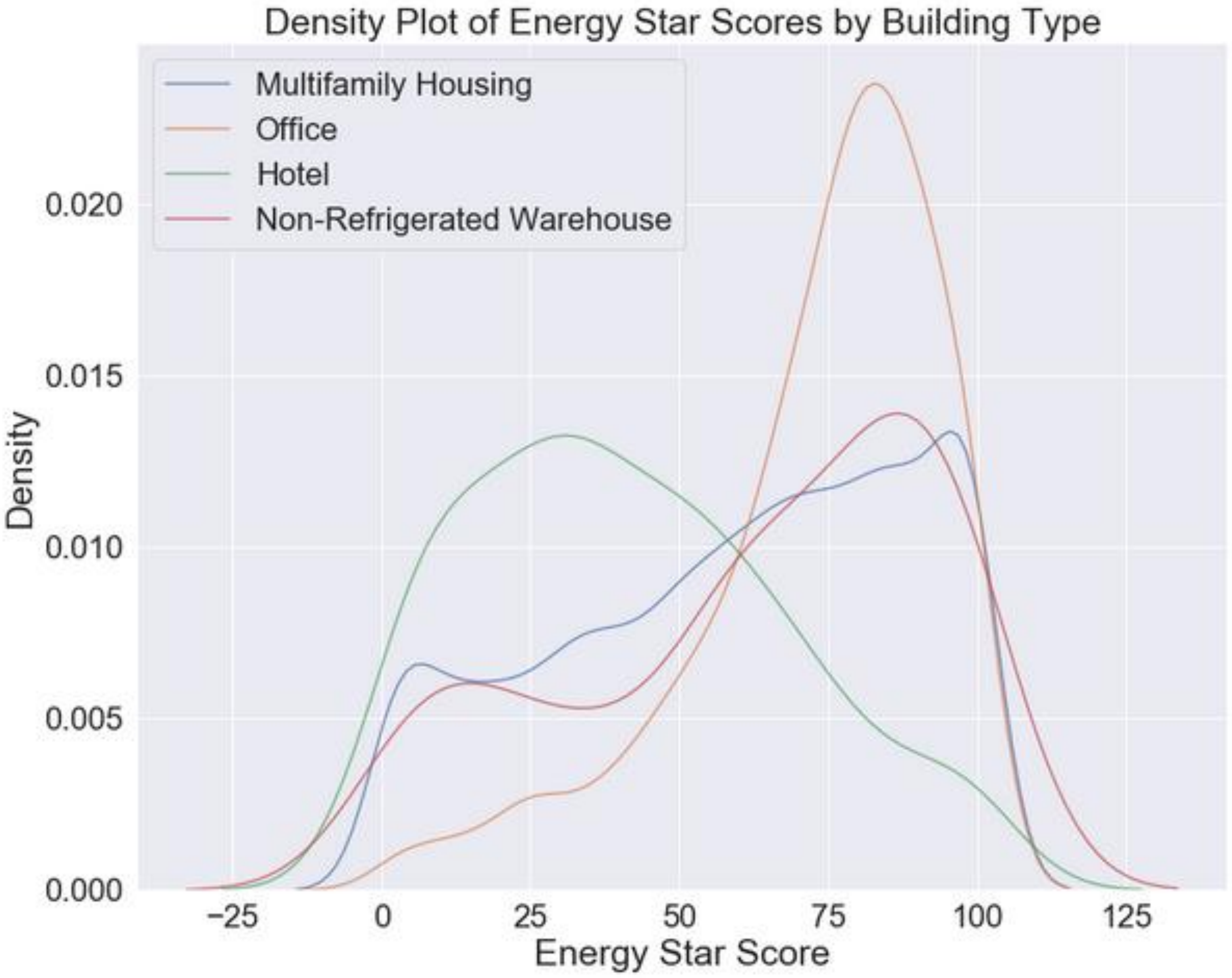
- The first plot we will make shows the distribution of scores by the property type.
- In order to not clutter the plot with large data, we will limit the graph to building types that have more than 100 observations in the dataset.

```
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

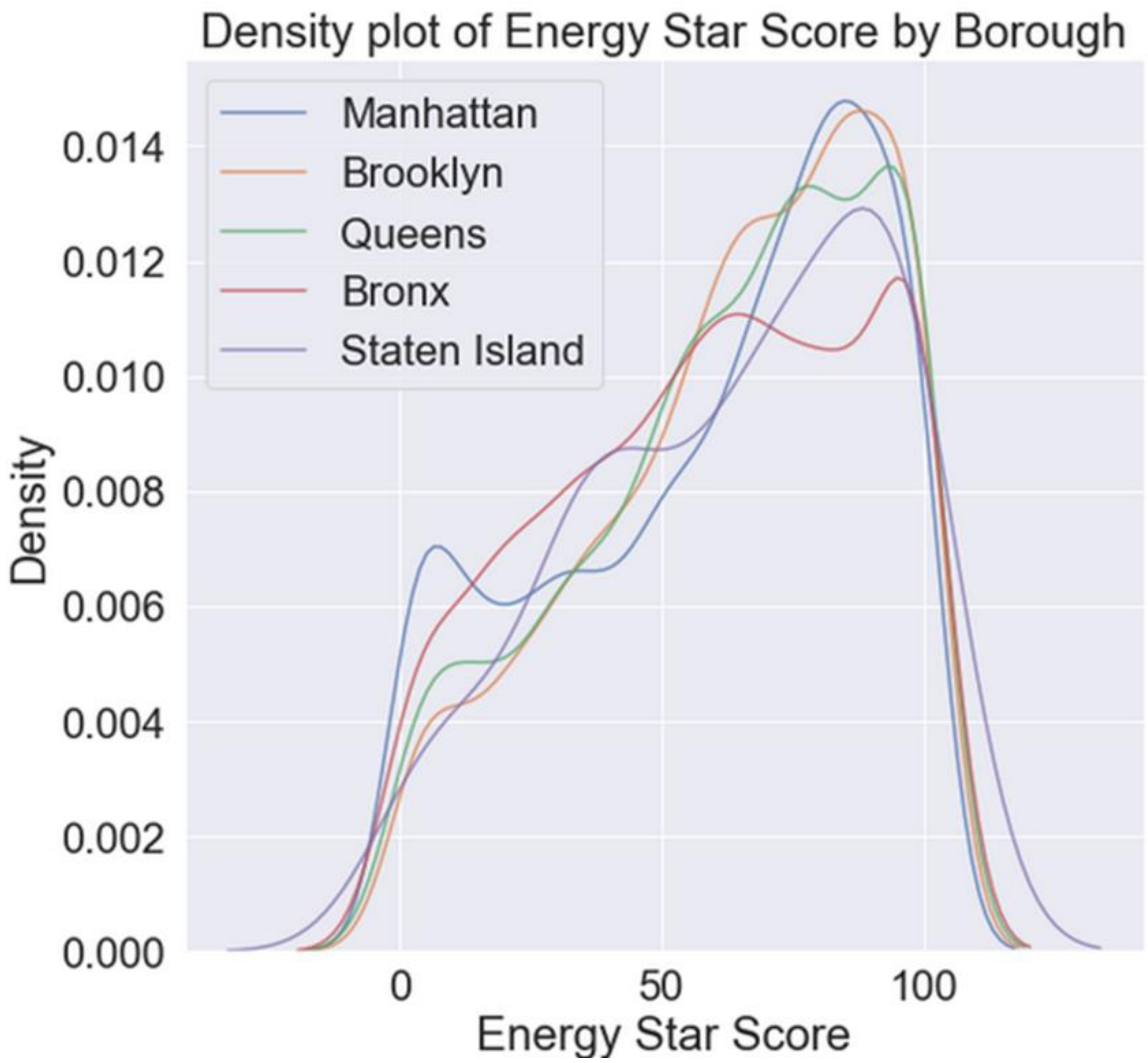
```
# Plot of distribution of scores for building categories

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
0.020
0.015
0.010
0.005
0.000
Density
−25
0
25
50
75
100
125
Energy Star Score

```python
# 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']
```

```python
# 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
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
Density
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
```

```python
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

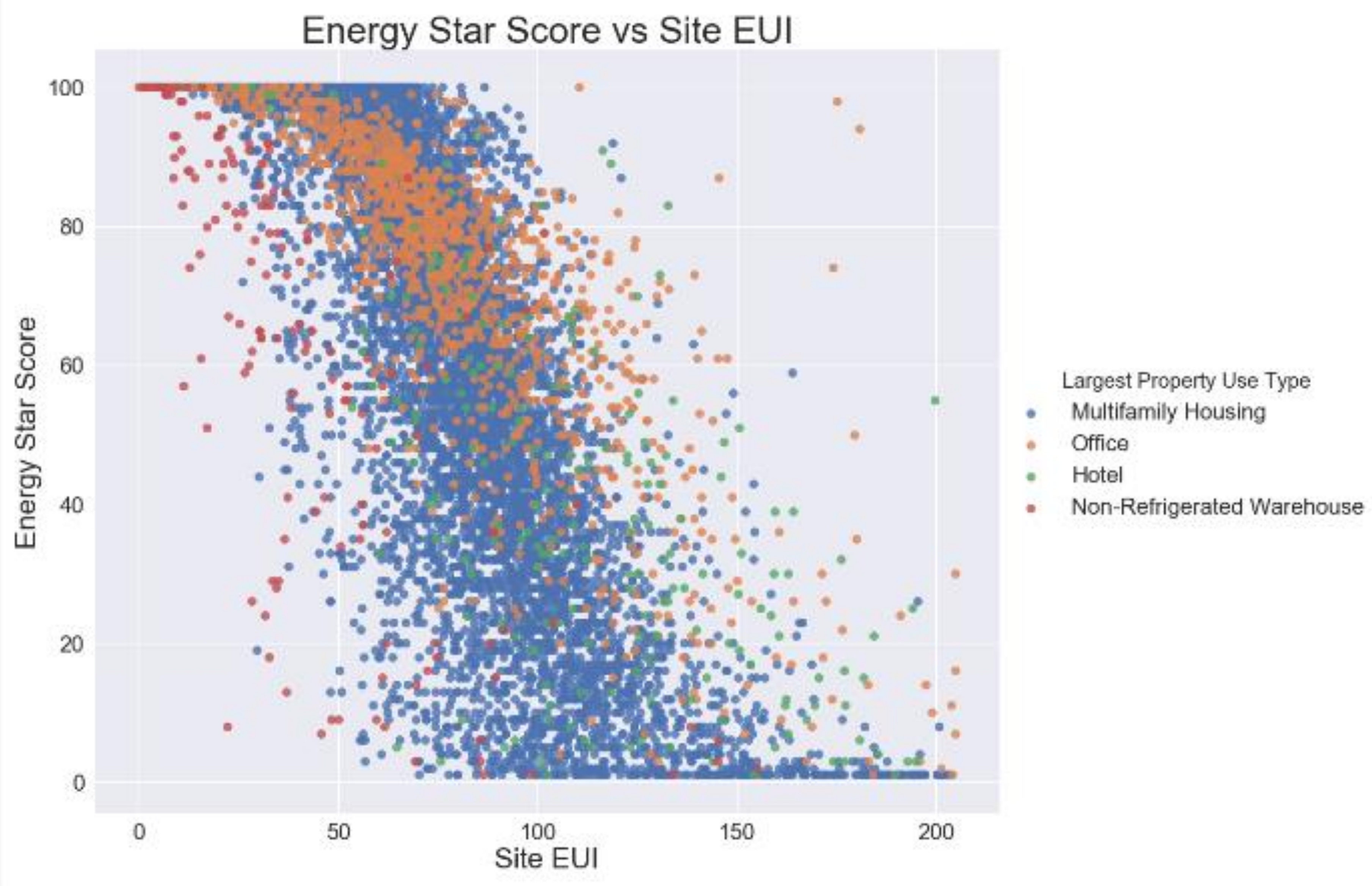
```
# Display most negative correlations
correlations.head(15)
```

```
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
```

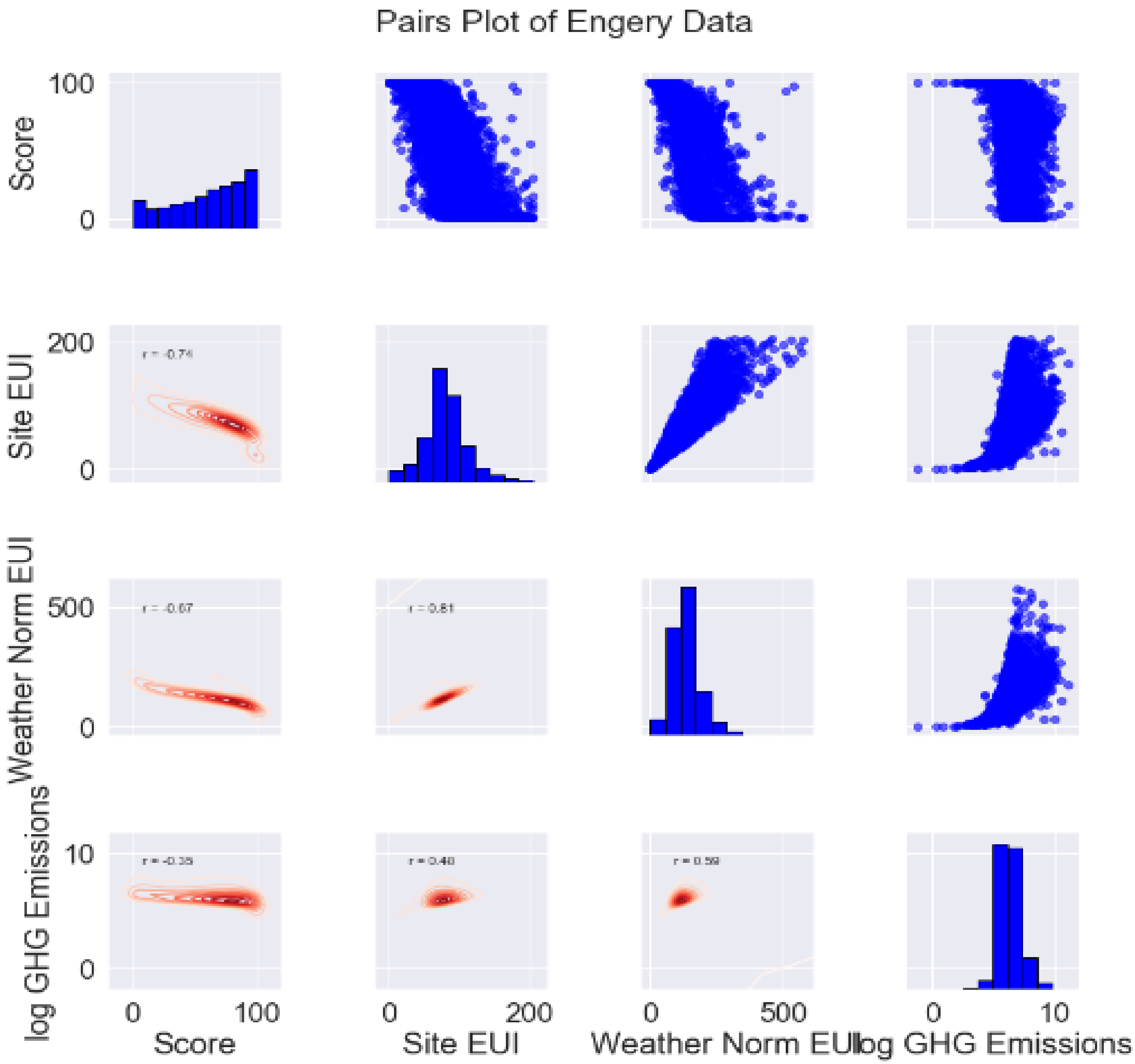

Energy Star Score vs Site EUI
Energy Star Score
100
80
60
40
20
0
0
50
100
150
200
Site EUI
Largest Property Use Type
Multifamily Housing
Office
Hotel
Non-Refrigerated Warehouse

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

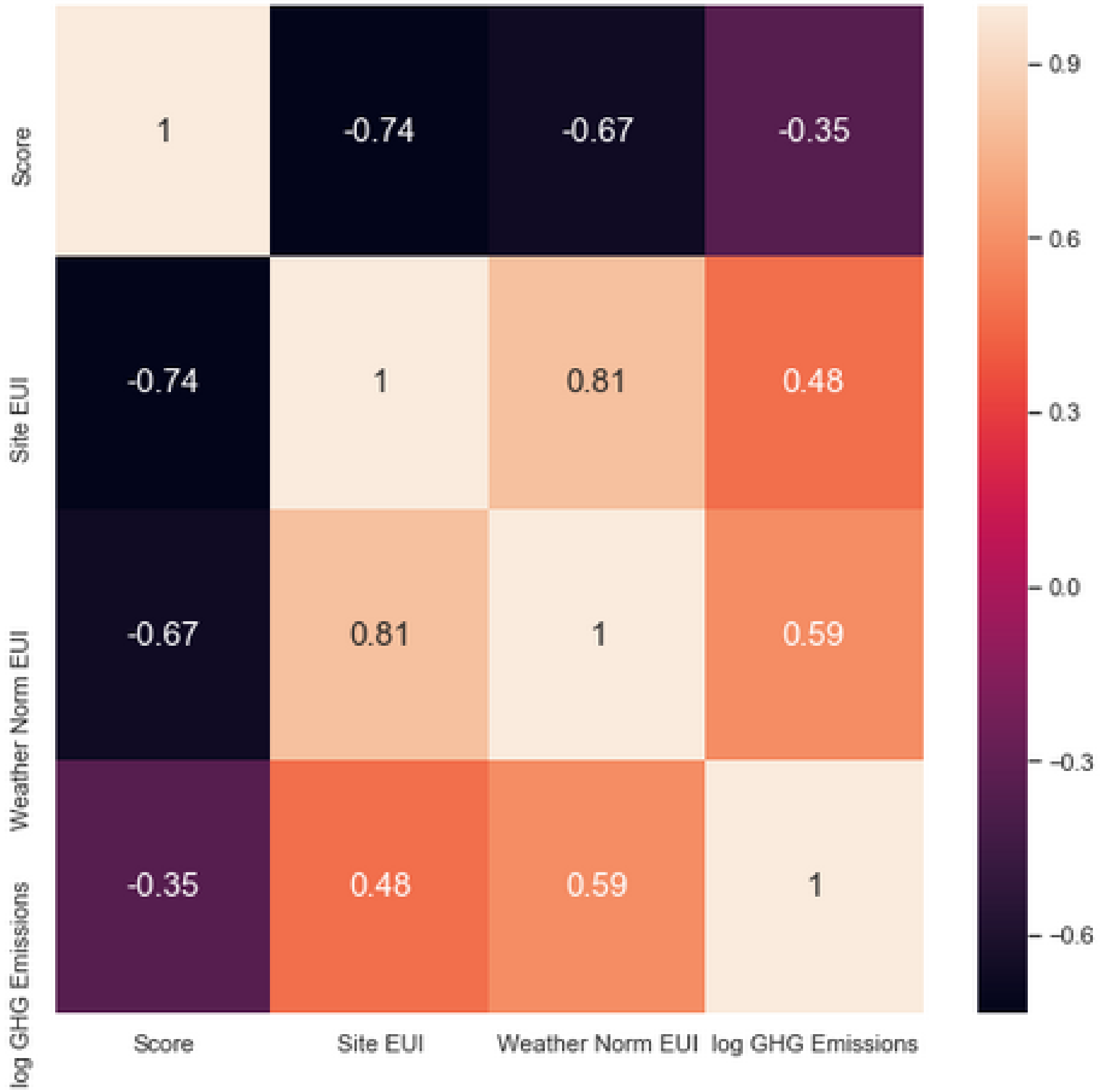
```
# Create the pairgrid object
grid = sns.PairGrid(data = plot_data, size = 3)

# 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
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.

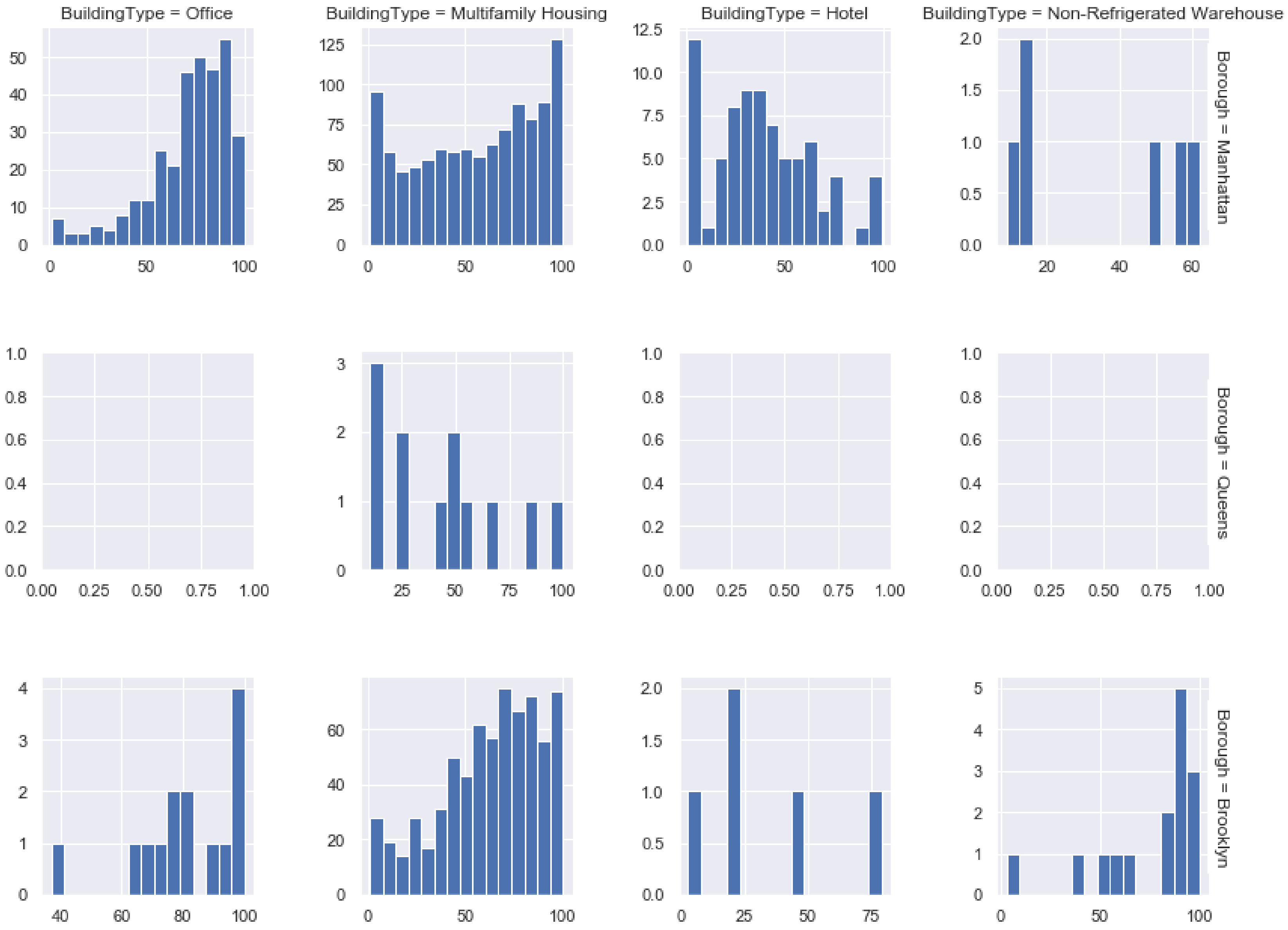
```
multivari_data = data[data['Largest Property Use Type'].isin(building_types) &
                      data['Borough'].isin(boroughs)].dropna()
multivari_data.rename(columns = {'Largest Property Use Type':"BuildingType"},
                      inplace = True)
multivari_data.head()
```

Multivariate Analysis

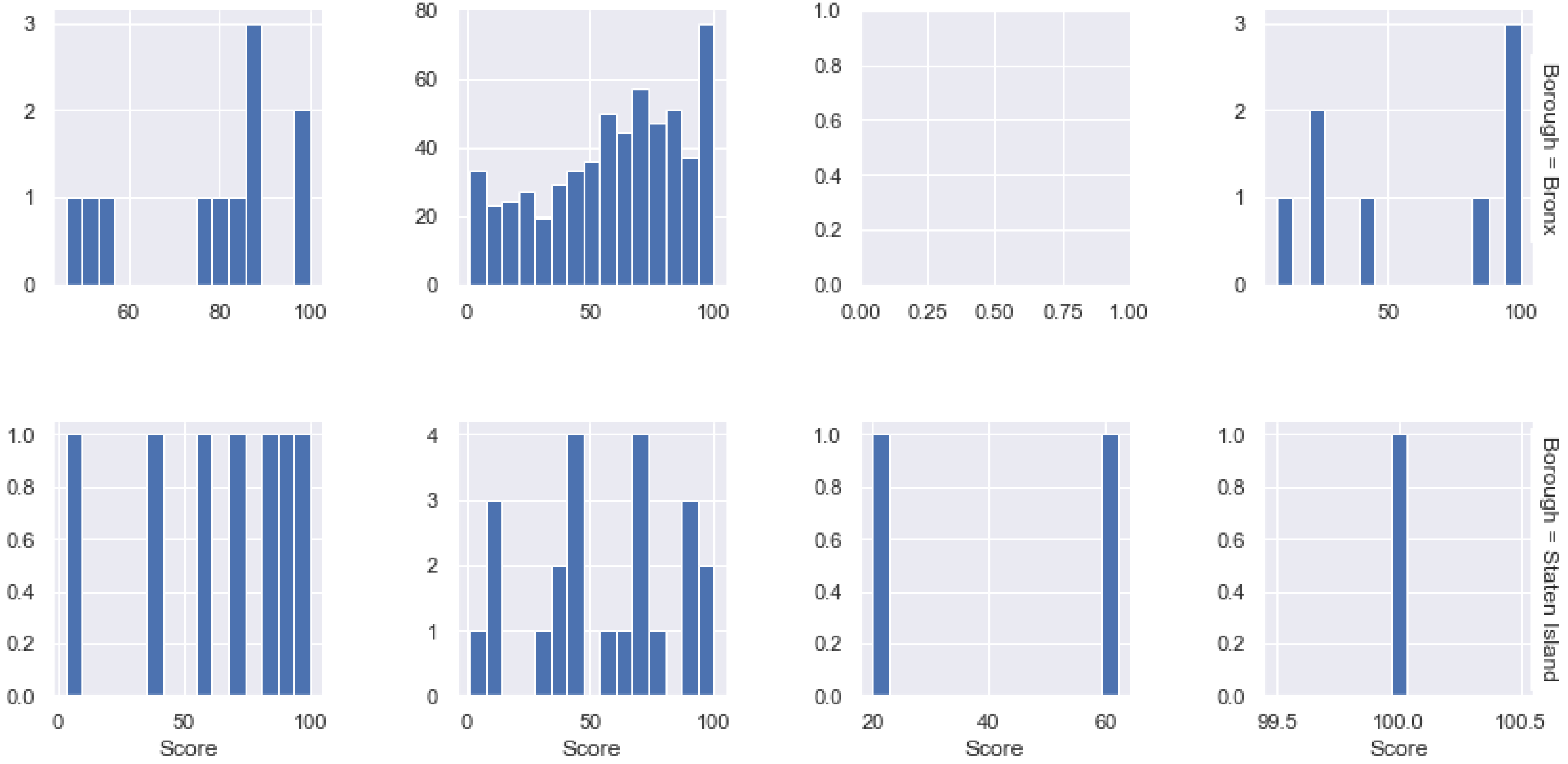
	Order	Property Id	Property Name	Parent Property Id	Parent Property Name	BBL - 10 digits	NYC Borough, Block and Lot (BBL) self-reported	NYC Building Identification Number (BIN)	Address 1 (self-reported)	Postal Code	Street Number
99	102	2605684	Hammer Health Sciences Center	3614737	Columbia University 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

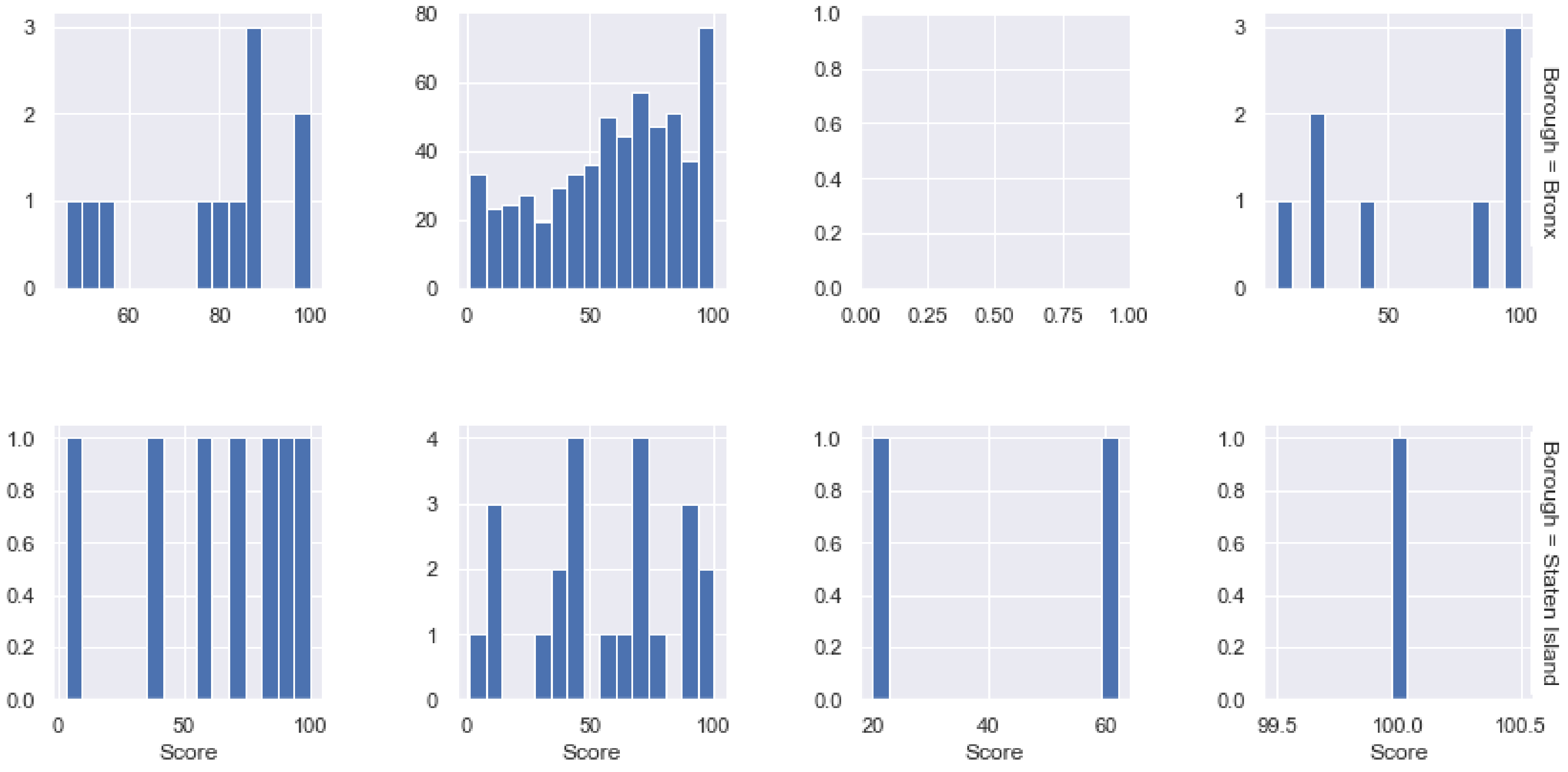
```
plt.figure(figsize=(20,12))
x=sns.FacetGrid(multivari_data, row='Borough',col = 'BuildingType',
                palette='husl',sharex=False,sharey=False, margin_titles=True)
x=x.map(plt.hist, 'Score', bins=15)
x=x.fig.subplots_adjust(wspace=0.5, hspace=0.5)
```


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"