IMPORTING LIBRARIES

```
import pandas as pd
import pandas_profiling
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

READING .CSV FILE

```
data = pd.read_csv('dataset.csv')
```

```
data
```

```
     Relative Compactness  Surface Area  Wall Area  Roof Area  Overall
Heigt  \
0                    0.98         514.5      294.0     110.25
7.0
1                    0.98         514.5      294.0     110.25
7.0
2                    0.98         514.5      294.0     110.25
7.0
3                    0.98         514.5      294.0     110.25
7.0
4                    0.90         563.5      318.5     122.50
7.0
..                    ...           ...        ...        ...
...
763                  0.64         784.0      343.0     220.50
3.5
764                  0.62         808.5      367.5     220.50
3.5
765                  0.62         808.5      367.5     220.50
3.5
766                  0.62         808.5      367.5     220.50
3.5
767                  0.62         808.5      367.5     220.50
3.5

     Orientation  Glazing Area  Glazing Area Distribution  Heating
Load  \
0              2           0.0                          0
15.55
1              3           0.0                          0
15.55
2              4           0.0                          0
15.55
3              5           0.0                          0
15.55
```

```
4              2           0.0                           0
20.84
..           ...           ...                         ...           ..
.
763            5           0.4                           5
17.88
764            2           0.4                           5
16.54
765            3           0.4                           5
16.44
766            4           0.4                           5
16.48
767            5           0.4                           5
16.64

     Cooling Load
0           21.33
1           21.33
2           21.33
3           21.33
4           28.28
..            ...
763         21.40
764         16.88
765         17.11
766         16.61
767         16.03

[768 rows x 10 columns]
```

FIRST FIVE ROWS OF THE DATASET

```
data.head()

   Relative Compactness  Surface Area  Wall Area  Roof Area  Overall
Heigt  \
0                  0.98         514.5      294.0     110.25
7.0
1                  0.98         514.5      294.0     110.25
7.0
2                  0.98         514.5      294.0     110.25
7.0
3                  0.98         514.5      294.0     110.25
7.0
4                  0.90         563.5      318.5     122.50
7.0

   Orientation  Glazing Area  Glazing Area Distribution  Heating Load
\
0            2           0.0                           0         15.55
```

```
1            3           0.0                           0         15.55

2            4           0.0                           0         15.55

3            5           0.0                           0         15.55

4            2           0.0                           0         20.84


   Cooling Load
0         21.33
1         21.33
2         21.33
3         21.33
4         28.28
```

LAST FIVE ROWS OF THE DATASET

```
data.tail()
```

```
     Relative Compactness  Surface Area  Wall Area  Roof Area  Overall
Heigt  \
763                  0.64         784.0      343.0      220.5
3.5
764                  0.62         808.5      367.5      220.5
3.5
765                  0.62         808.5      367.5      220.5
3.5
766                  0.62         808.5      367.5      220.5
3.5
767                  0.62         808.5      367.5      220.5
3.5

     Orientation  Glazing Area  Glazing Area Distribution  Heating
Load  \
763            5           0.4                          5
17.88
764            2           0.4                          5
16.54
765            3           0.4                          5
16.44
766            4           0.4                          5
16.48
767            5           0.4                          5
16.64

     Cooling Load
763         21.40
764         16.88
```

```
765         17.11
766         16.61
767         16.03
```

ANALYSING THE SHAPE OF THE DATASET

```
print("shape of the dataset :", data.shape)
print("Number of rows       :", data.shape[0])
print("Number of columns    :", data.shape[1])
```

```
shape of the dataset : (768, 10)
Number of rows       : 768
Number of columns    : 10
```

The data.info() function provides information about a dataset, including the data types and number of non-null values for each column.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Relative Compactness       768 non-null    float64
 1   Surface Area               768 non-null    float64
 2   Wall Area                  768 non-null    float64
 3   Roof Area                  768 non-null    float64
 4   Overall Heigt              768 non-null    float64
 5   Orientation                768 non-null    int64  
 6   Glazing Area               768 non-null    float64
 7   Glazing Area Distribution  768 non-null    int64  
 8   Heating Load               768 non-null    float64
 9   Cooling Load               768 non-null    float64
dtypes: float64(8), int64(2)
memory usage: 60.1 KB
```

The data.describe() function provides statistical information about a dataset, including measures of central tendency, dispersion, and distribution for numerical columns.

```
data.describe()
```

```
       Relative Compactness  Surface Area   Wall Area   Roof Area  \
count            768.000000    768.000000  768.000000  768.000000   
mean               0.764167    671.708333  318.500000  176.604167   
std                0.105777     88.086116   43.626481   45.165950   
min                0.620000    514.500000  245.000000  110.250000   
25%                0.682500    606.375000  294.000000  140.875000   
50%                0.750000    673.750000  318.500000  183.750000   
75%                0.830000    741.125000  343.000000  220.500000   
max                0.980000    808.500000  416.500000  220.500000   
```

```
       Overall Heigt  Orientation  Glazing Area  Glazing Area
Distribution  \
count      768.00000   768.000000    768.000000
768.00000
mean         5.25000     3.500000      0.234375
2.81250
std          1.75114     1.118763      0.133221
1.55096
min          3.50000     2.000000      0.000000
0.00000
25%          3.50000     2.750000      0.100000
1.75000
50%          5.25000     3.500000      0.250000
3.00000
75%          7.00000     4.250000      0.400000
4.00000
max          7.00000     5.000000      0.400000
5.00000

       Heating Load  Cooling Load
count    768.000000    768.000000
mean      22.307201     24.587760
std       10.090196      9.513306
min        6.010000     10.900000
25%       12.992500     15.620000
50%       18.950000     22.080000
75%       31.667500     33.132500
max       43.100000     48.030000
```

The data.isnull().sum() function returns the number of missing values (null values) in each column of a dataset

```
data.isnull().sum()
```

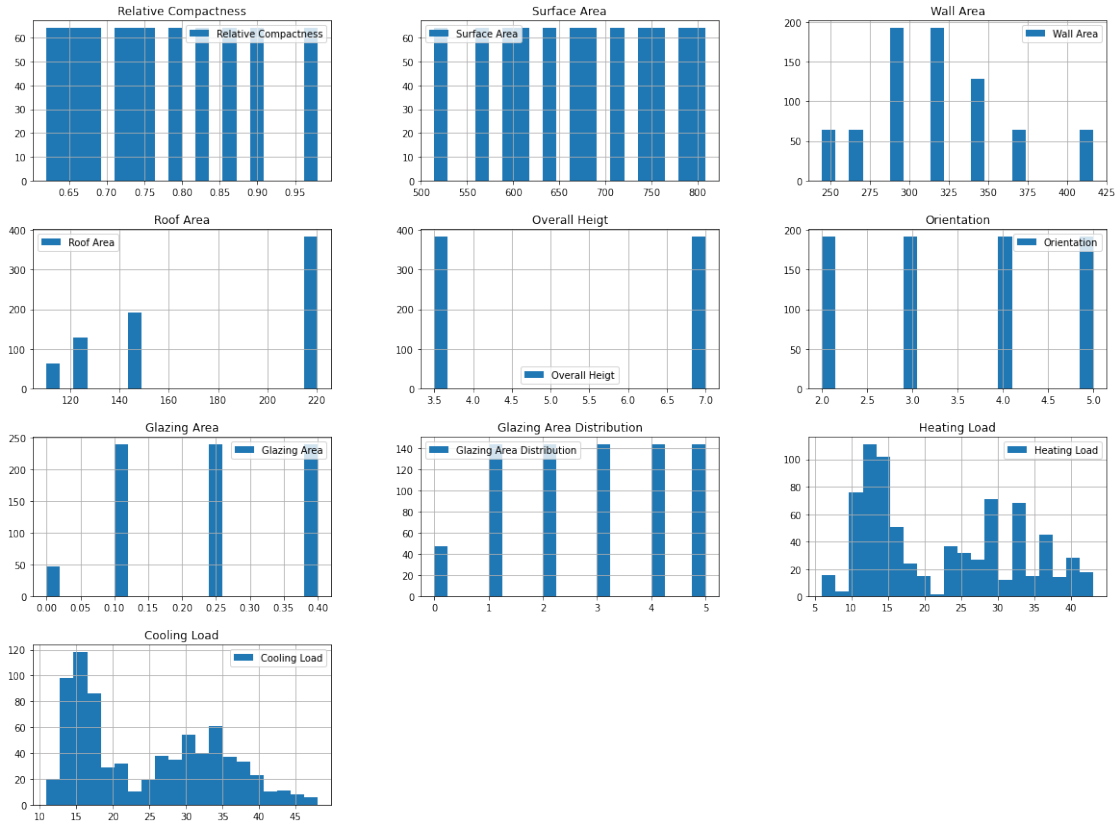
```
Relative Compactness         0
Surface Area                 0
Wall Area                    0
Roof Area                    0
Overall Heigt                0
Orientation                  0
Glazing Area                 0
Glazing Area Distribution    0
Heating Load                 0
Cooling Load                 0
dtype: int64
```

Data Visualization

PLOTTING HISTOGRAM

```
data.hist(figsize = (20,15), bins=20, legend=True)
```

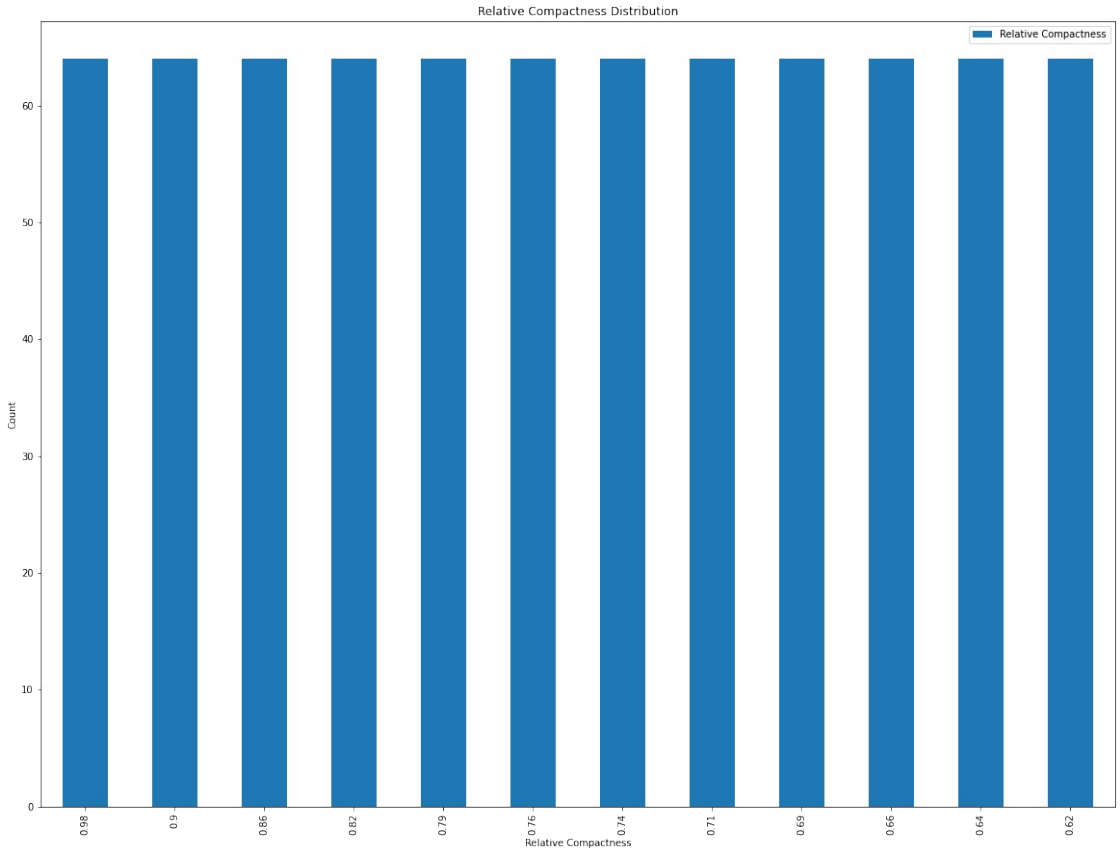
```
array([[<Axes: title={'center': 'Relative Compactness'}>,
        <Axes: title={'center': 'Surface Area'}>,
        <Axes: title={'center': 'Wall Area'}>],
       [<Axes: title={'center': 'Roof Area'}>,
        <Axes: title={'center': 'Overall Heigt'}>,
        <Axes: title={'center': 'Orientation'}>],
       [<Axes: title={'center': 'Glazing Area'}>,
        <Axes: title={'center': 'Glazing Area Distribution'}>,
        <Axes: title={'center': 'Heating Load'}>],
       [<Axes: title={'center': 'Cooling Load'}>, <Axes: >, <Axes:
>]],
      dtype=object)
```

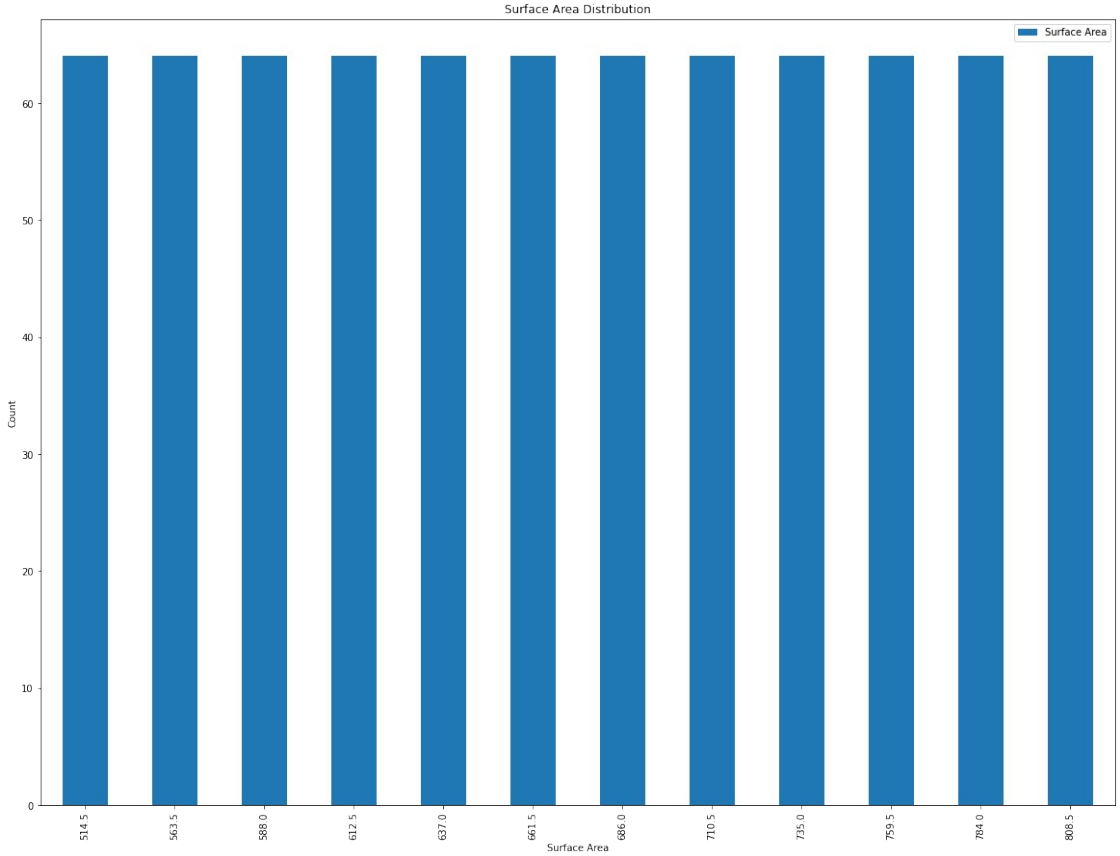


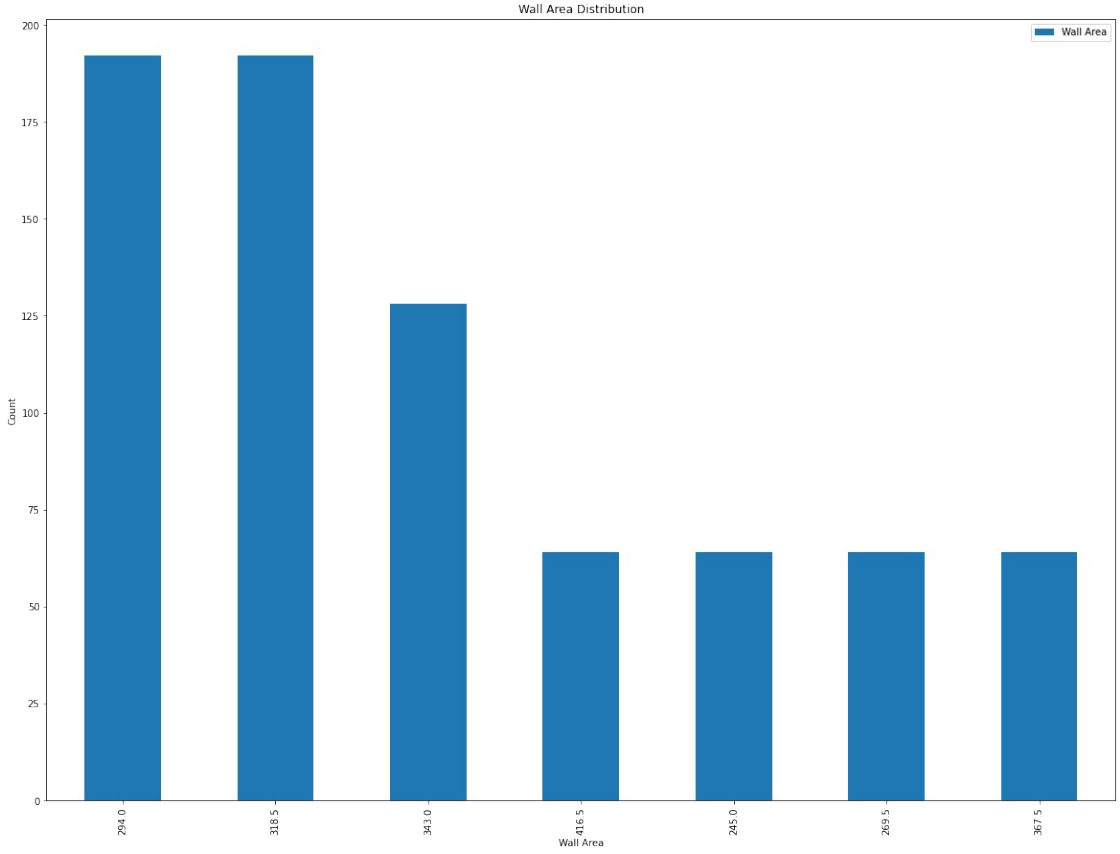
BAR CHART

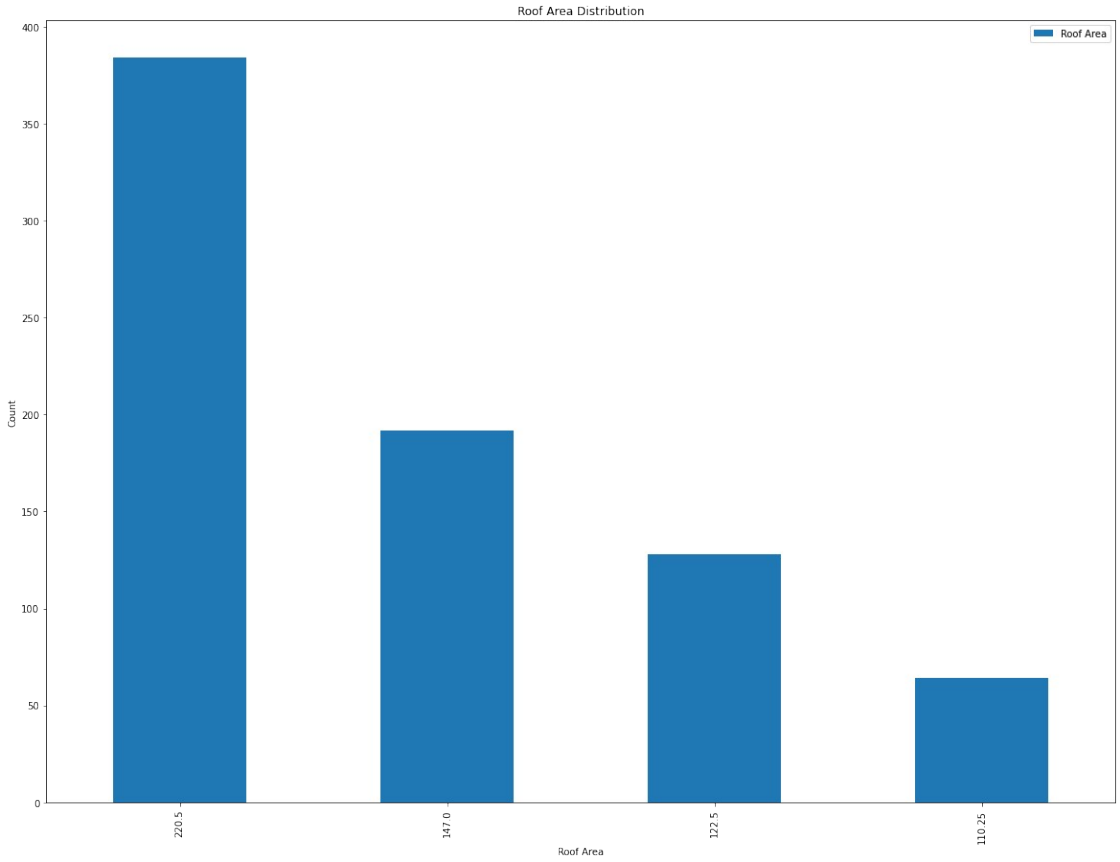
```python
import pandas as pd
import matplotlib.pyplot as plt

# read the data from CSV file into a pandas DataFrame
df = pd.read_csv('dataset.csv')

# iterate through the columns of the DataFrame and create a bar plot
for each column
for col in df.columns:
    fig, ax = plt.subplots(figsize=(20, 15))
```

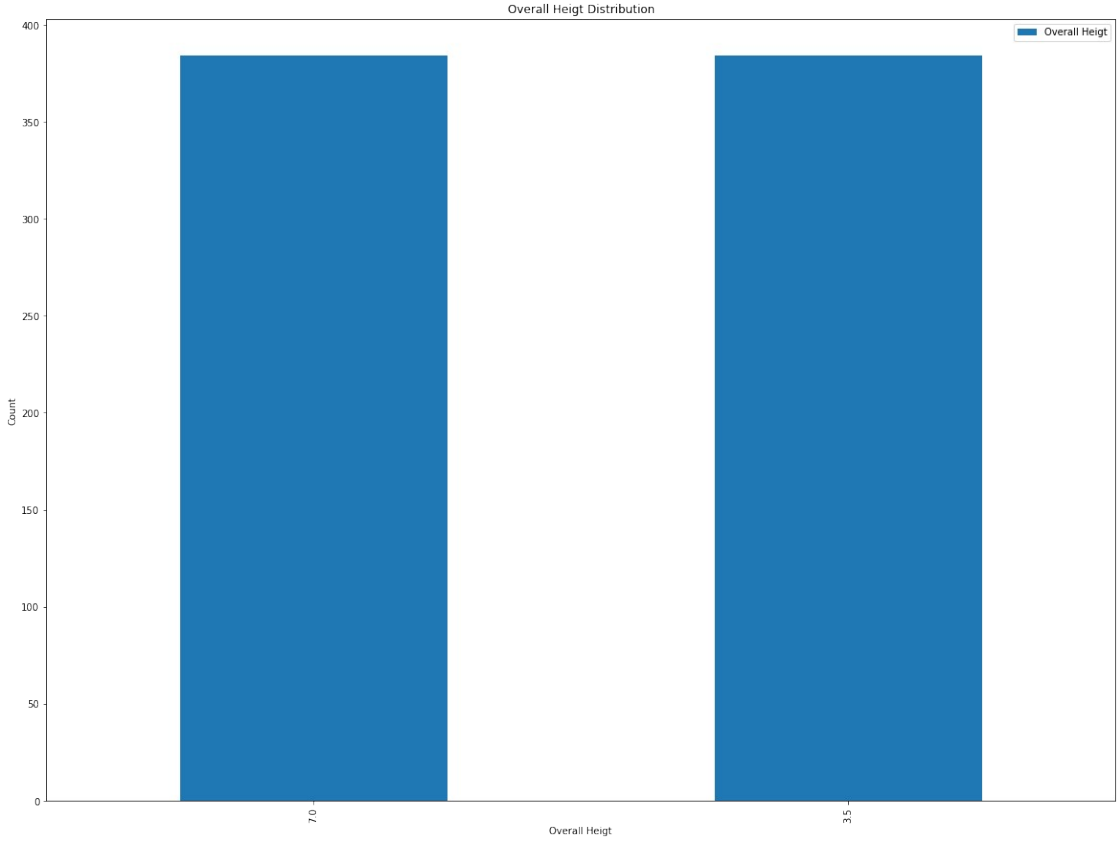
```
df[col].value_counts().plot(kind='bar', ax=ax)
ax.set_xlabel(col)
ax.set_ylabel('Count')
ax.set_title(f'{col} Distribution')
ax.legend()
plt.show()
```

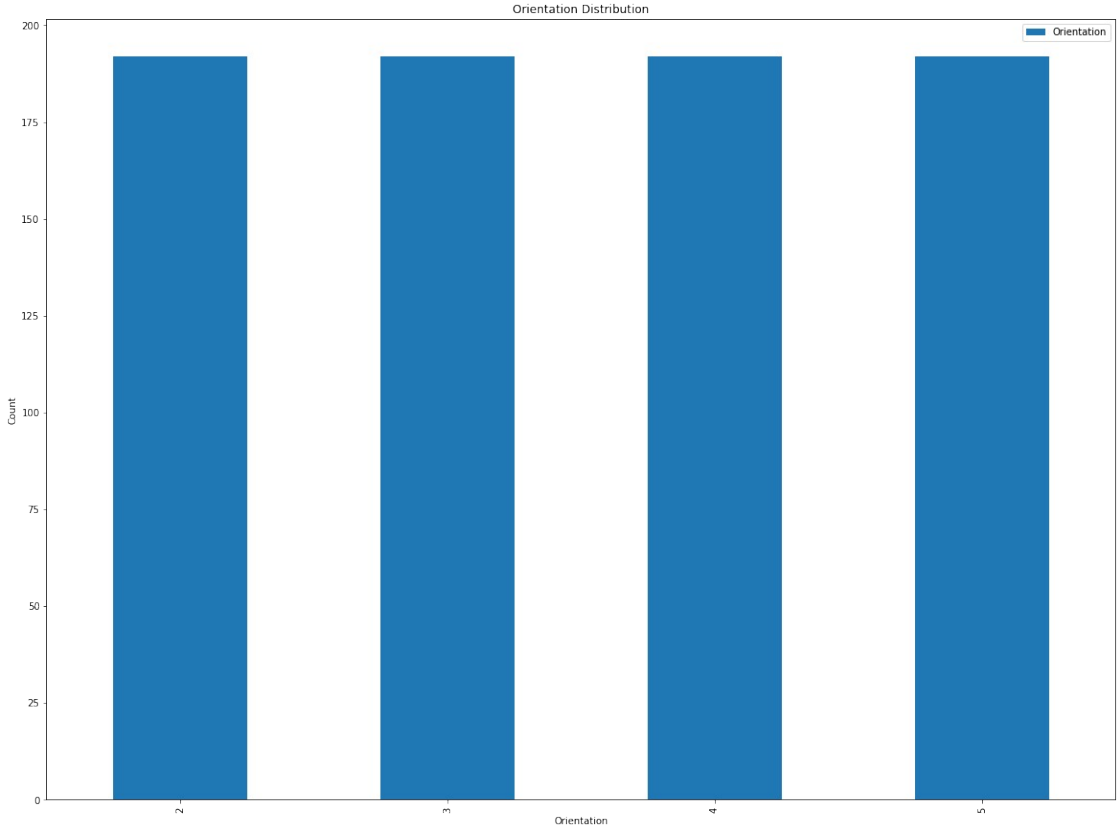


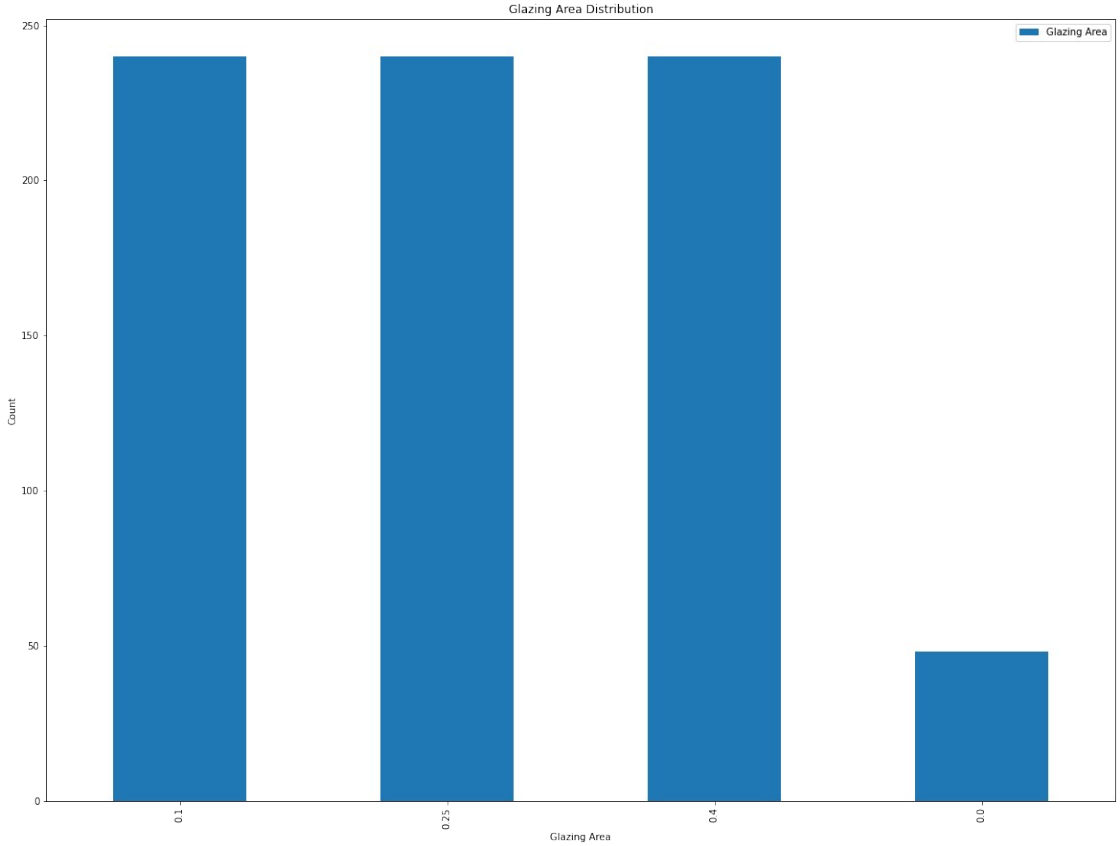

Surface Area Distribution
Surface Area
Count
0
10
20
30
40
50
60
514.5
563.5
588.0
612.5
637.0
661.5
686.0
710.5
735.0
759.5
784.0
808.5
Surface Area

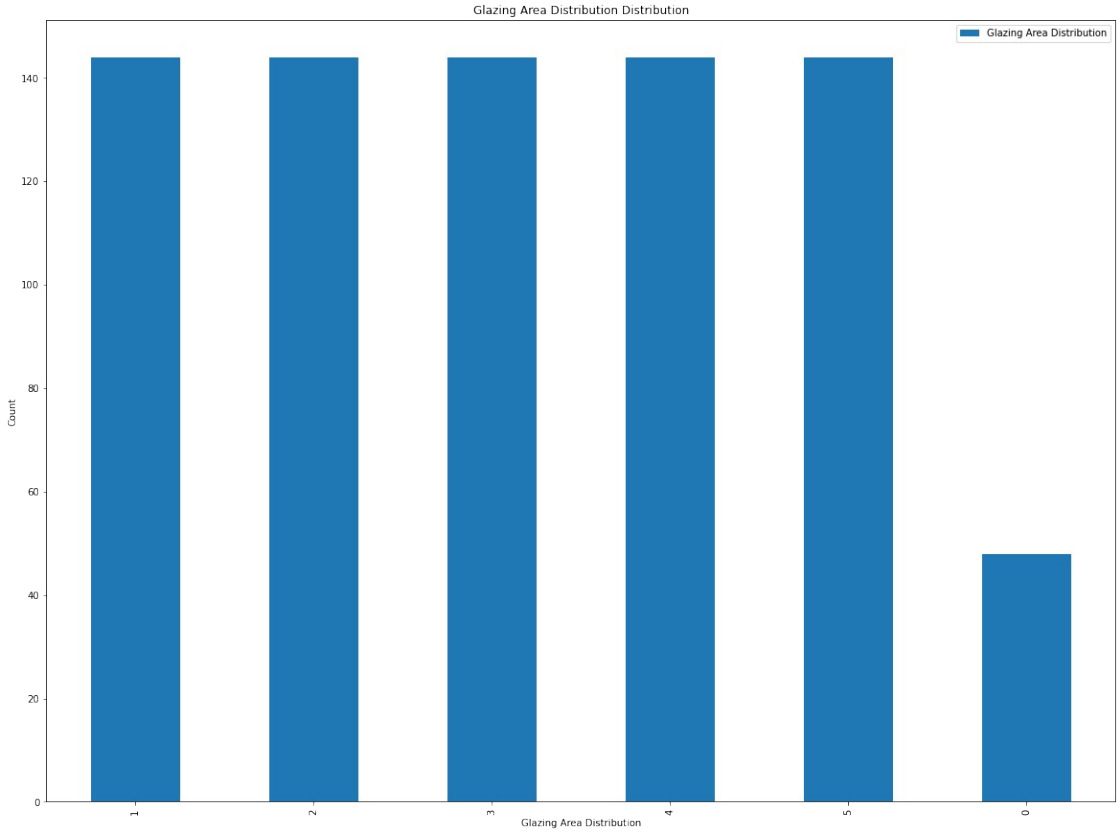

Wall Area Distribution
Wall Area
Count
200
175
150
125
100
75
50
25
0
294.0
318.5
343.0
416.5
245.0
269.5
367.5
Wall Area

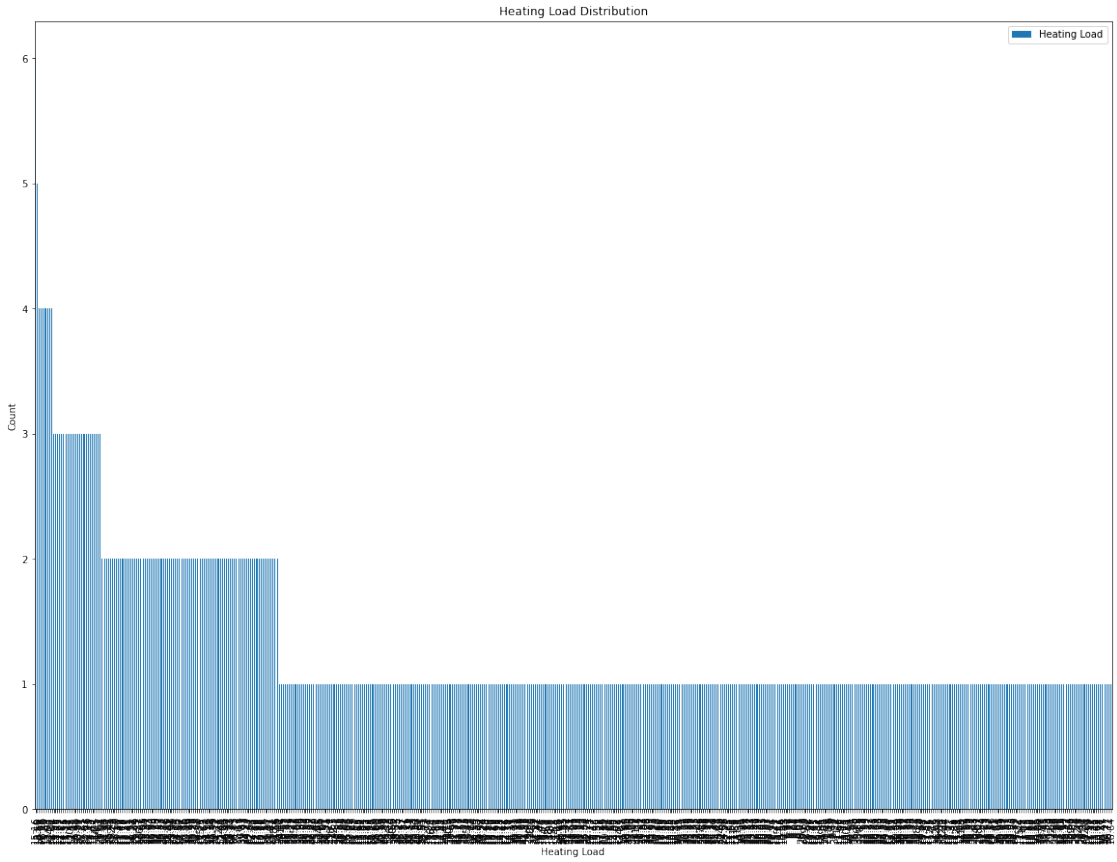

Roof Area Distribution
Roof Area
Count
Roof Area
400
350
300
250
200
150
100
50
0
220.5
147.0
122.5
110.25

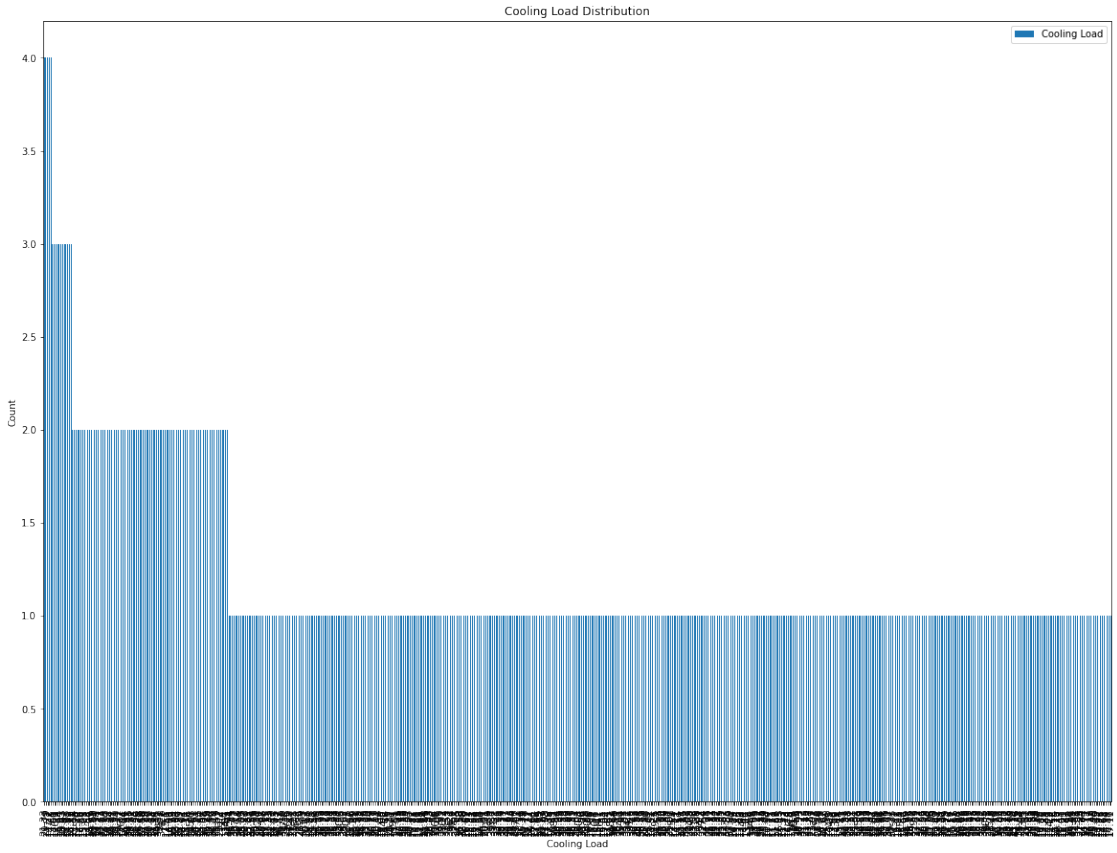

Overall Heigt Distribution
Overall Heigt
Count
Overall Heigt
400
350
300
250
200
150
100
50
0
7.0
3.5


Orientation Distribution
Orientation
Count
Orientation
200
175
150
125
100
75
50
25
0
2
3
4
5


Glazing Area Distribution
Glazing Area
Count
Glazing Area
0.1
0.25
0.4
0.0
0
50
100
150
200
250


Glazing Area Distribution Distribution
Glazing Area Distribution
Count
Glazing Area Distribution
1
2
3
4
5
0
0
20
40
60
80
100
120
140

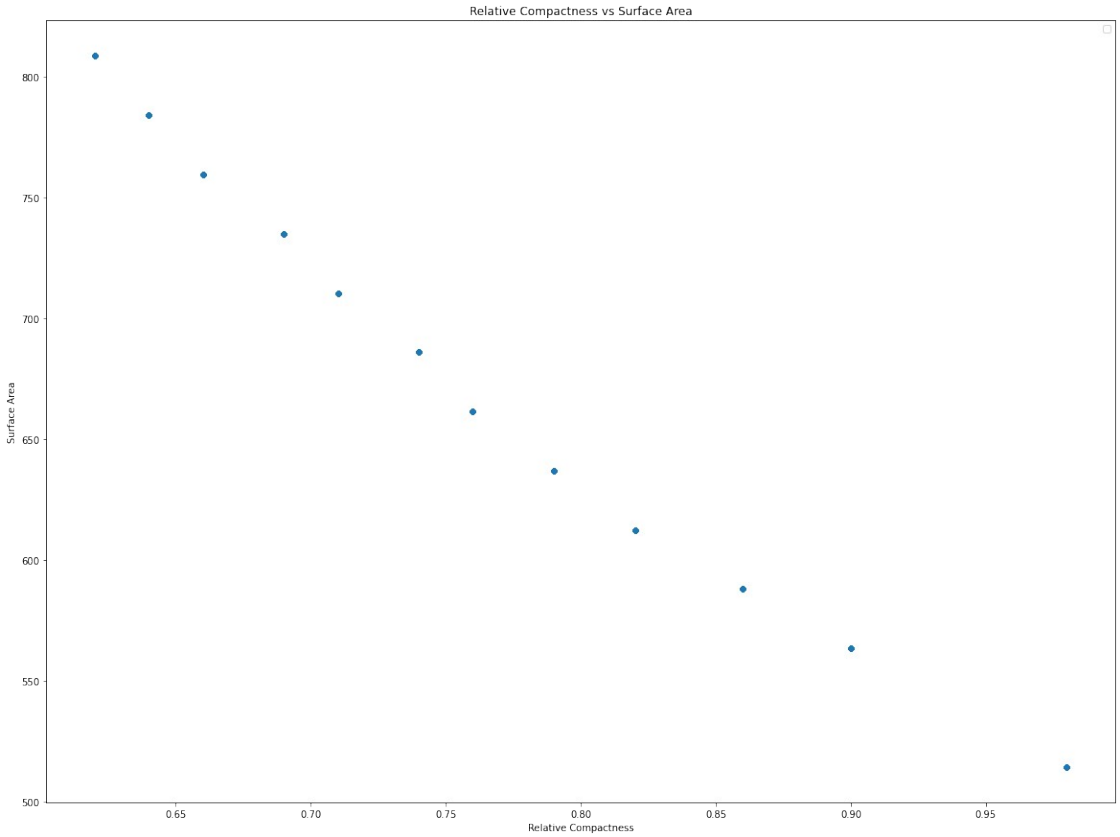

Heating Load Distribution
Heating Load
Count
0
1
2
3
4
5
6
Heating Load



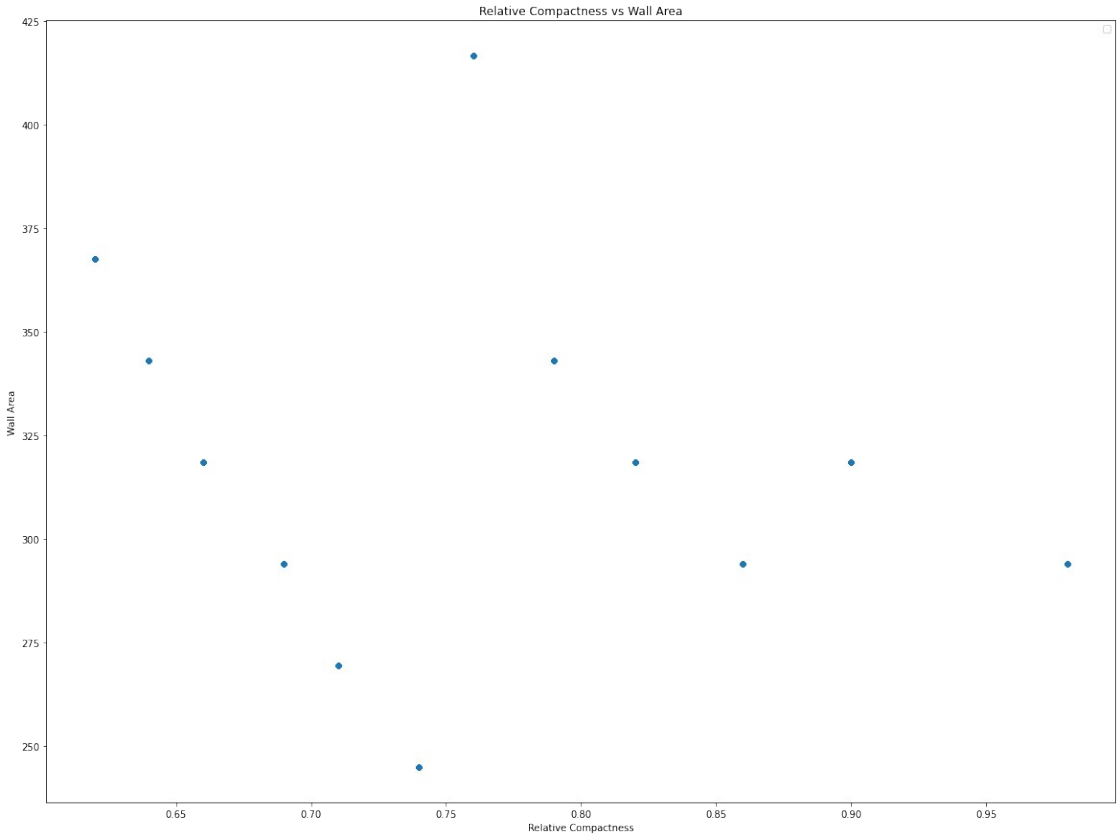
SCATTER PLOT

```
# read the data from CSV file into a pandas DataFrame
df = pd.read_csv('dataset.csv')

# iterate through pairs of columns in the DataFrame and create a
scatter plot for each pair
for i, col1 in enumerate(df.columns):
    for j, col2 in enumerate(df.columns):
        if i < j:
            fig, ax = plt.subplots(figsize=(20, 15))
            df.plot.scatter(x=col1, y=col2, ax=ax)
            ax.set_xlabel(col1)
            ax.set_ylabel(col2)
            ax.set_title(f'{col1} vs {col2}')
            ax.legend()
            plt.show()
```

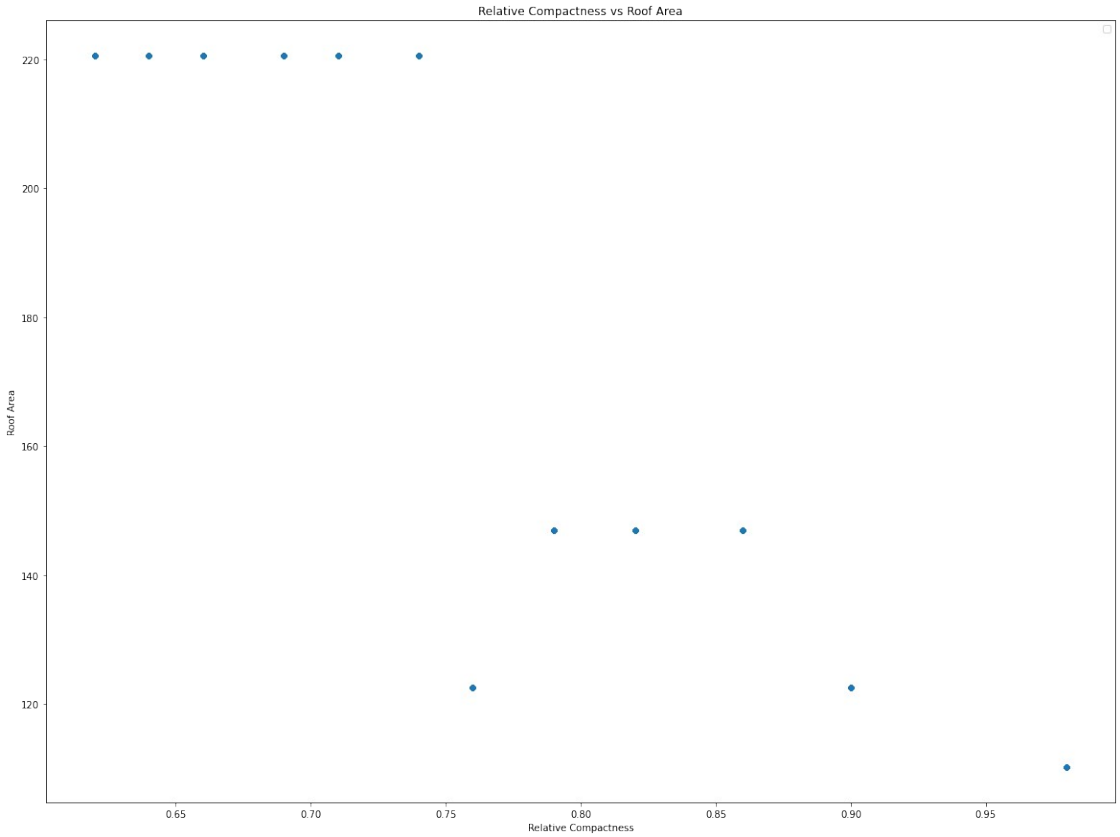
```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/
core.py:1114: UserWarning: No data for colormapping provided via 'c'.
Parameters 'cmap' will be ignored
  scatter = ax.scatter(
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



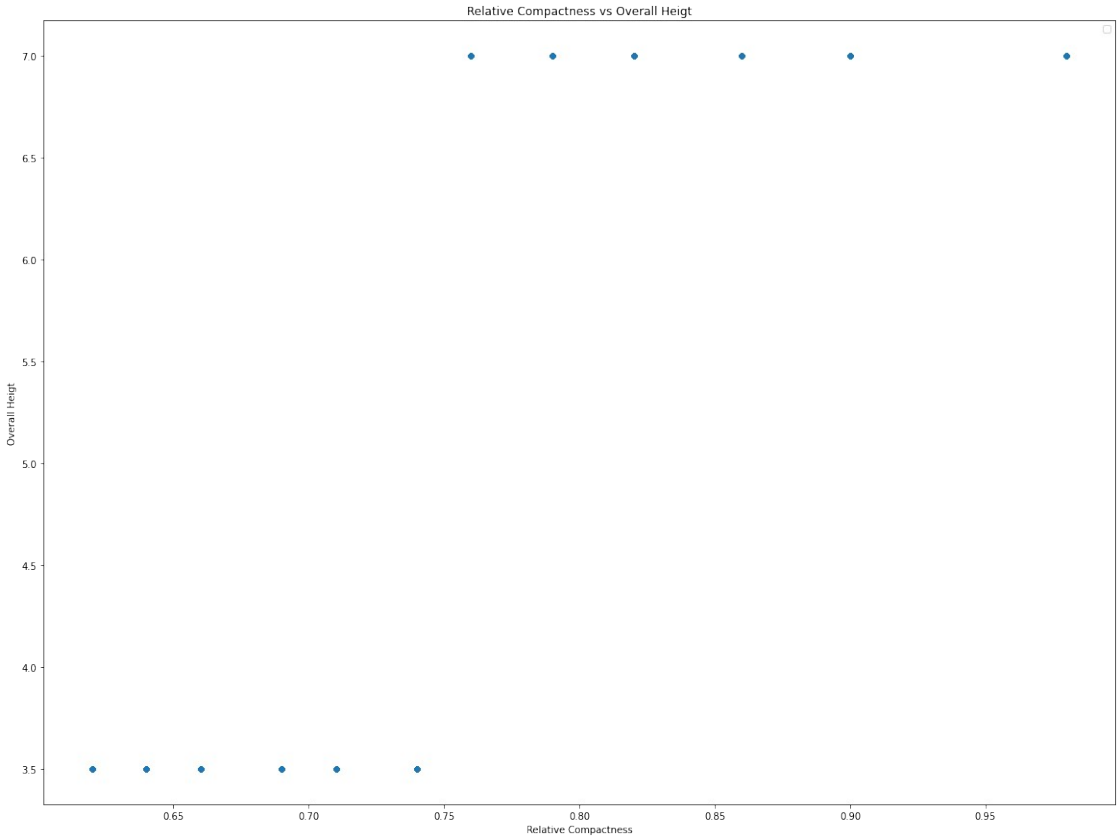

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



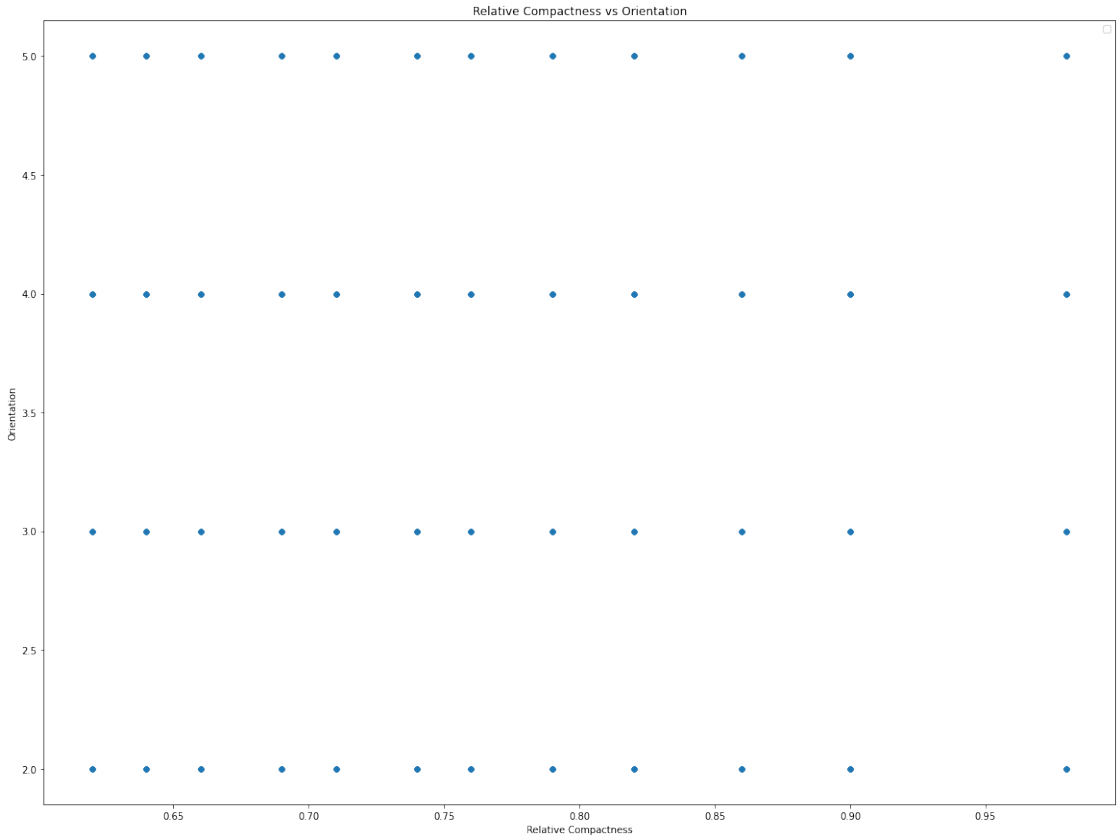

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



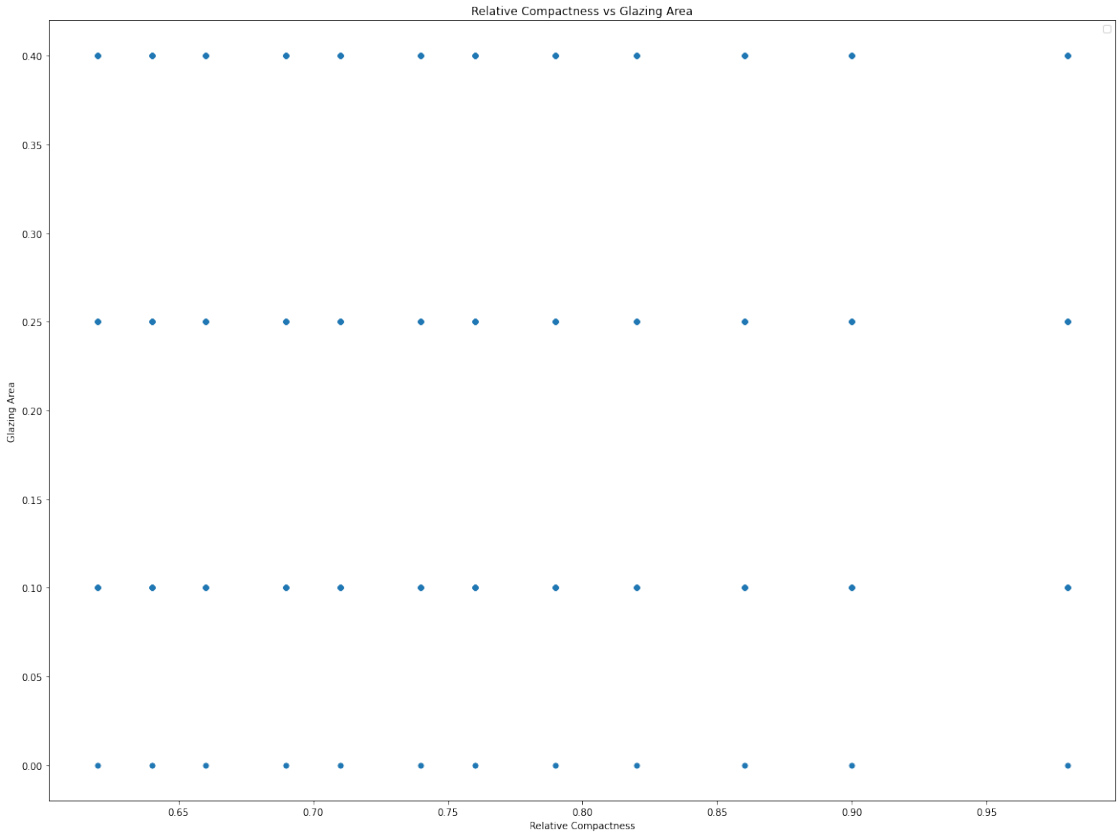

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



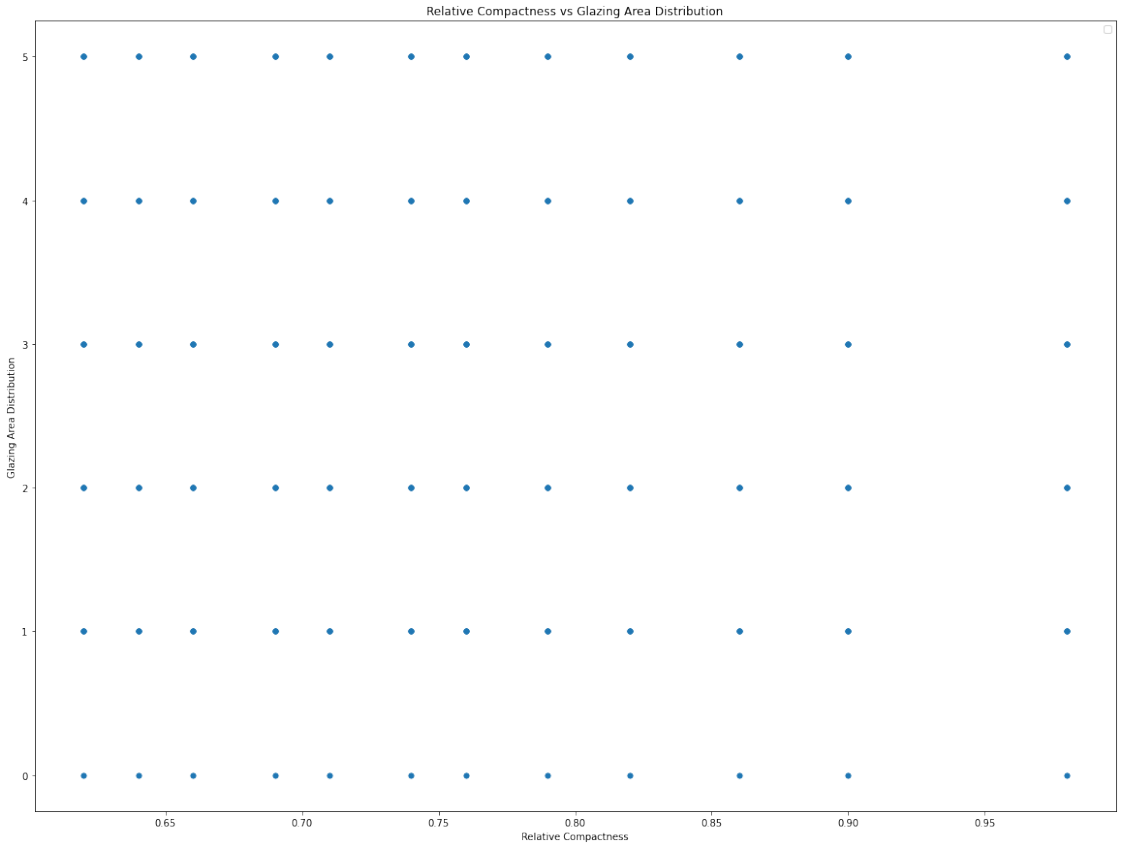

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



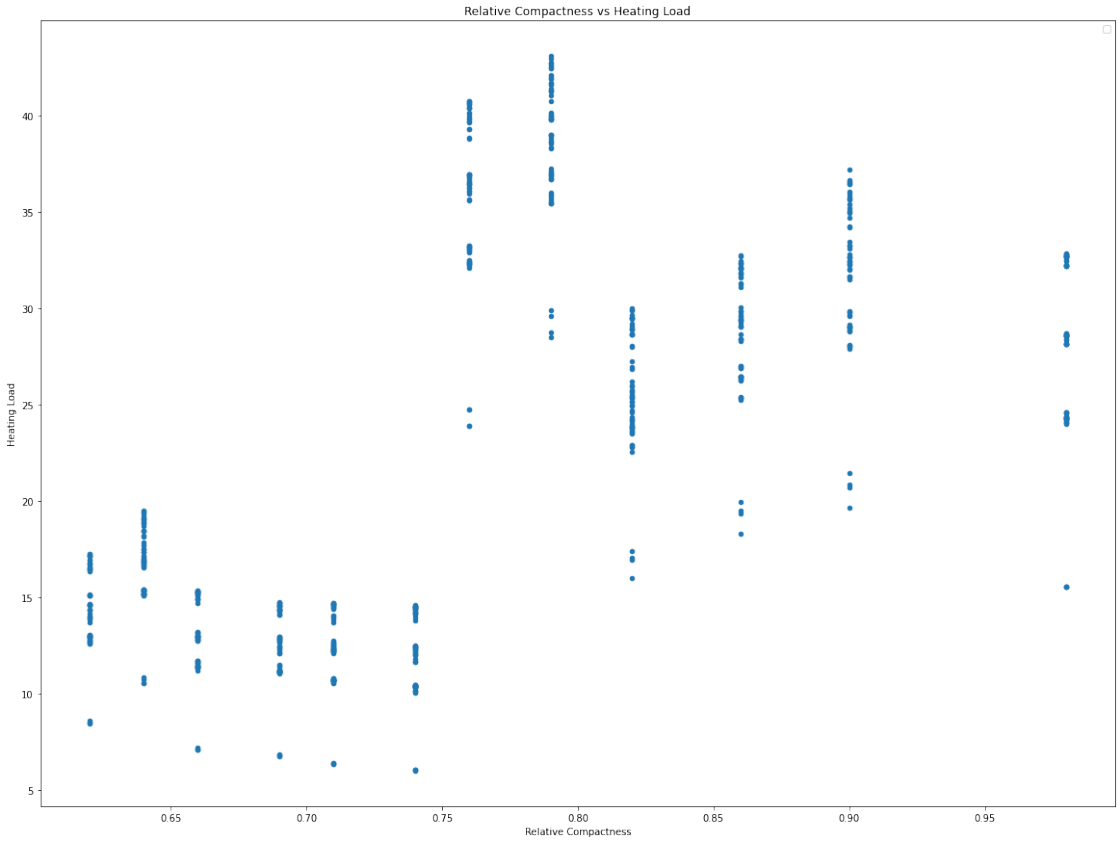

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



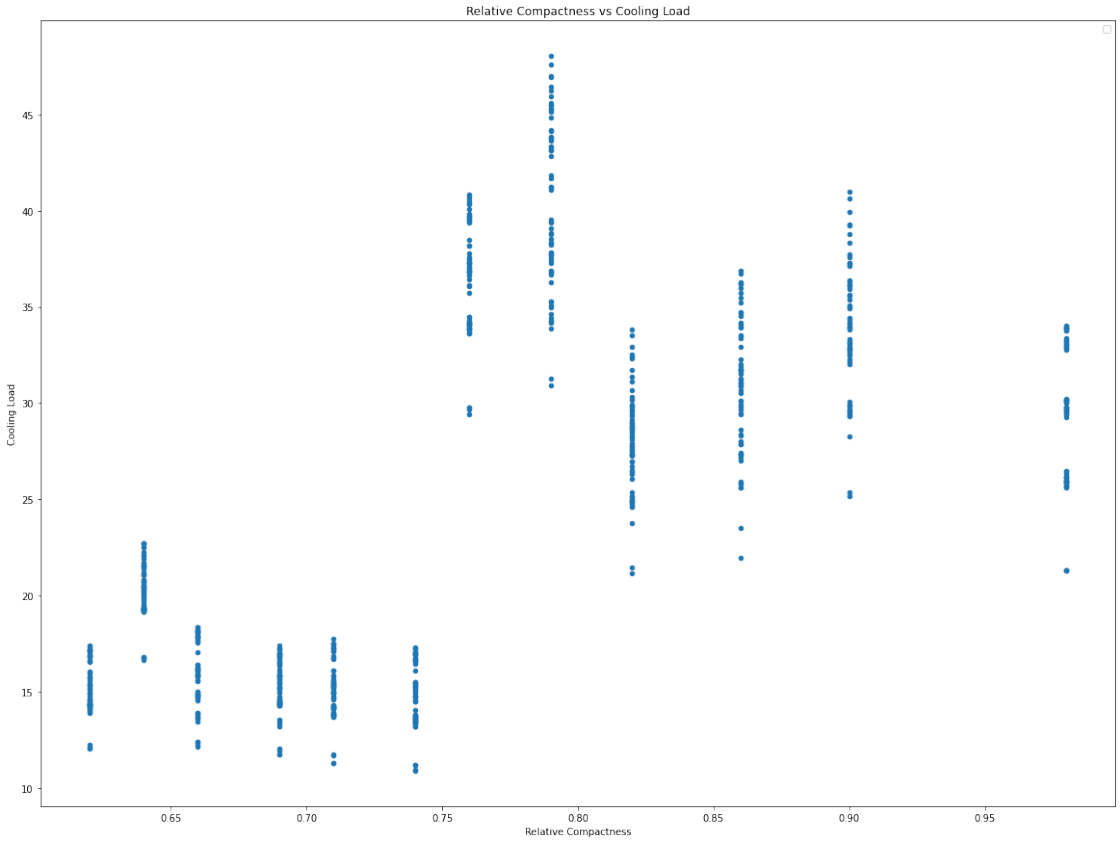

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



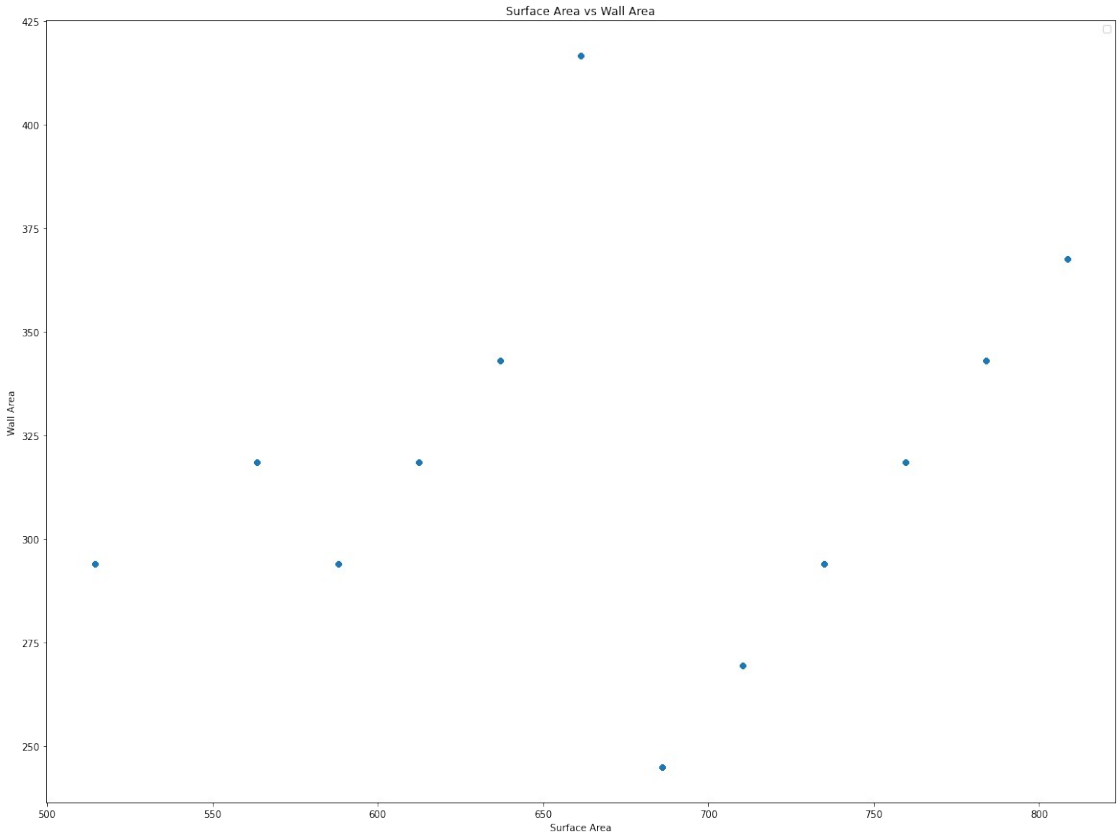

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



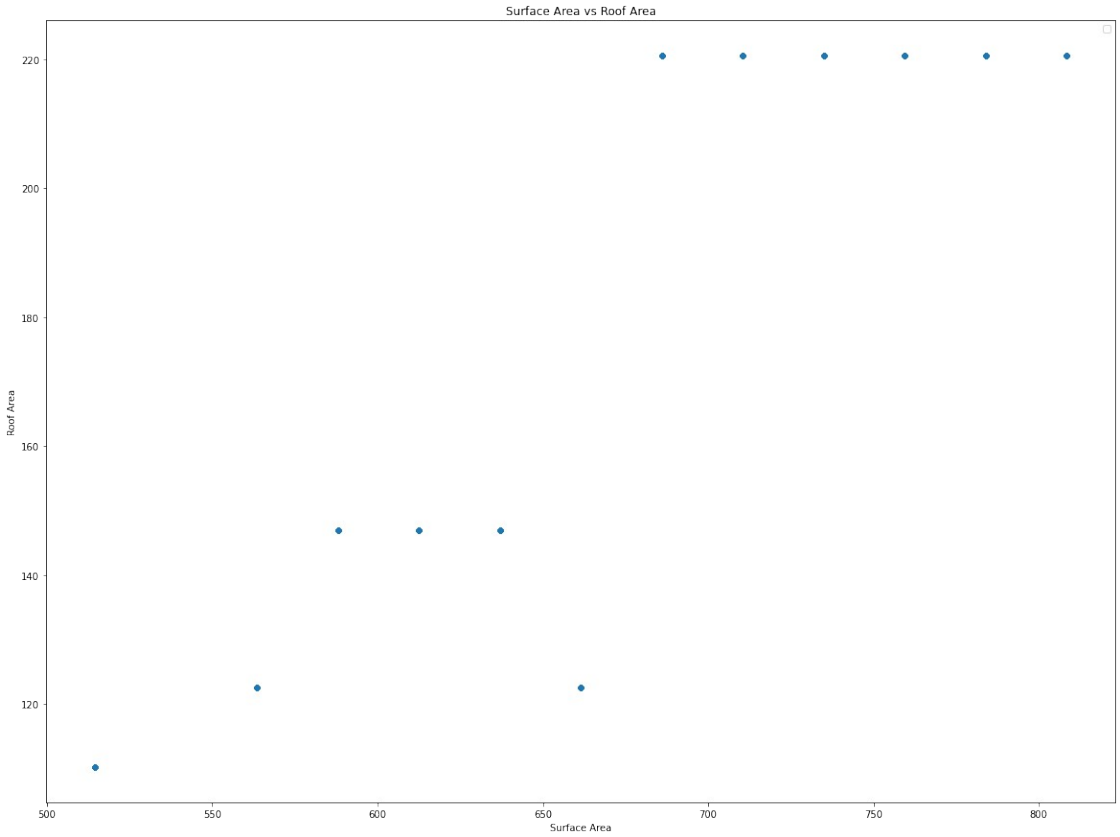

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



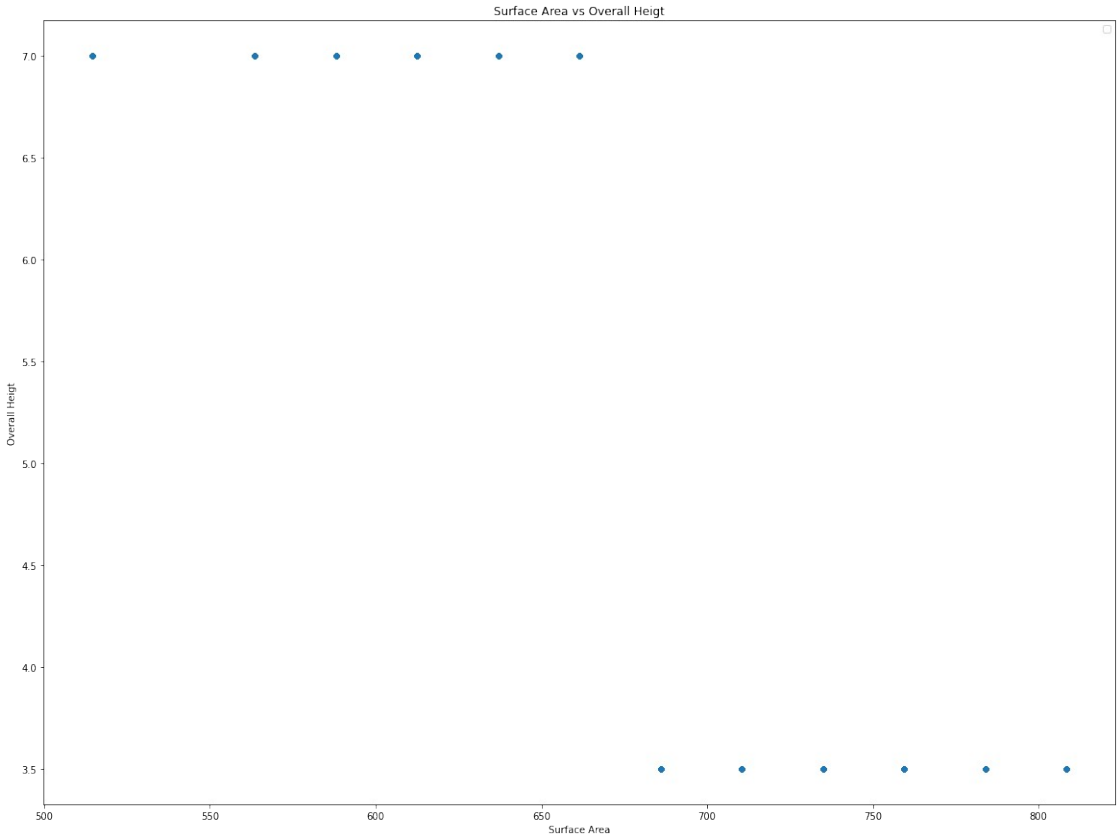

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



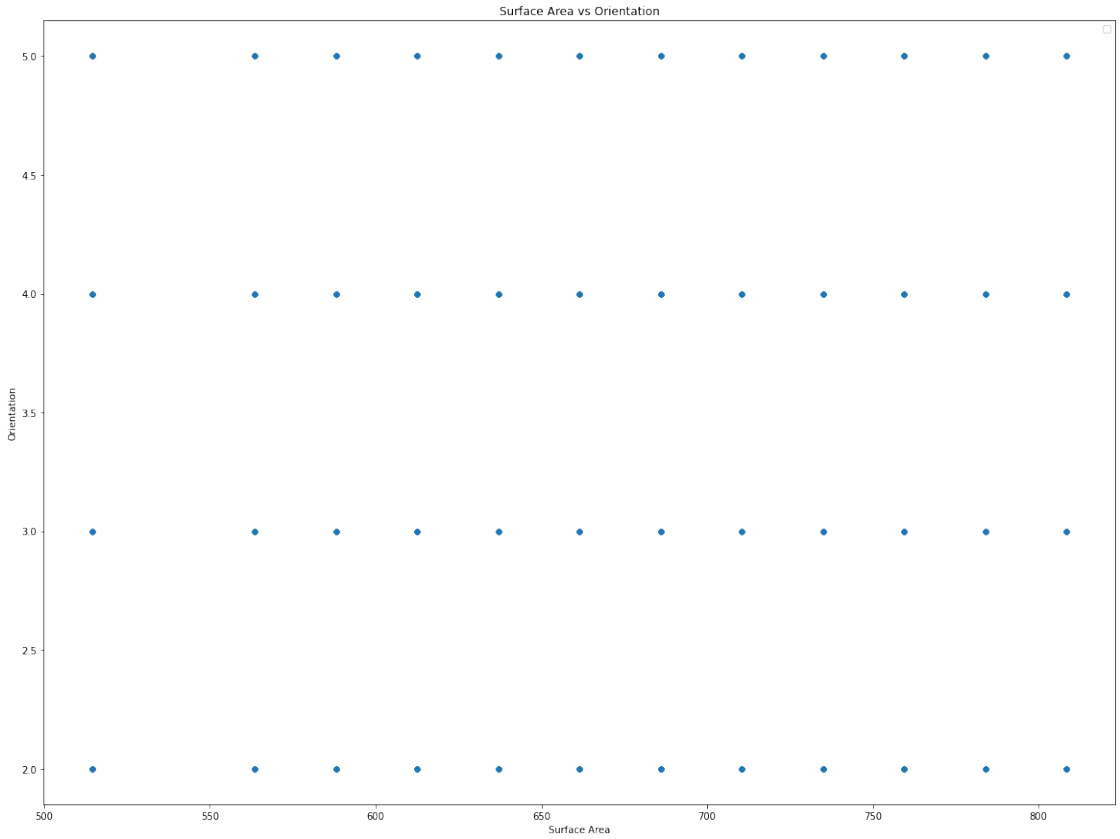

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



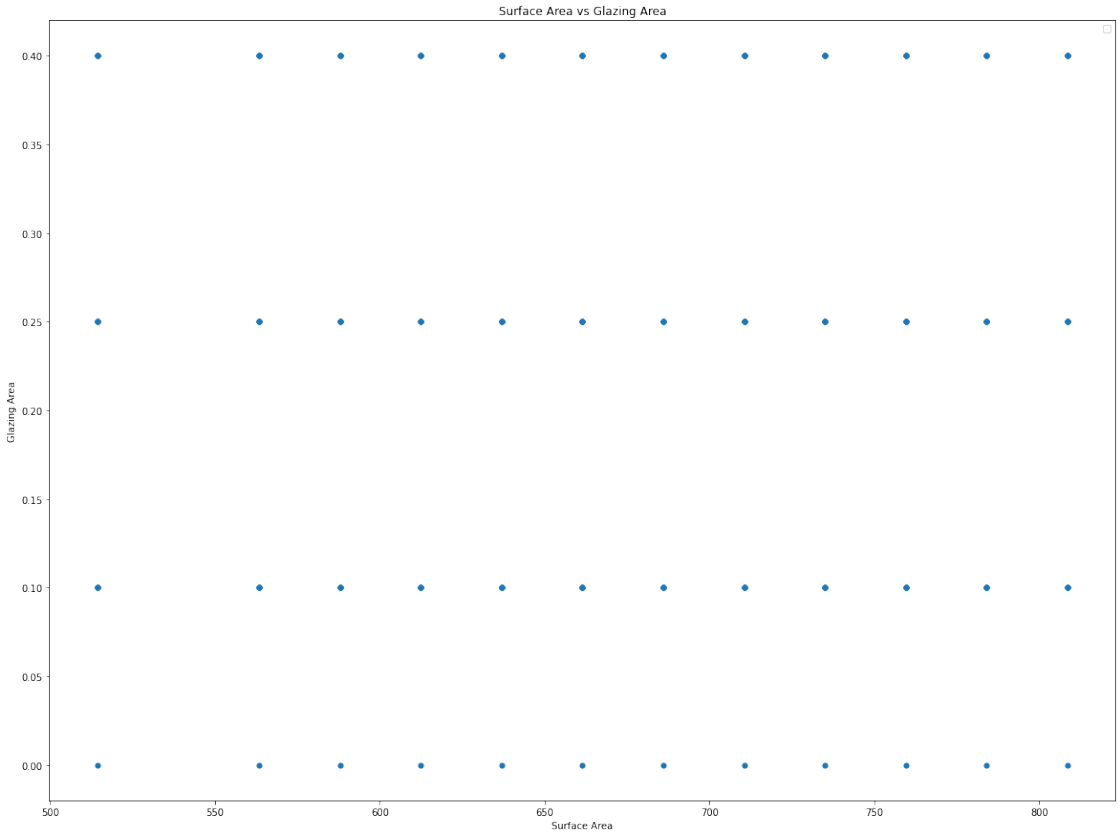

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



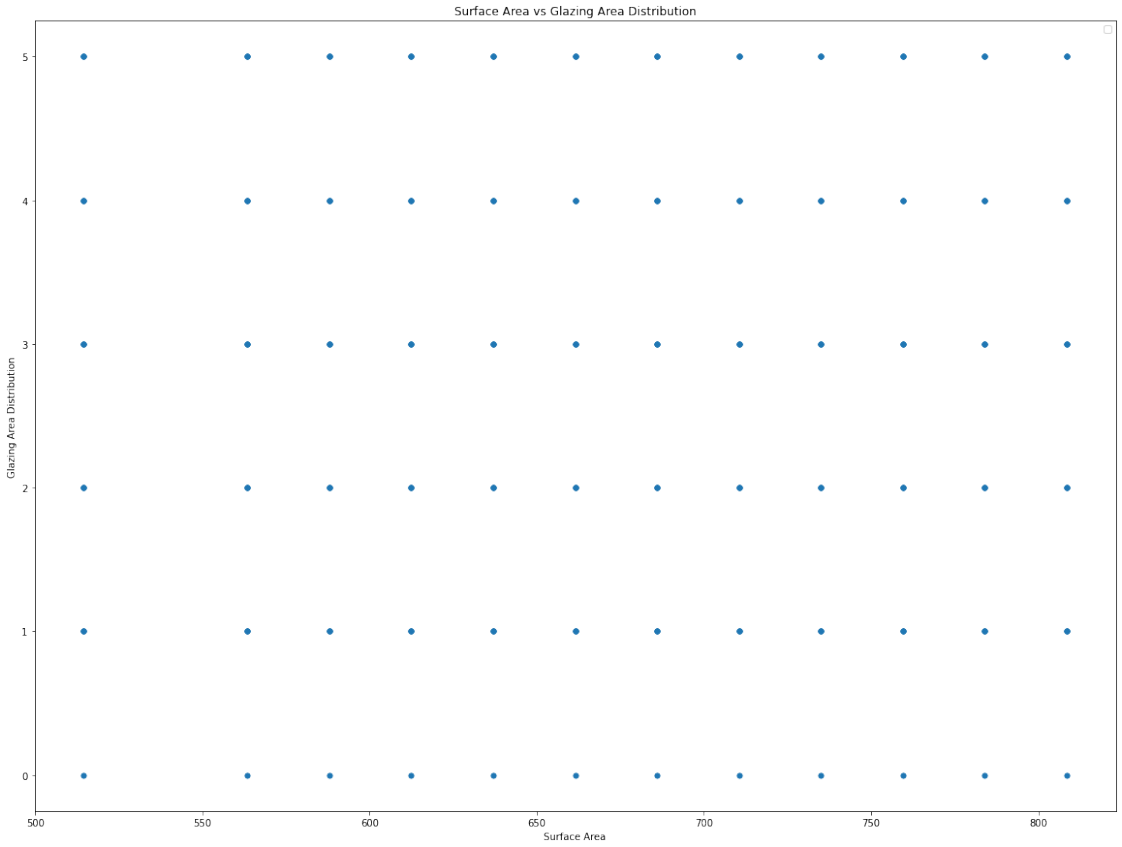

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



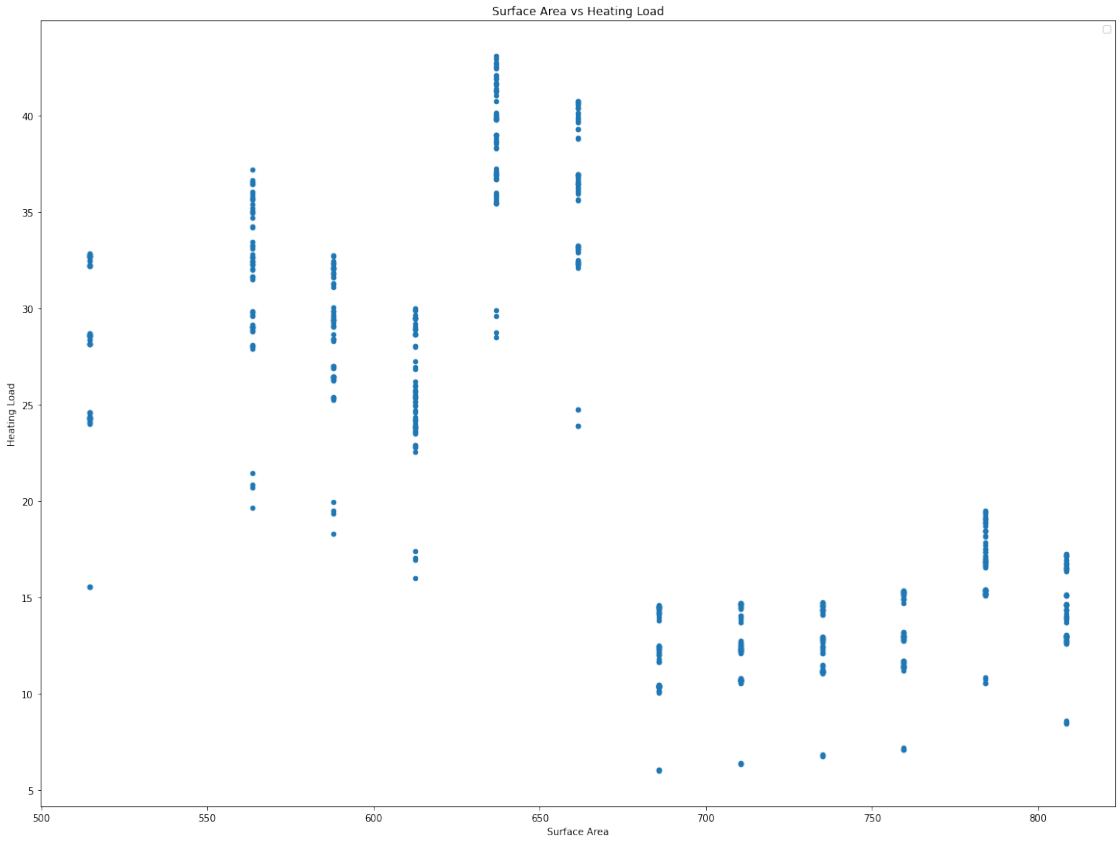

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



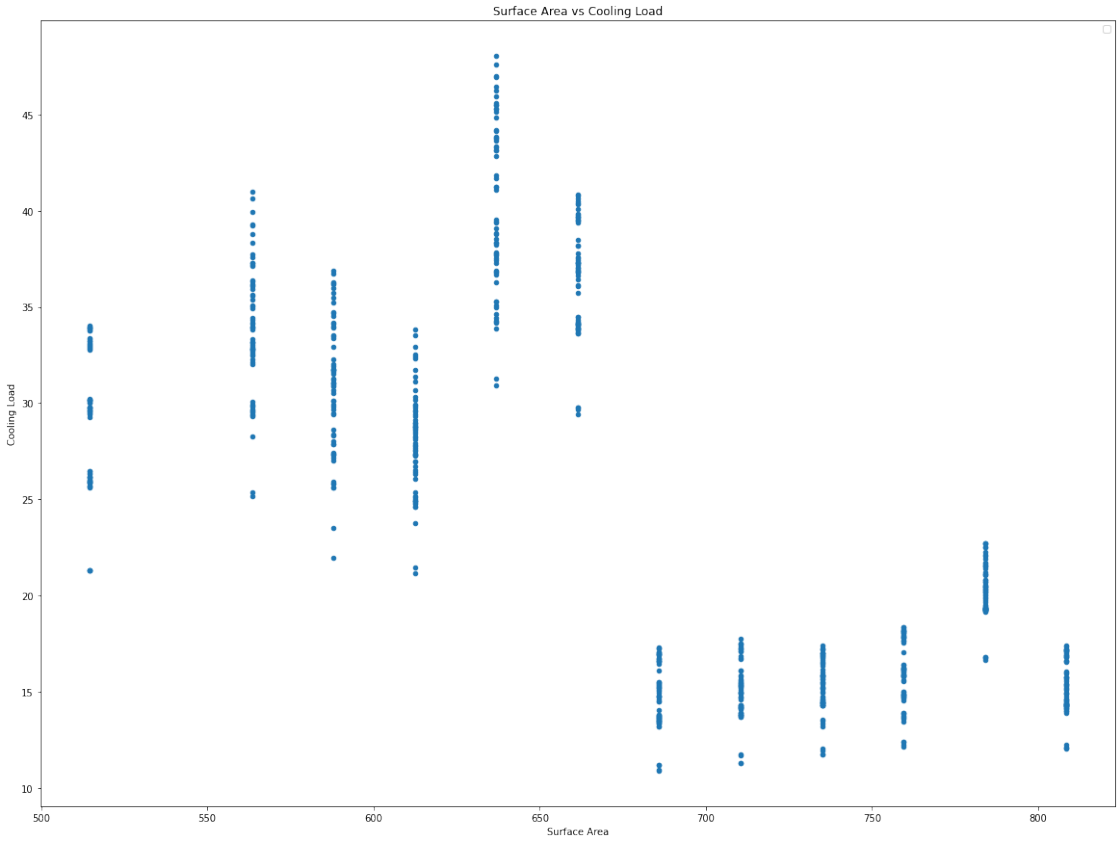

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



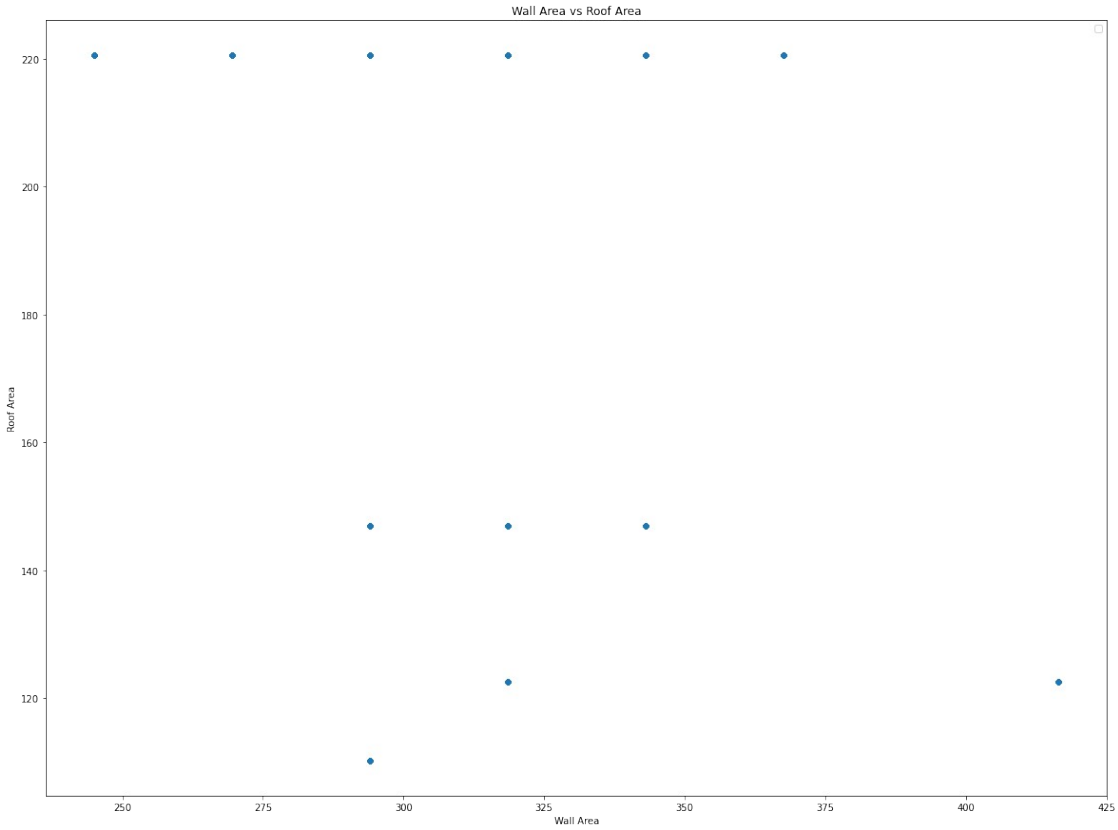

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



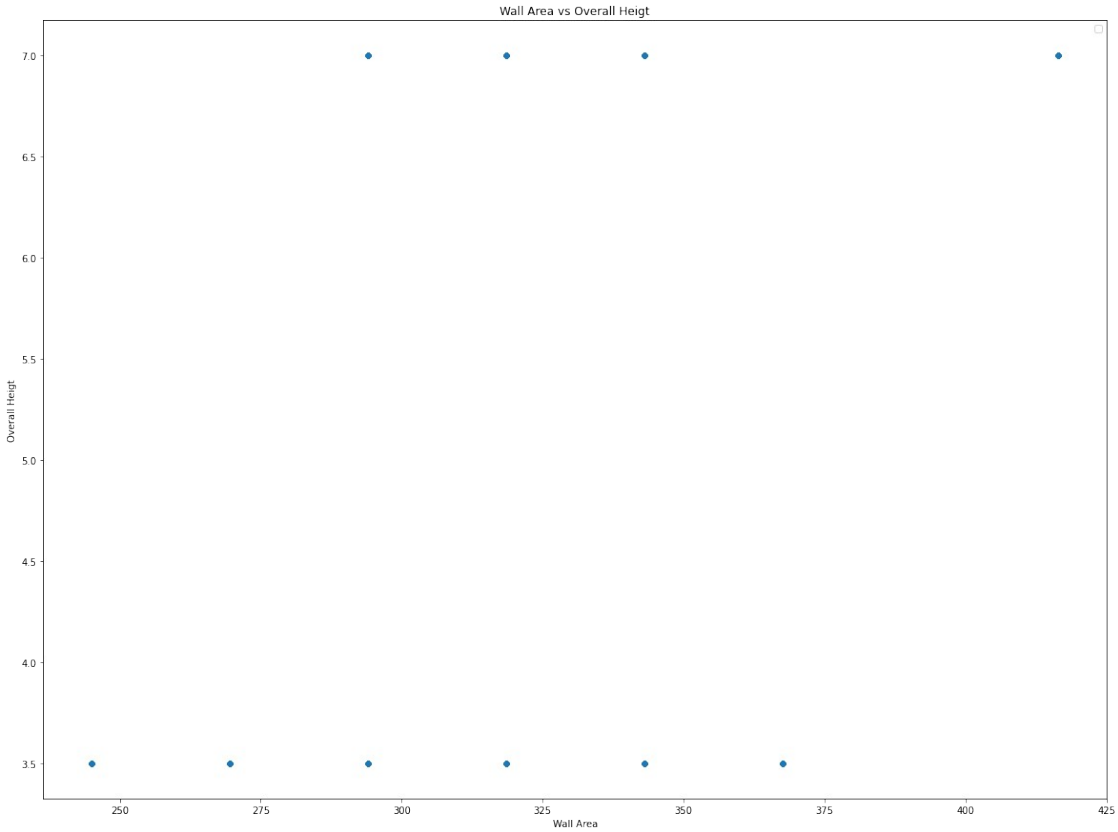

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



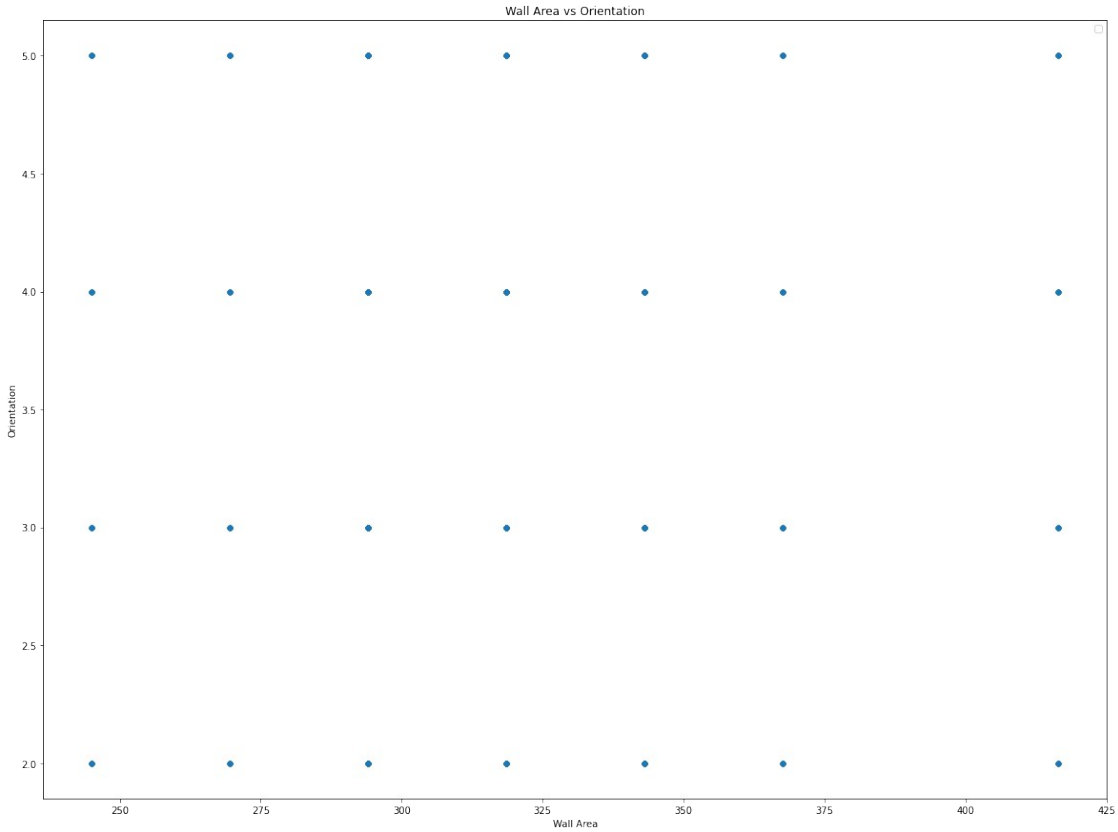

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



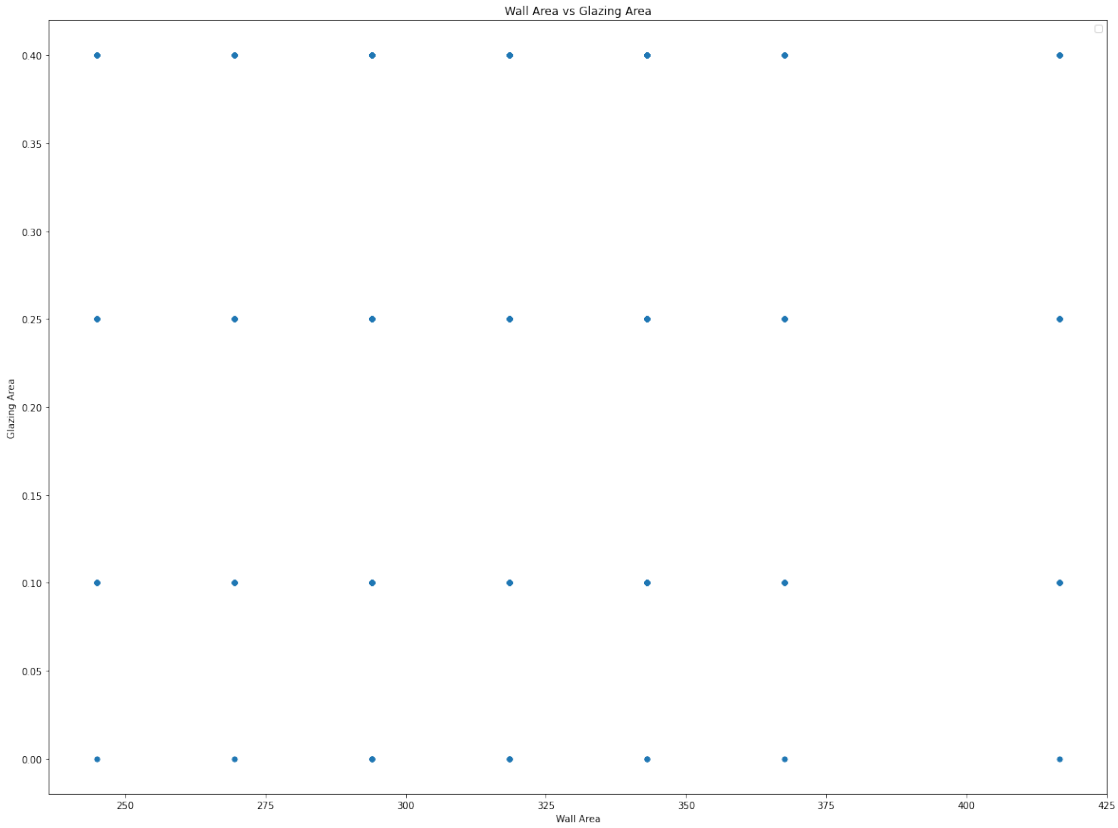

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



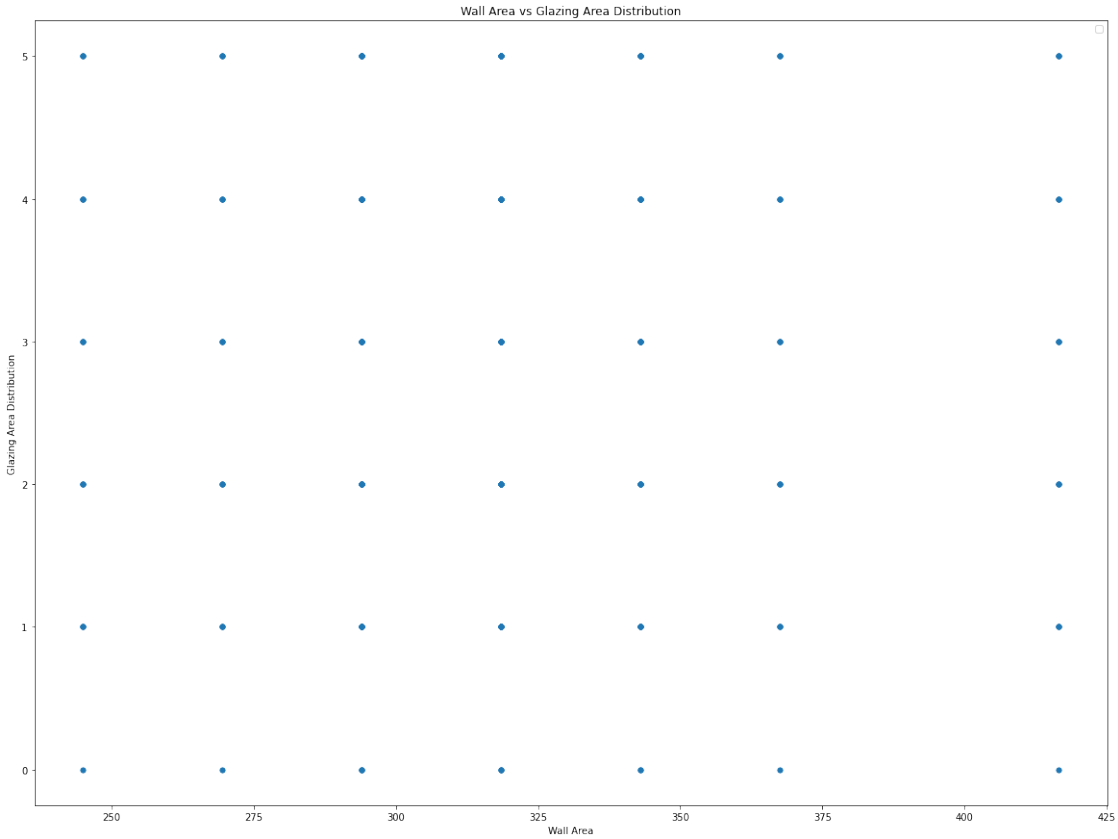

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



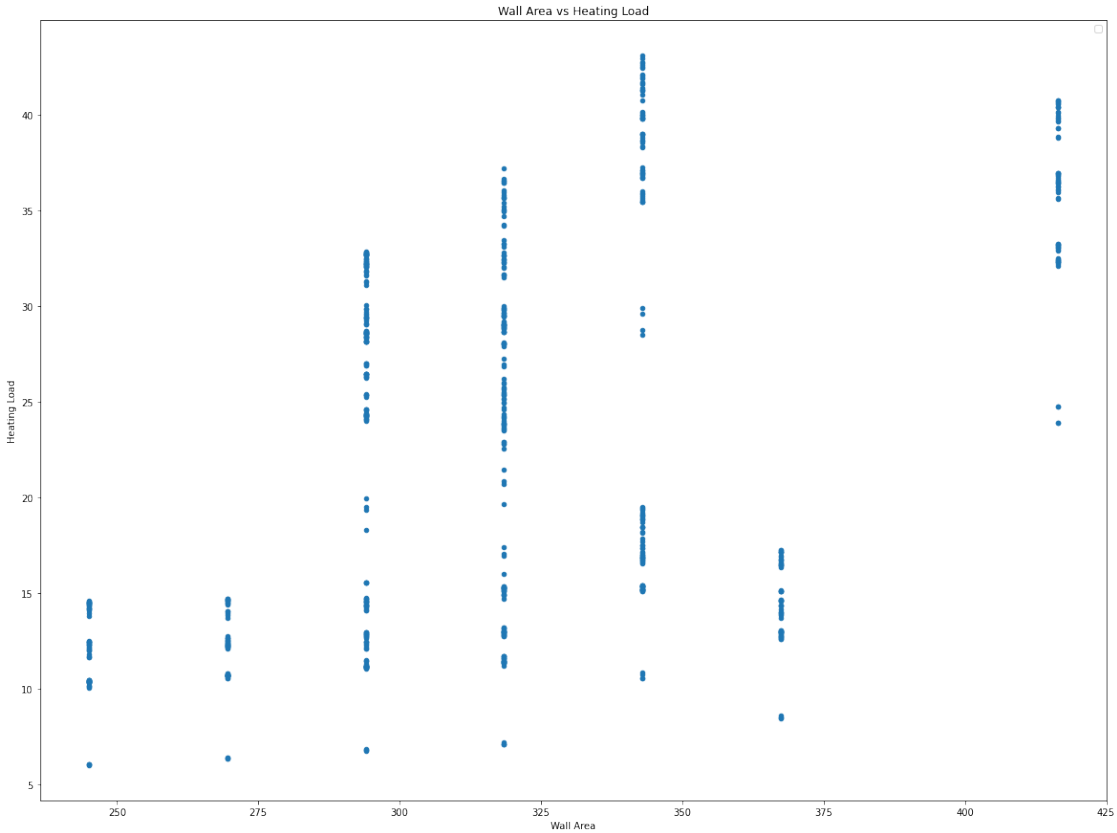

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



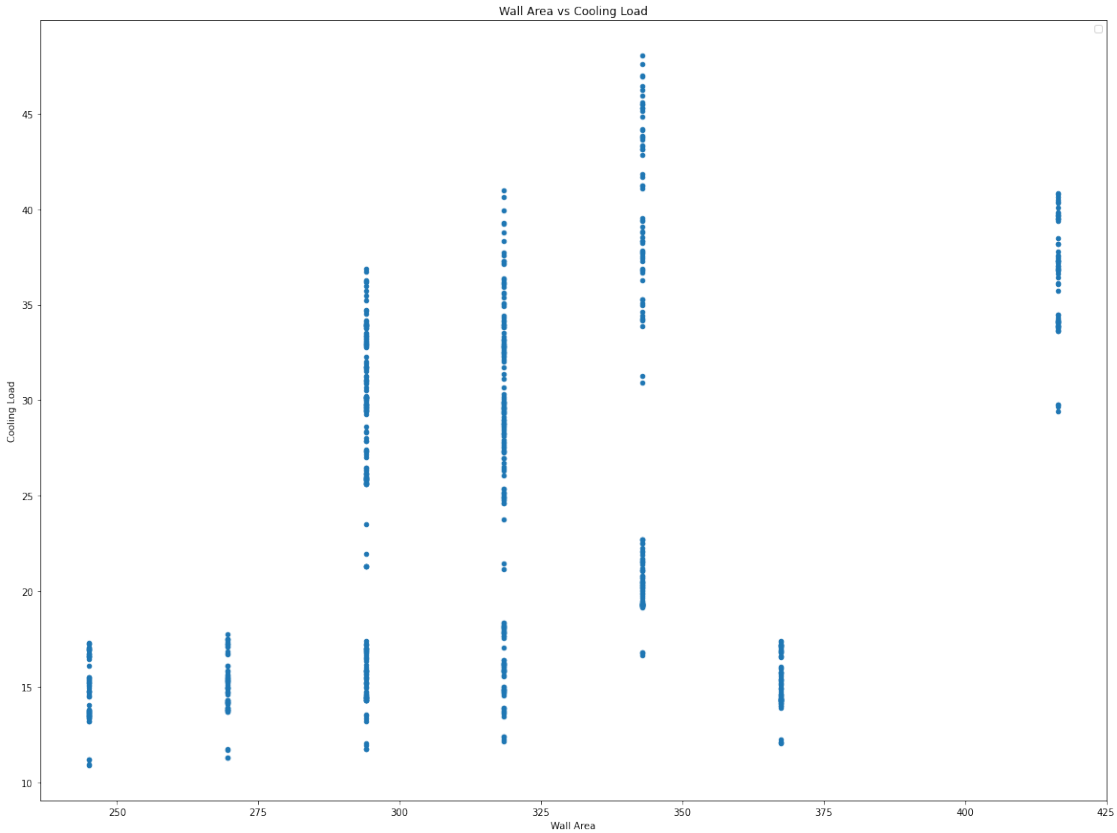

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



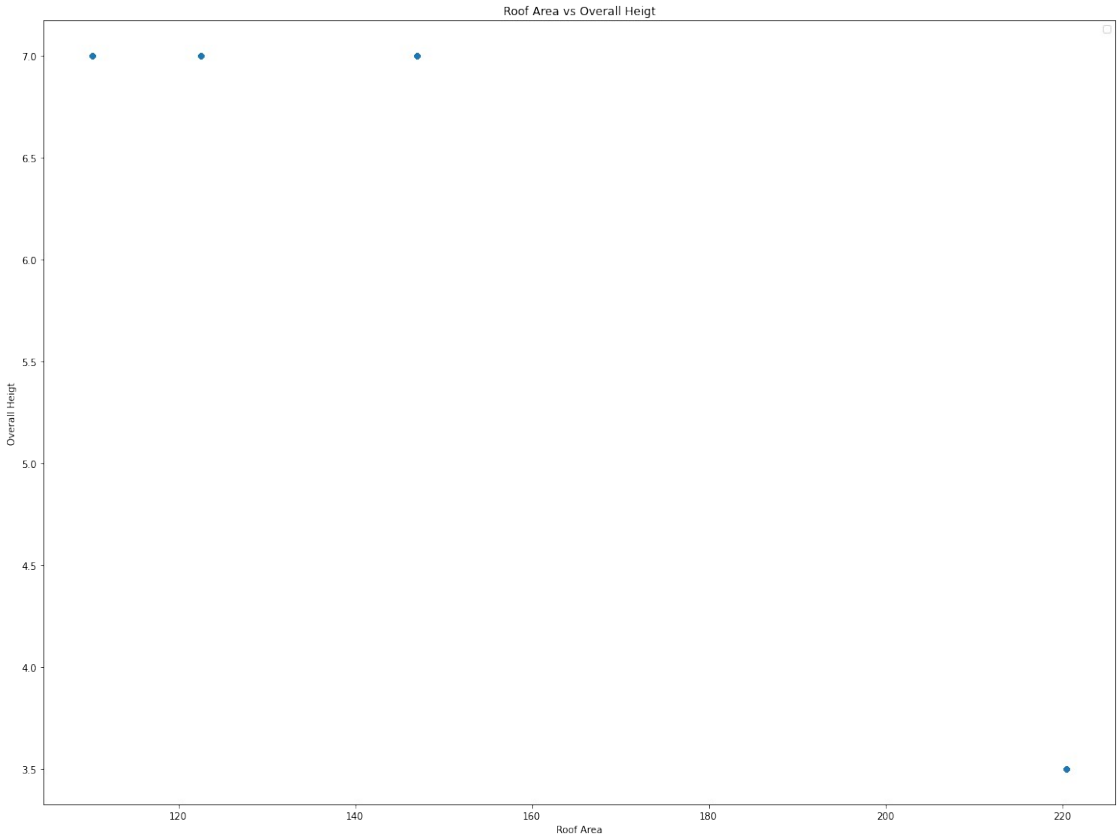

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



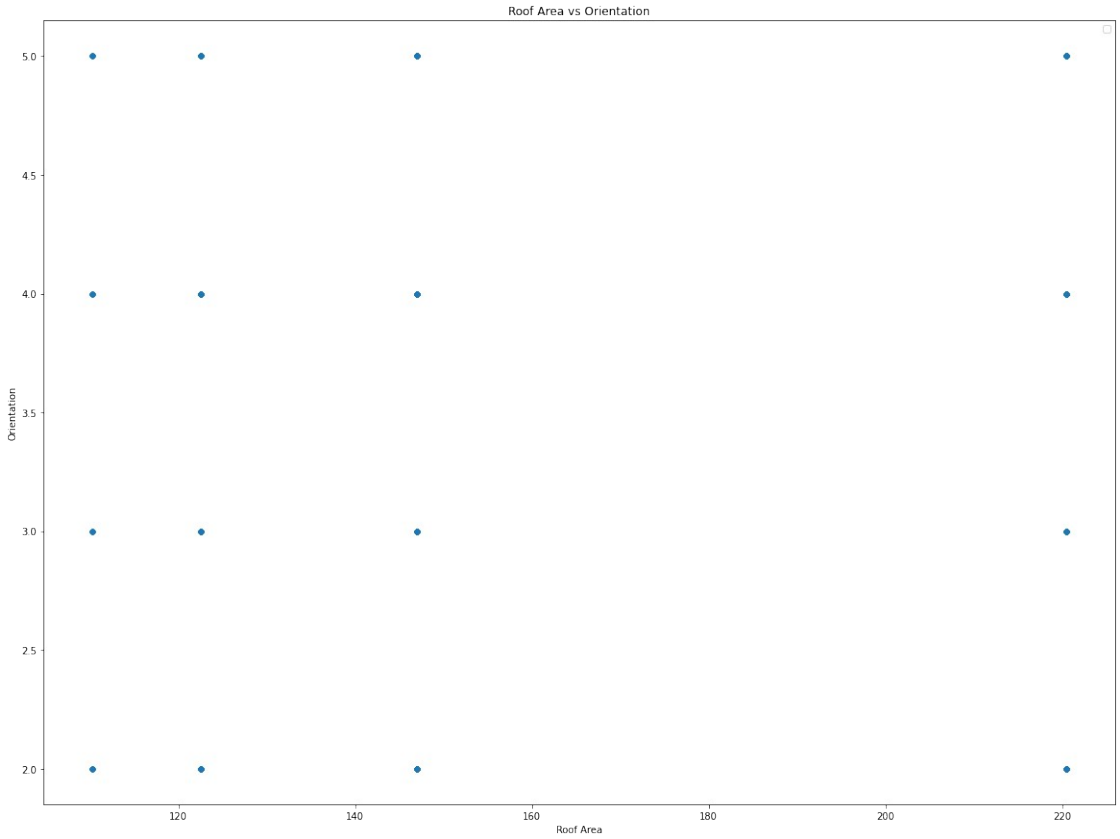

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



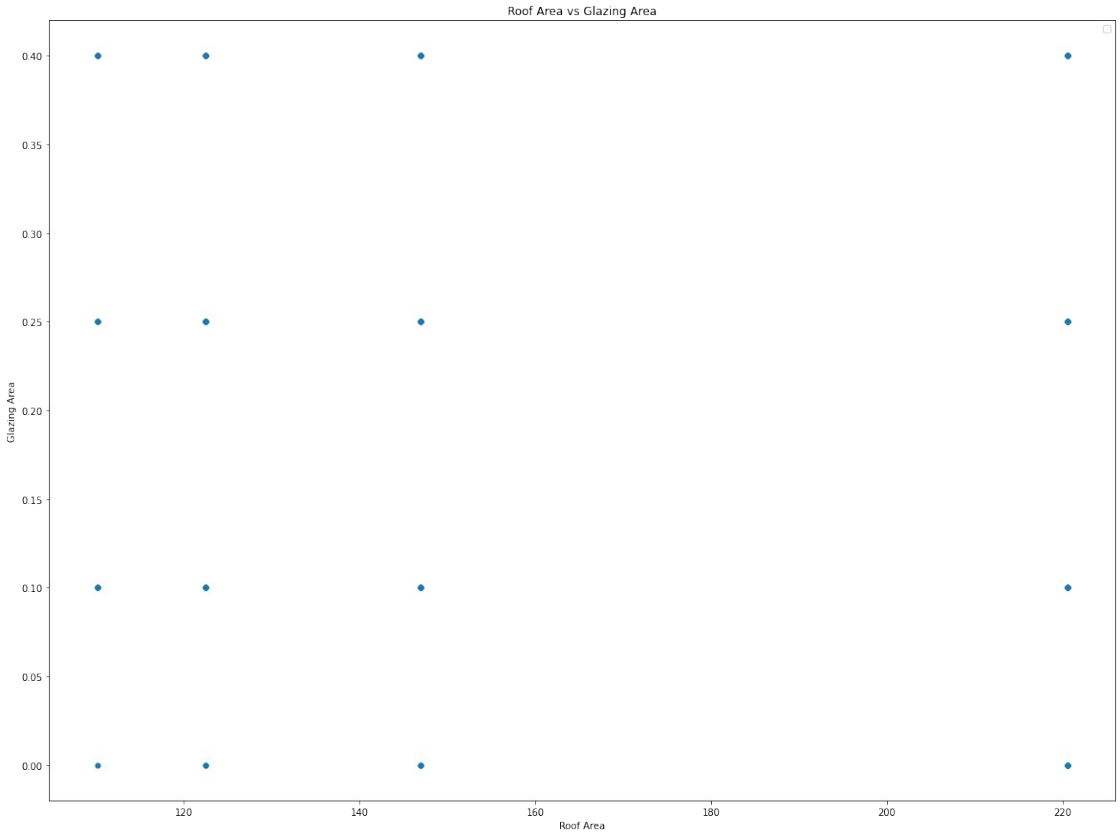

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



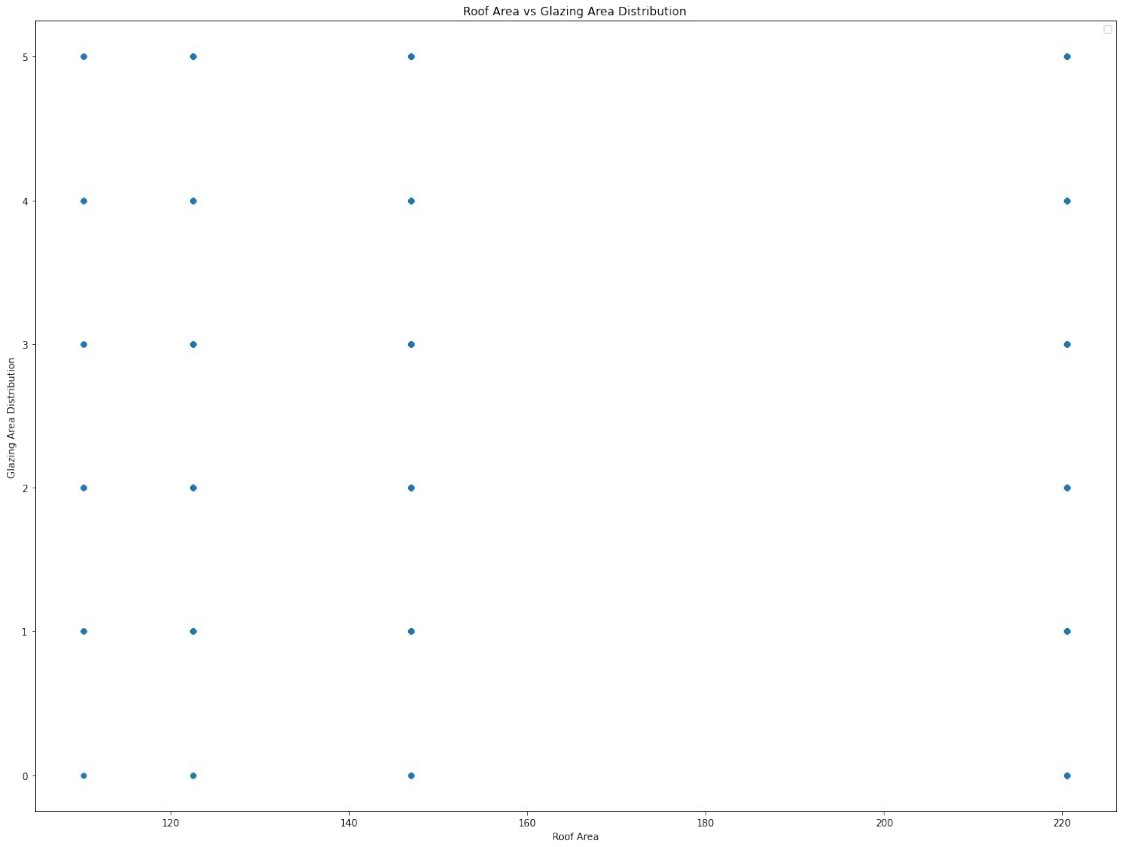

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



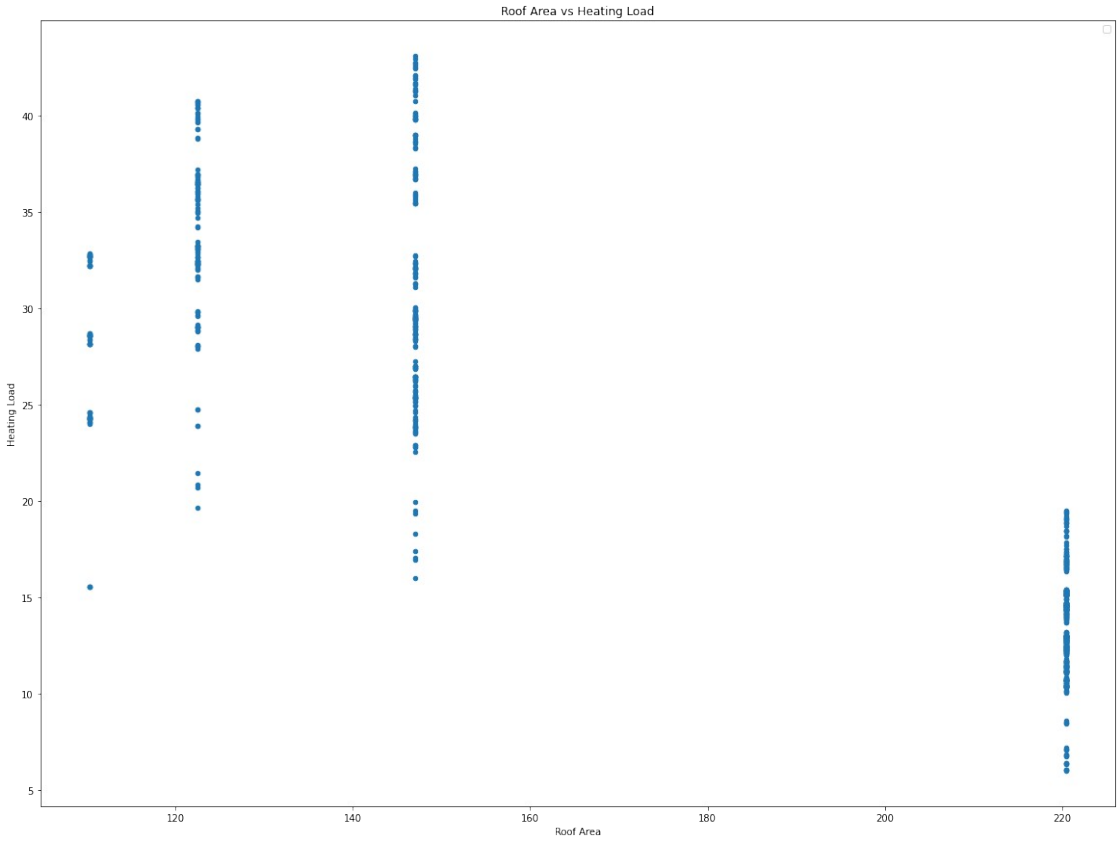

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



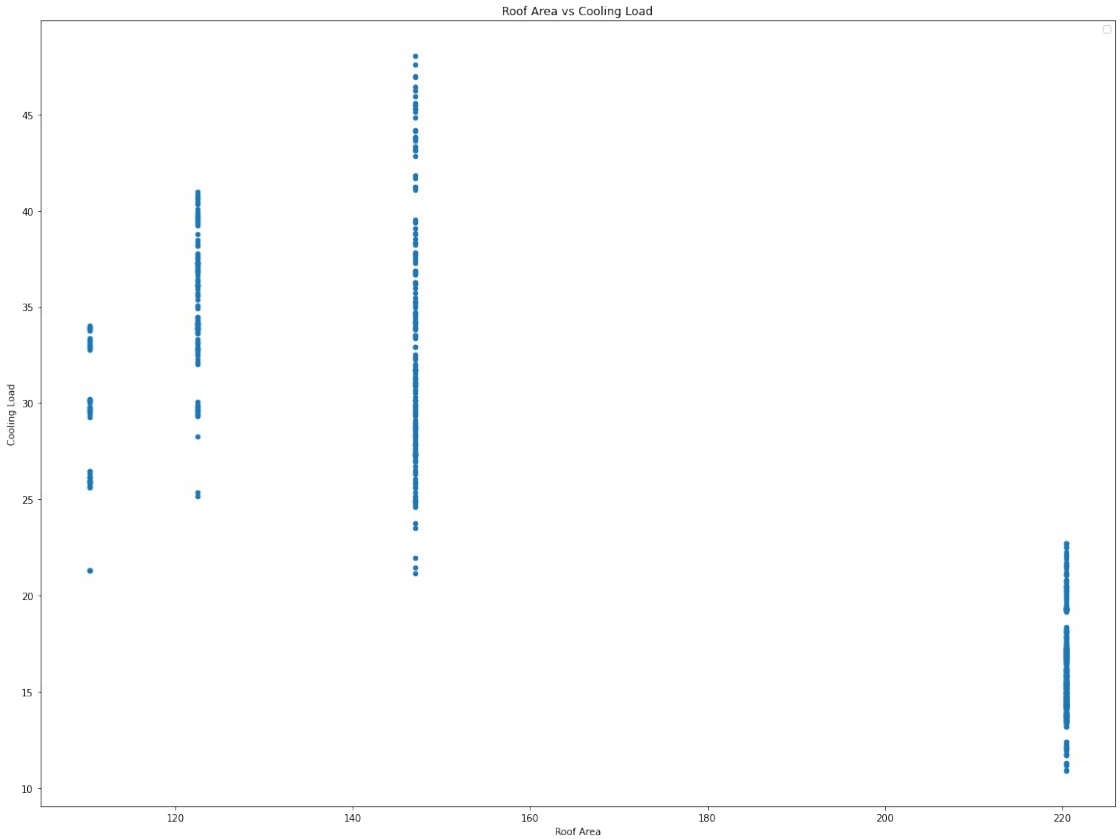

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



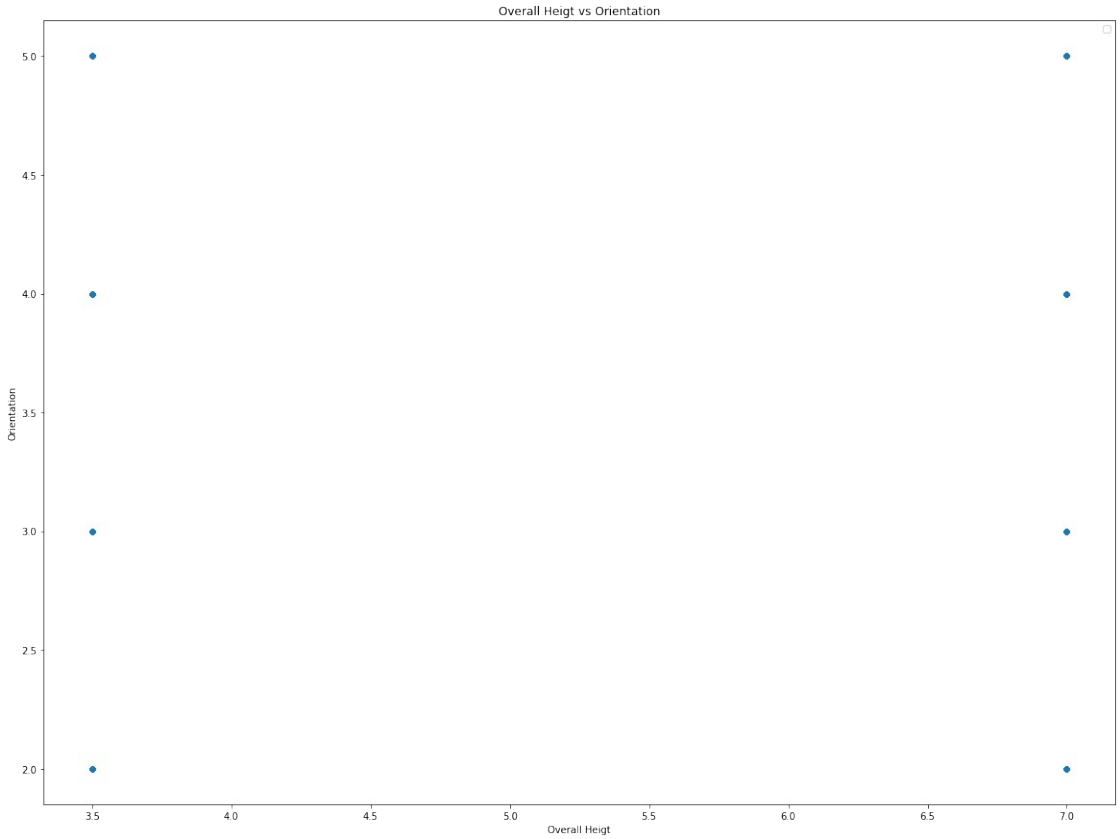

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



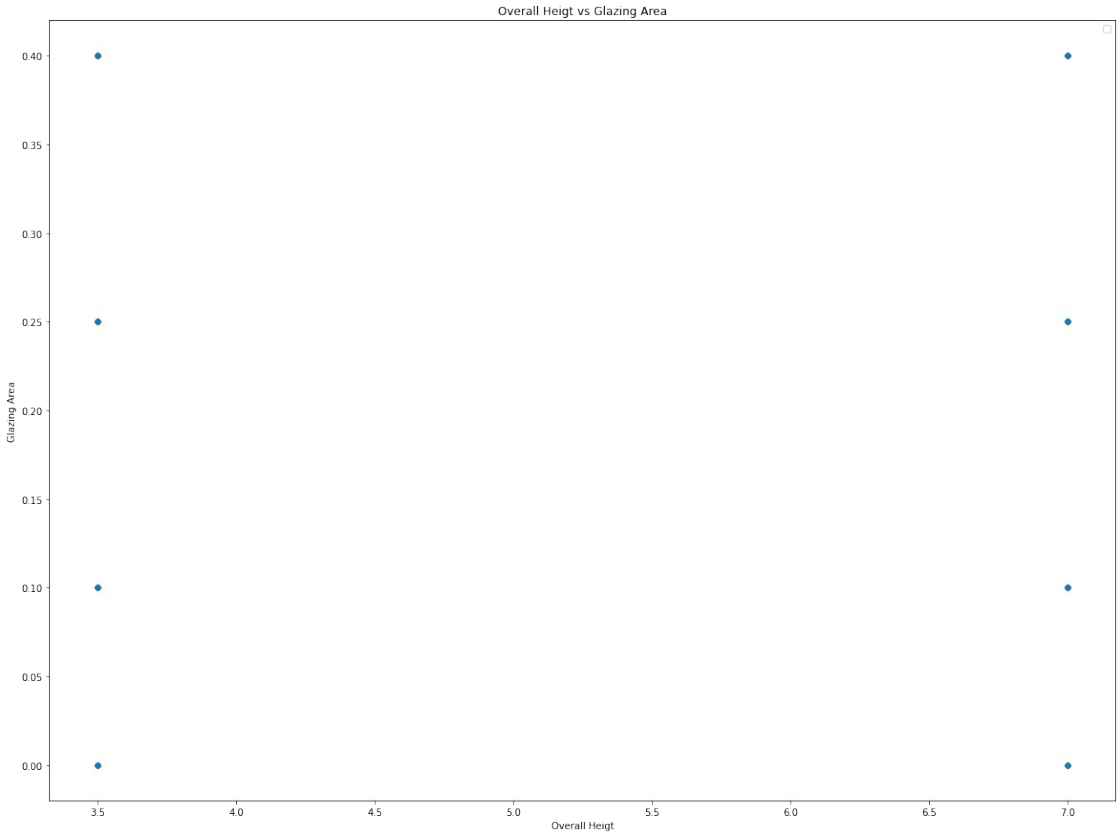

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



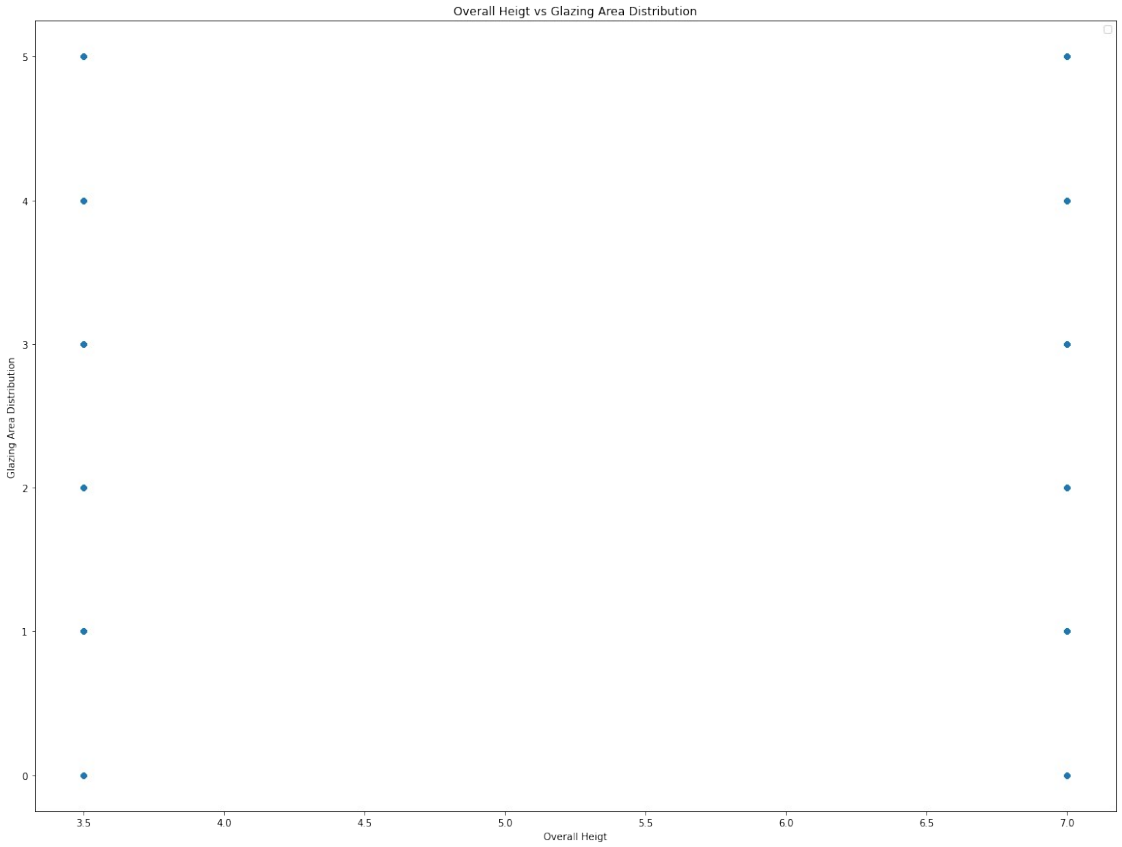

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



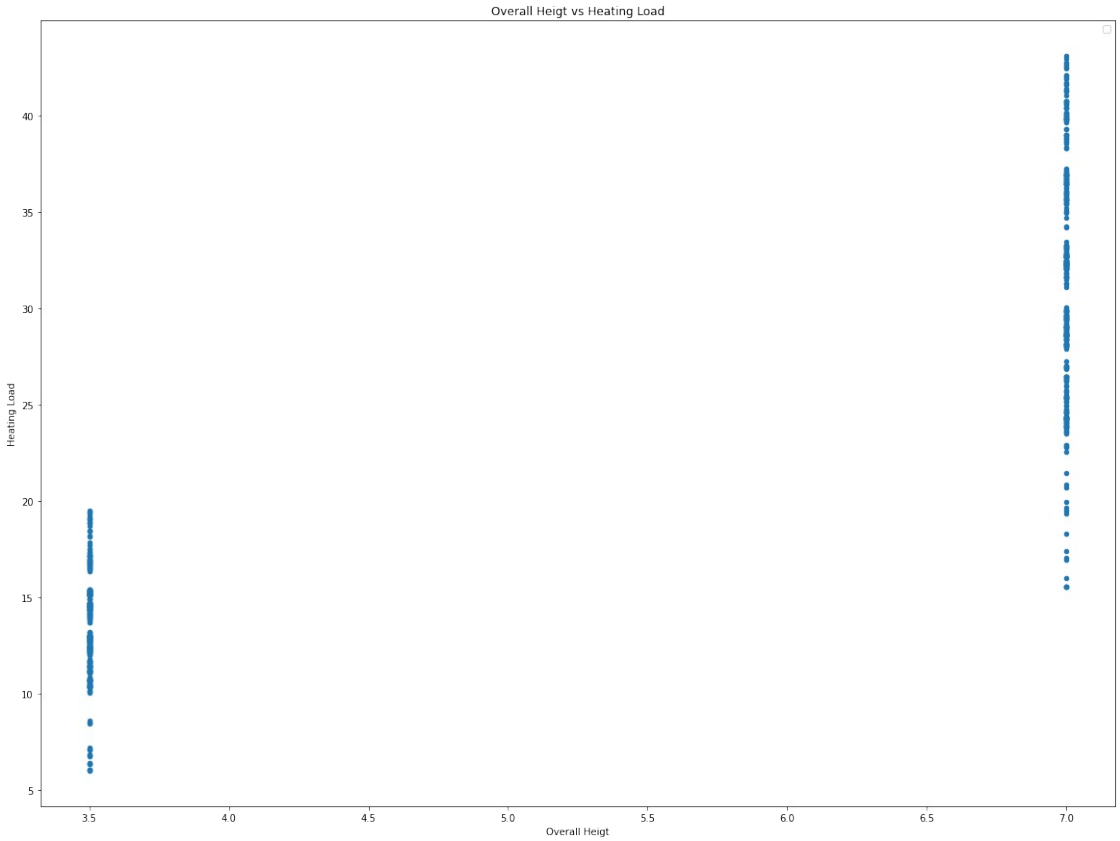

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



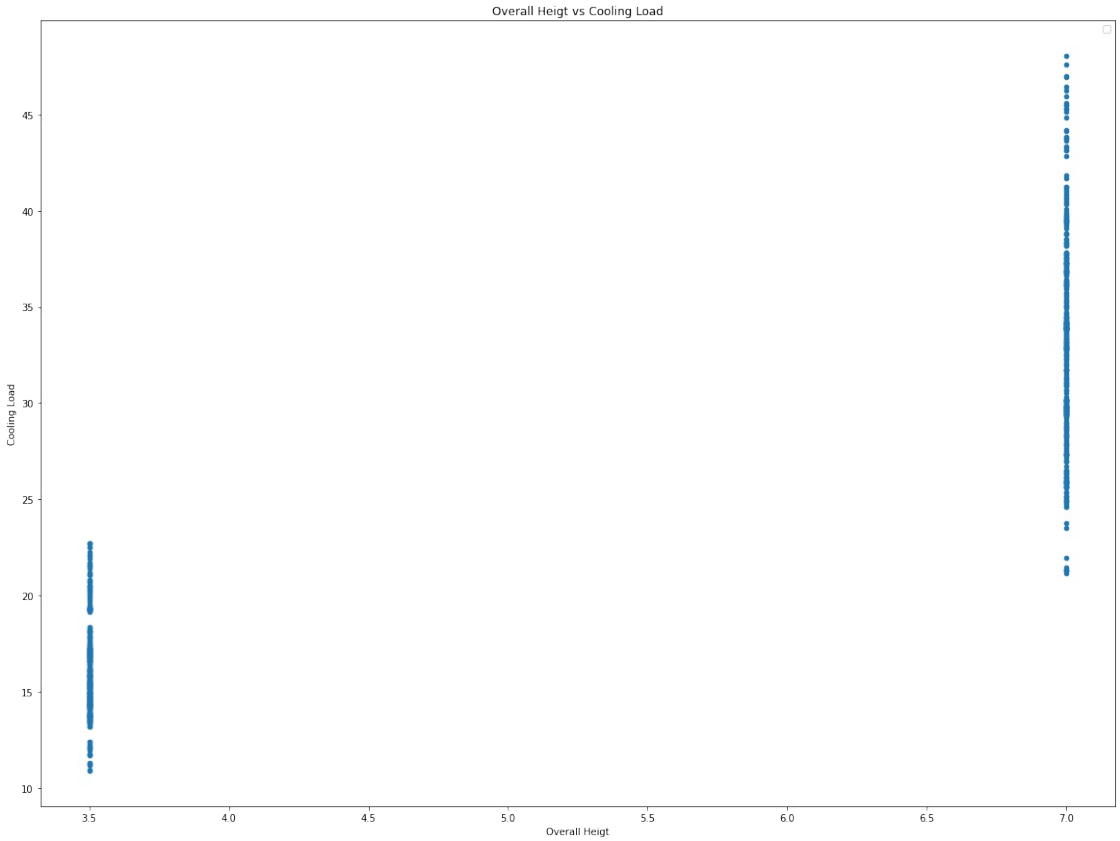

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



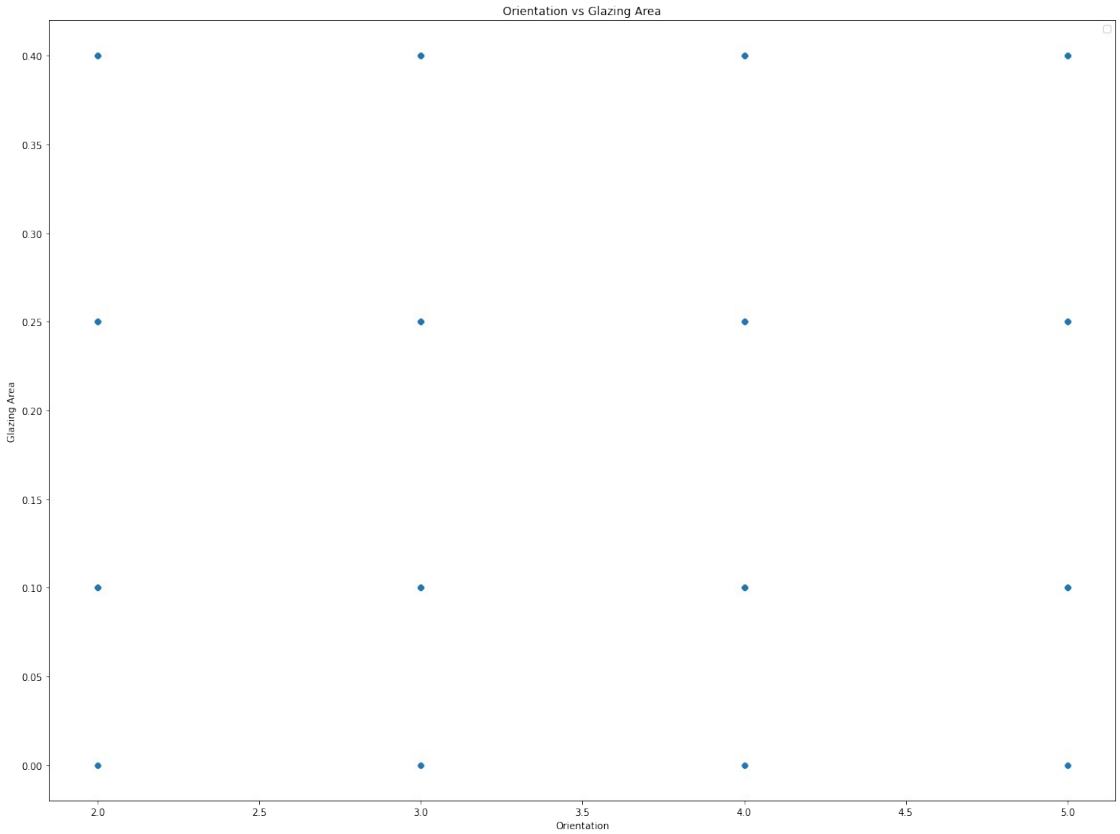

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



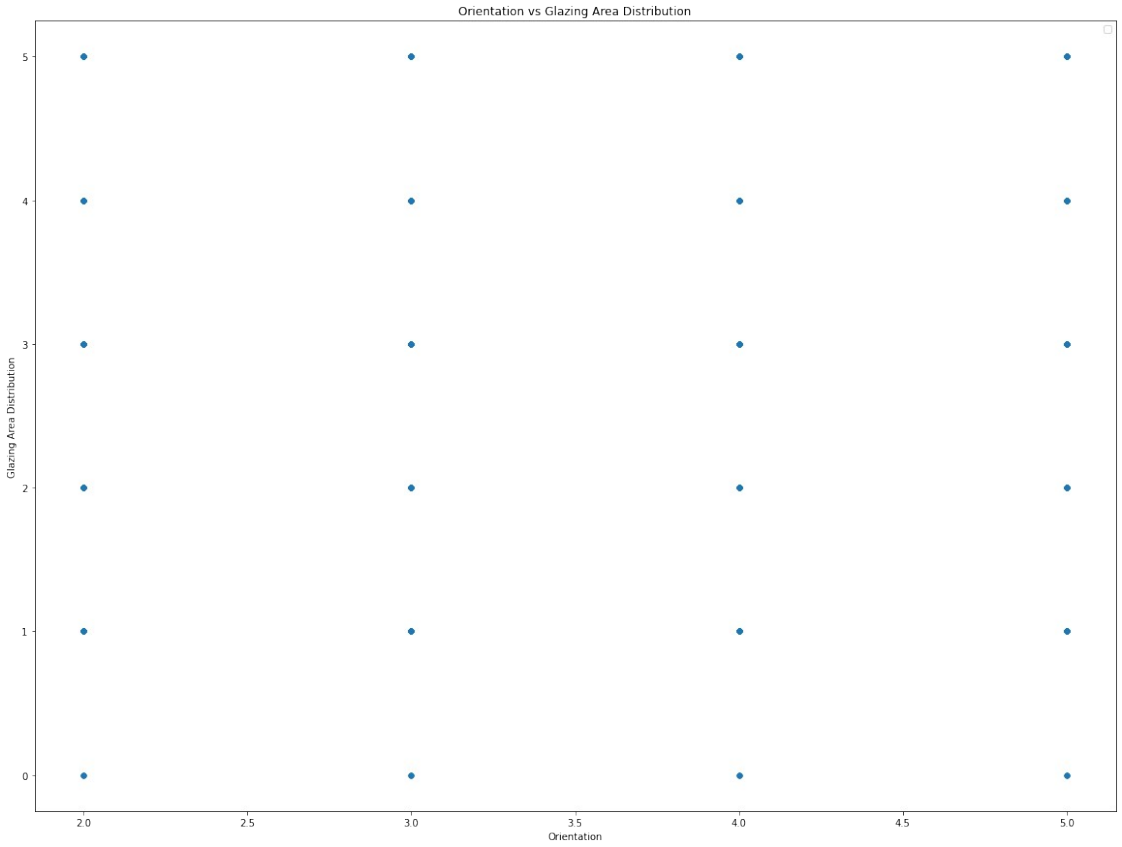

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



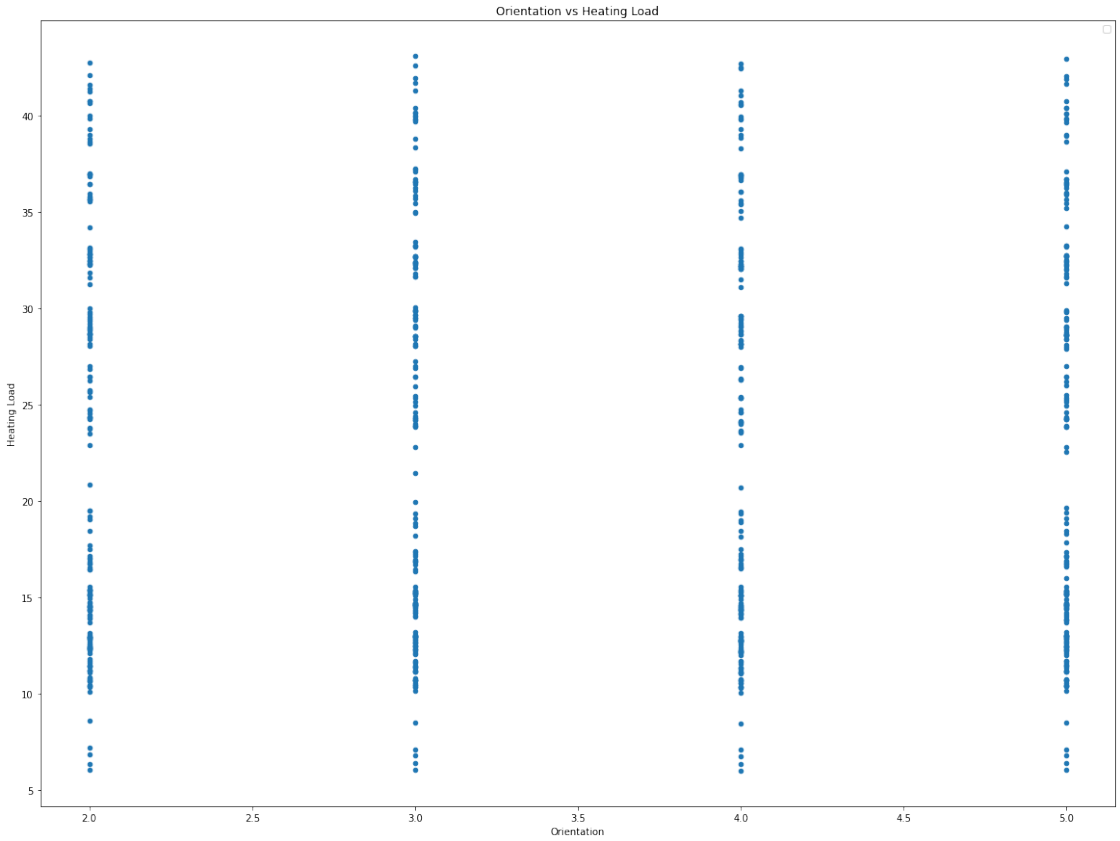

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



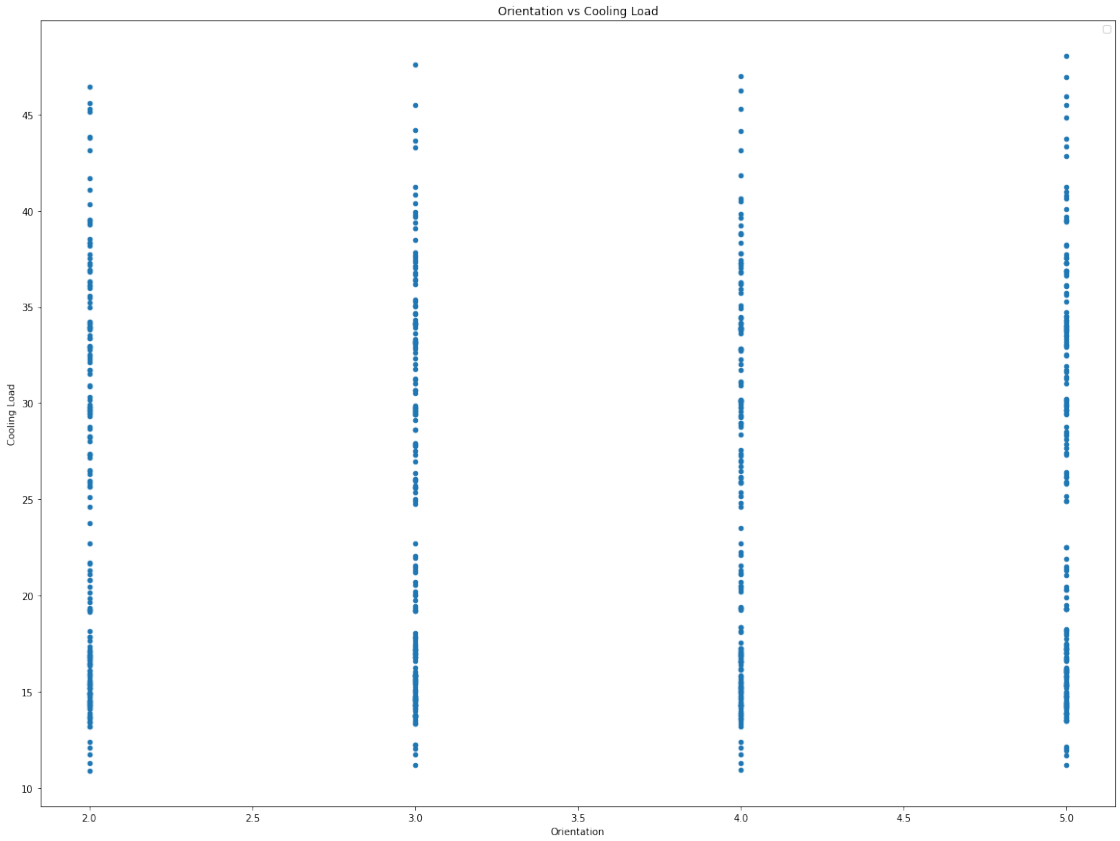

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



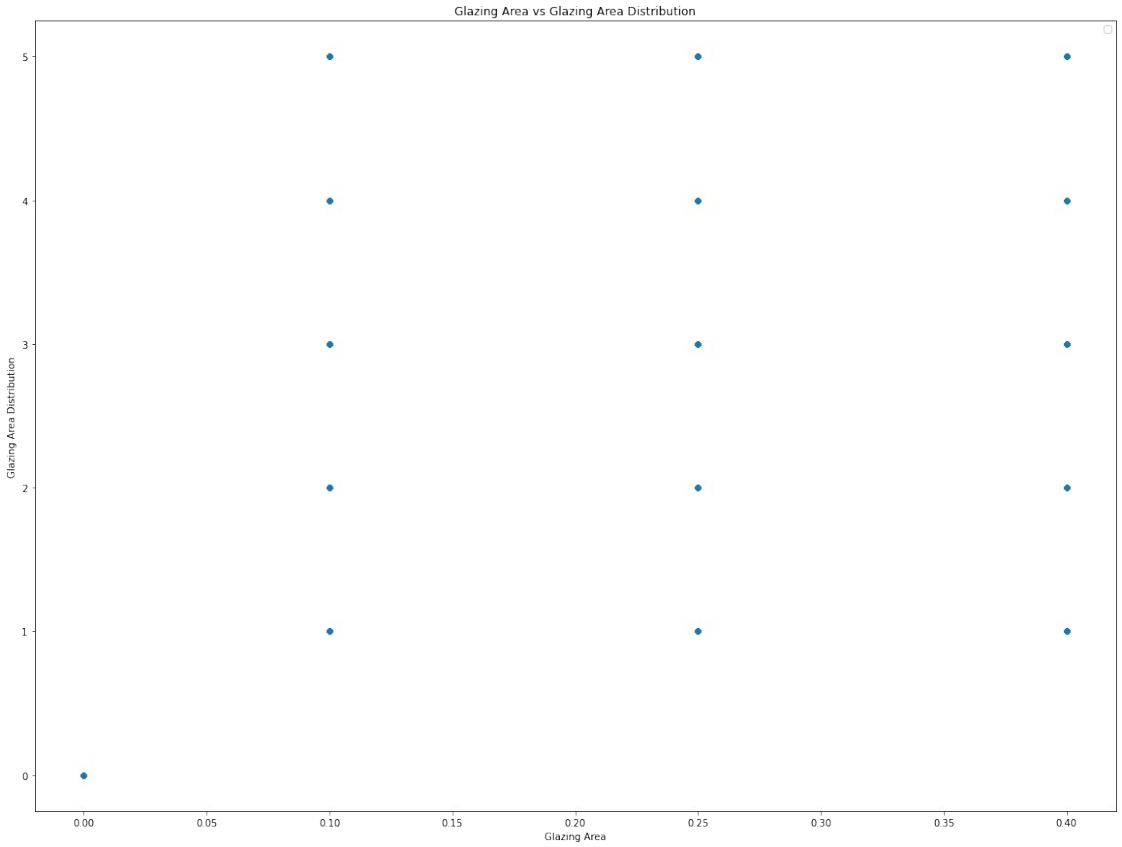

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



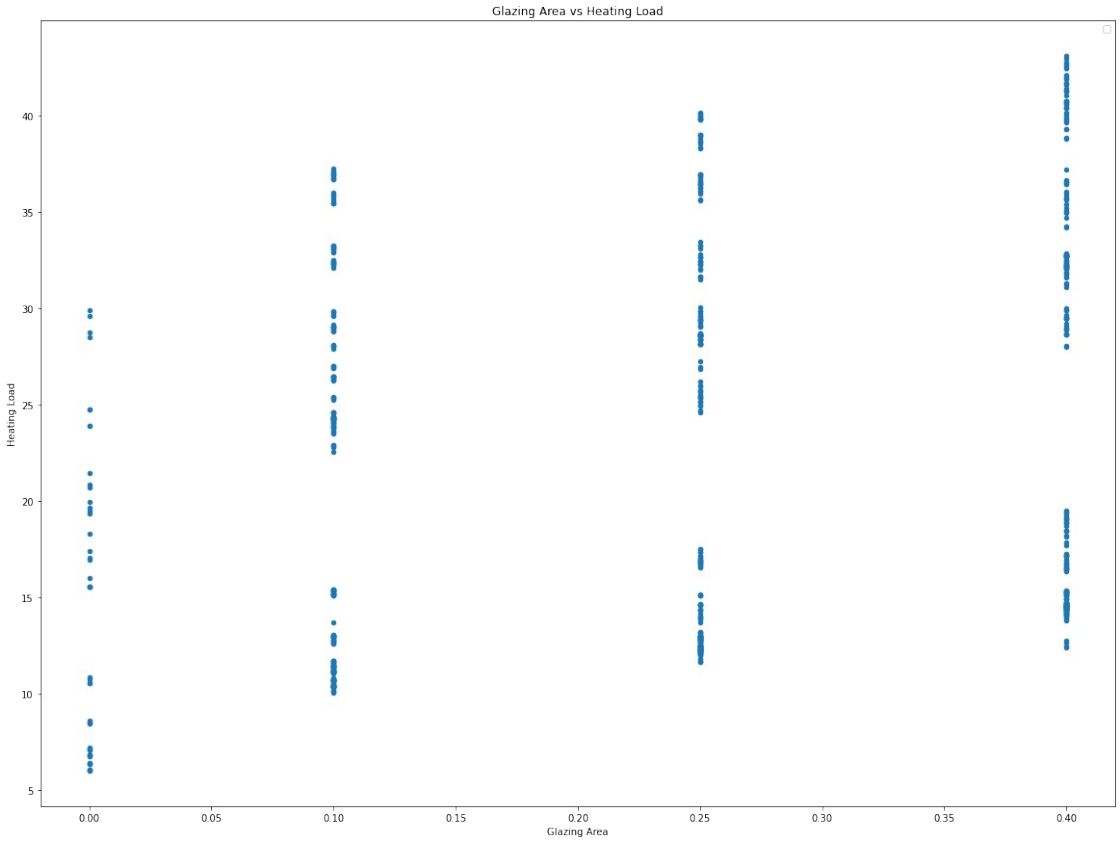

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



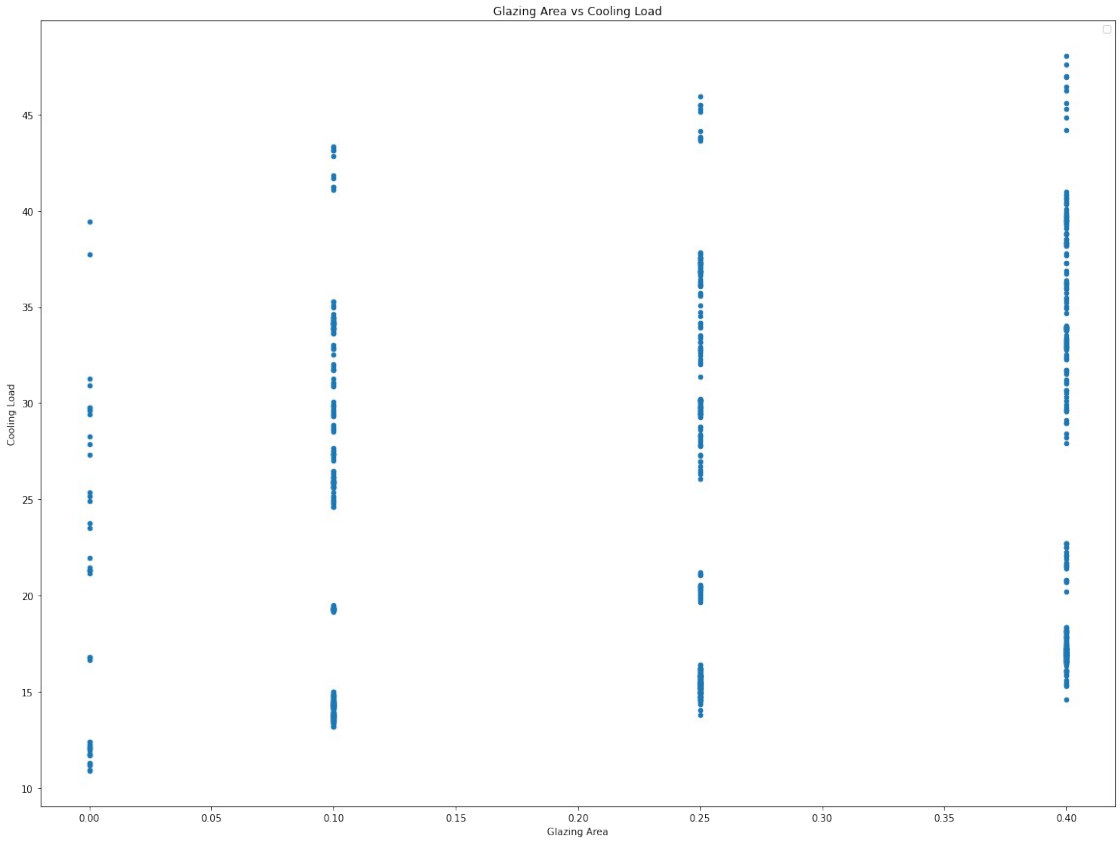

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



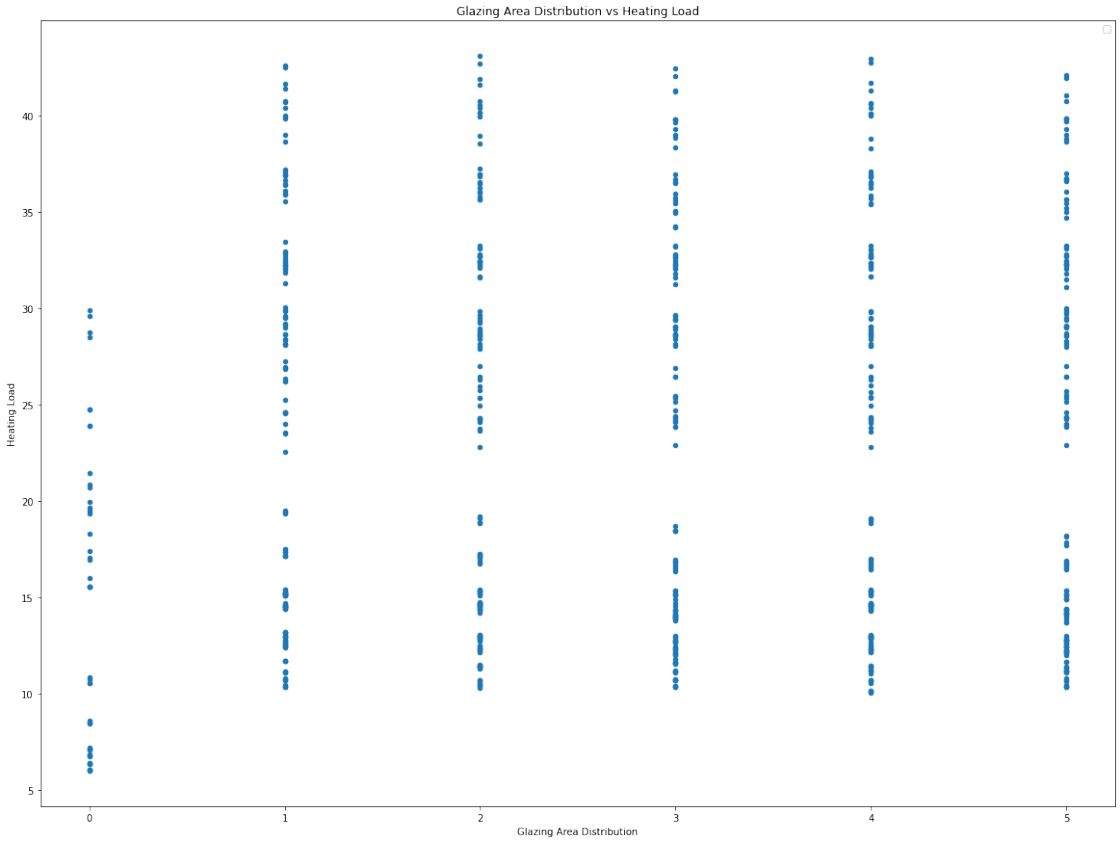

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



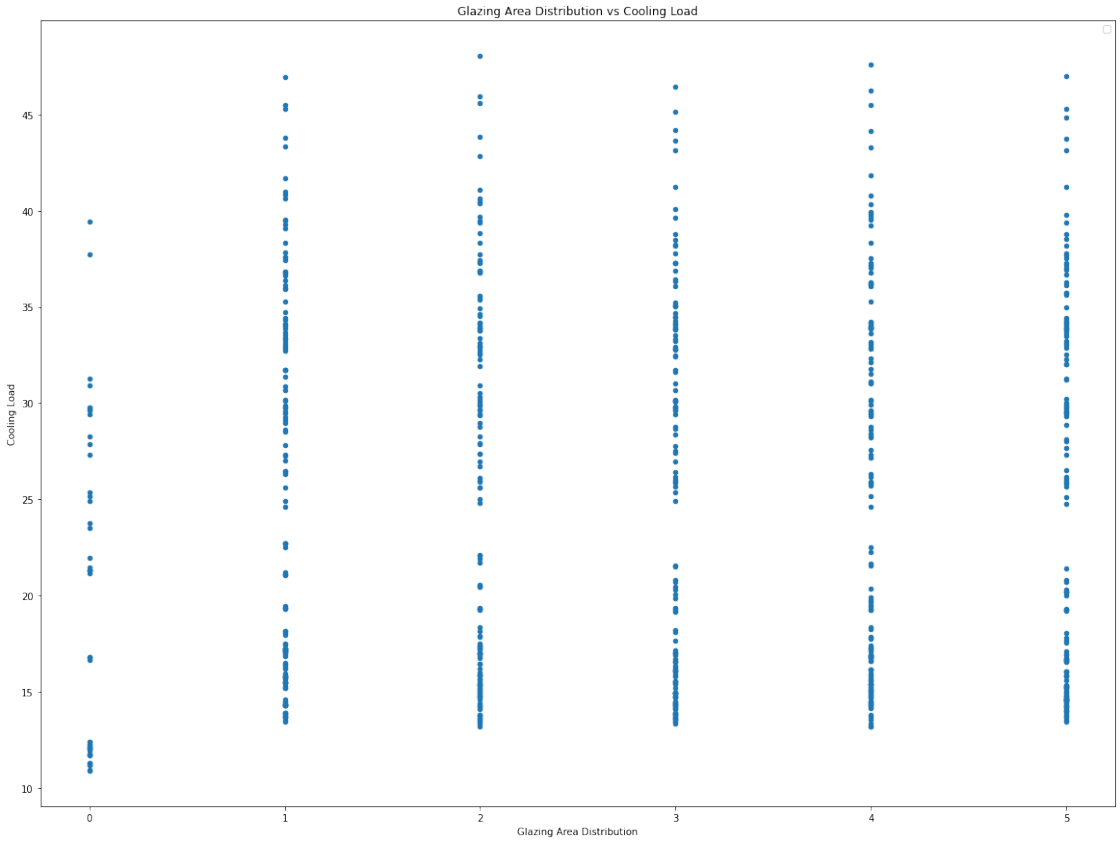

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



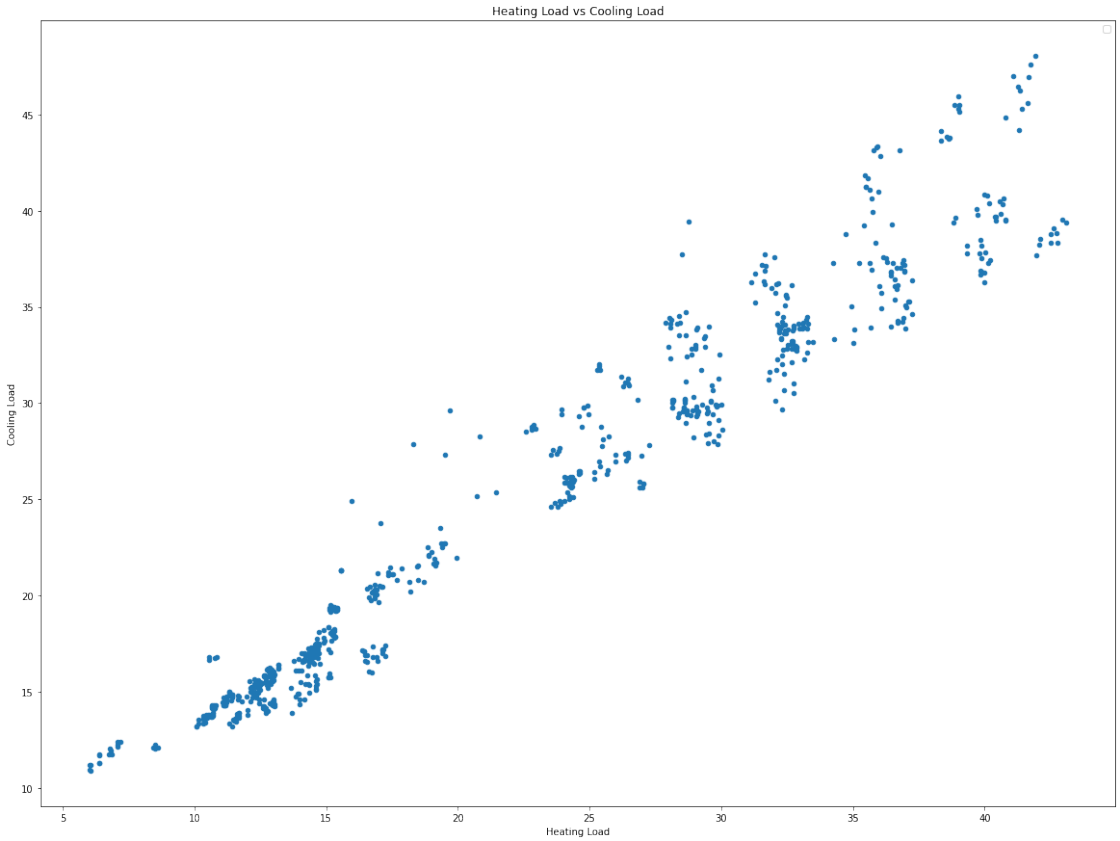

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```




```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```

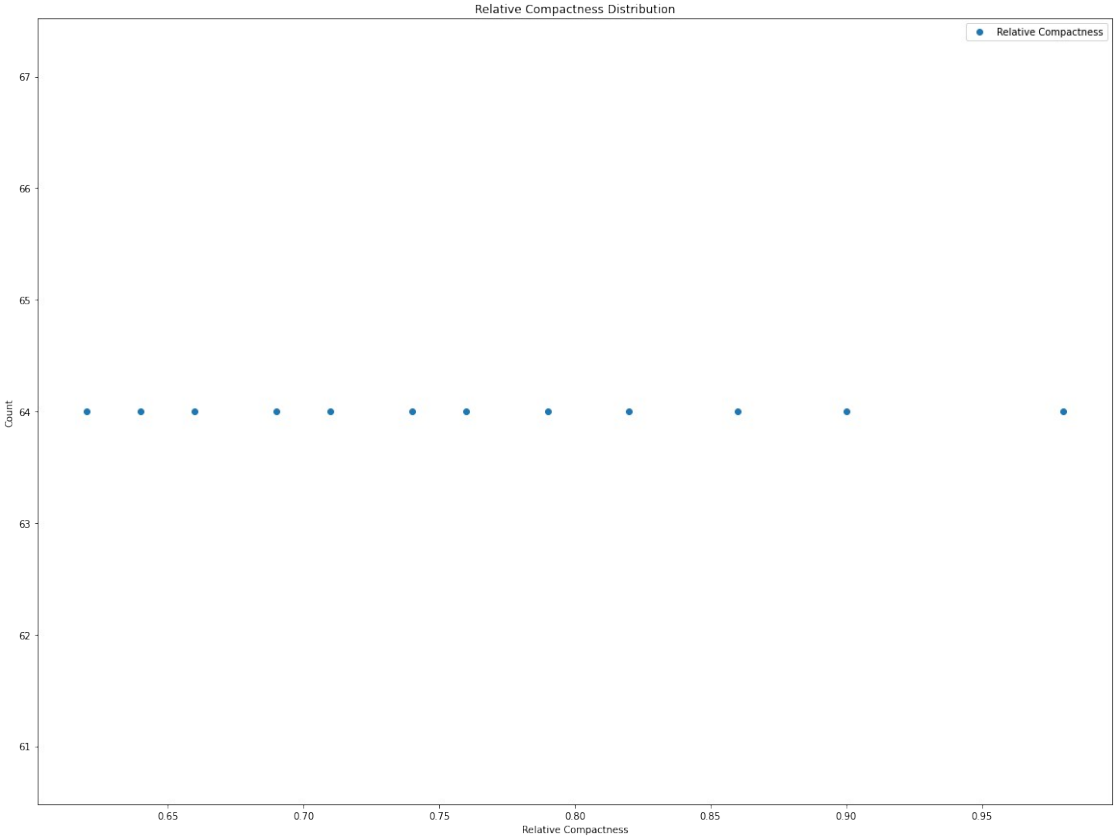


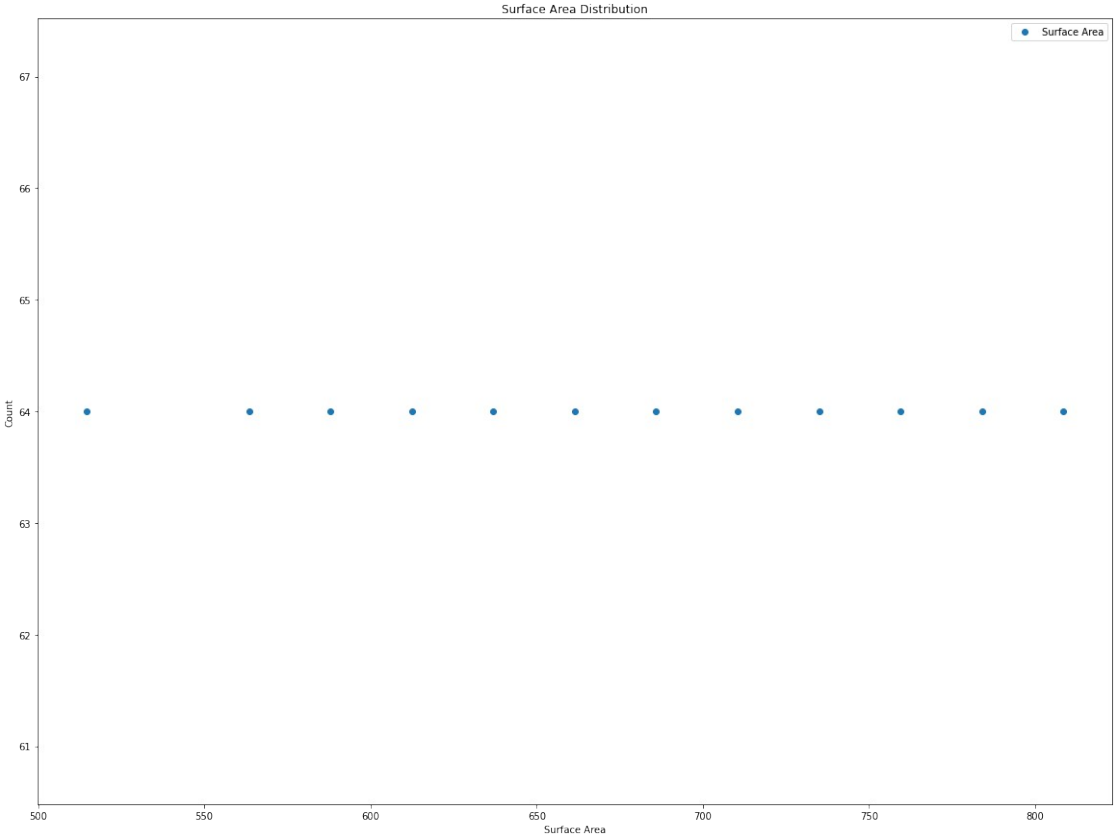

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```

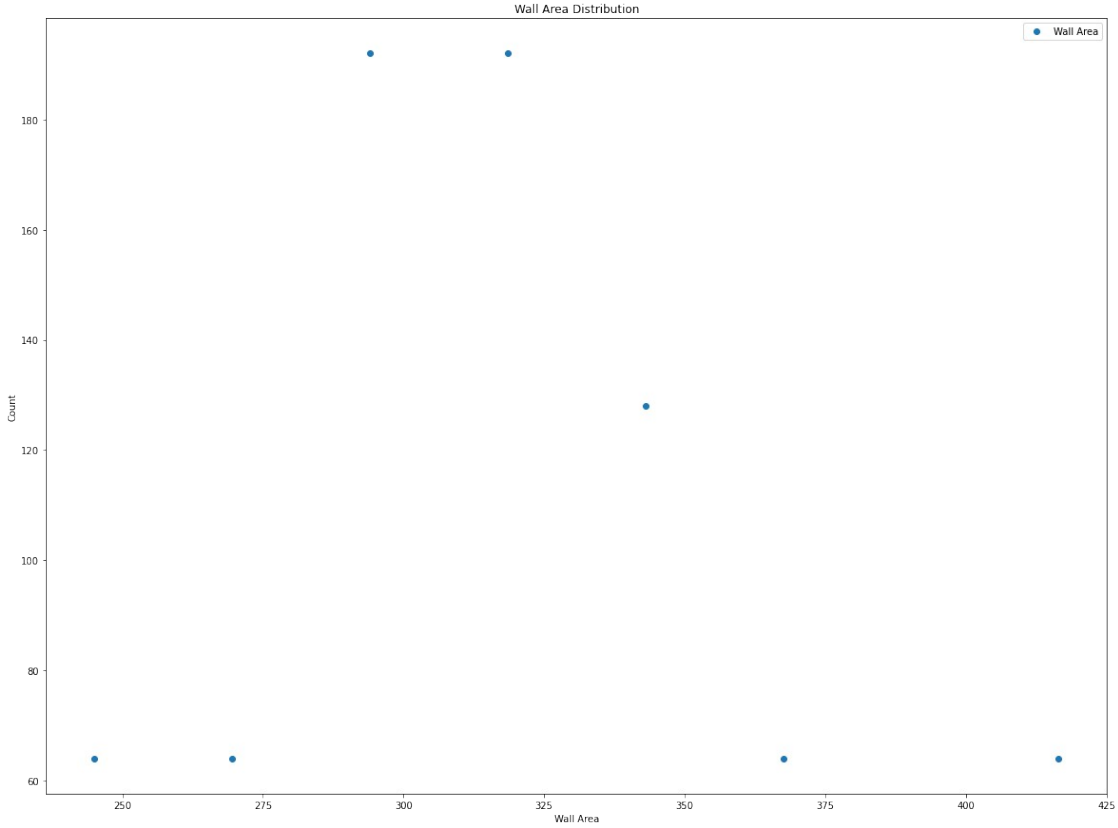


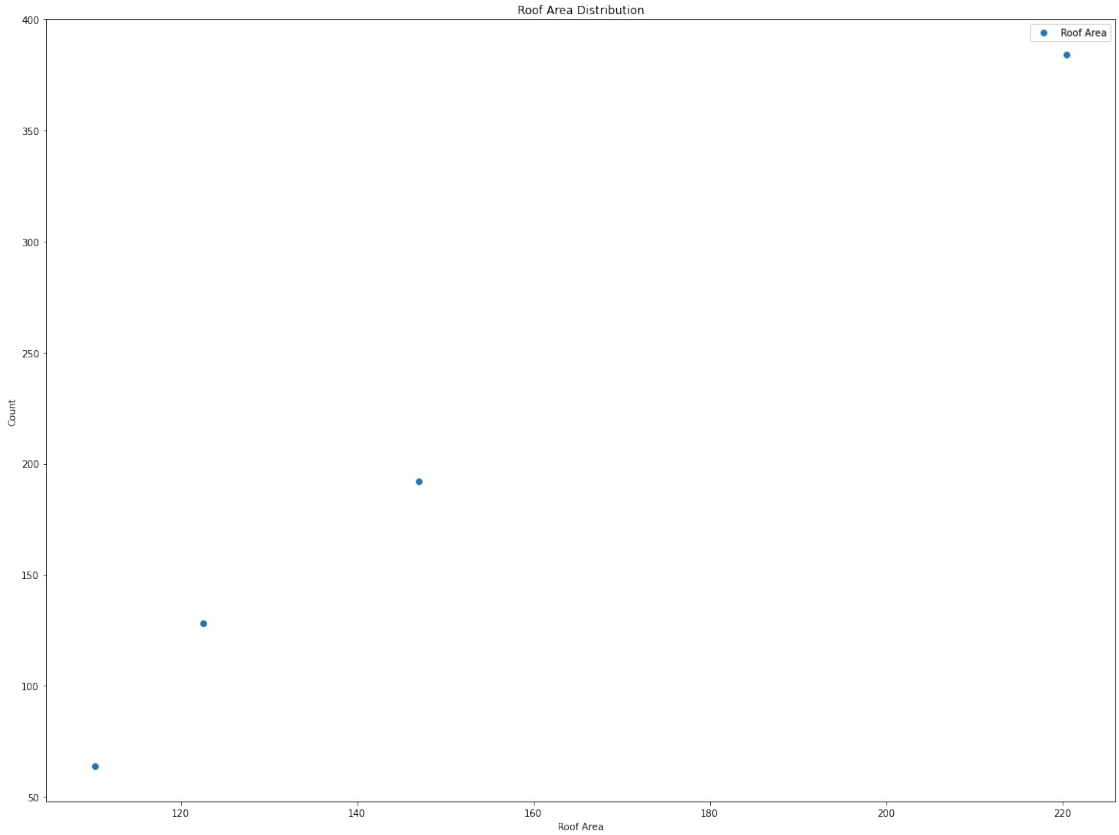
DOT PLOT

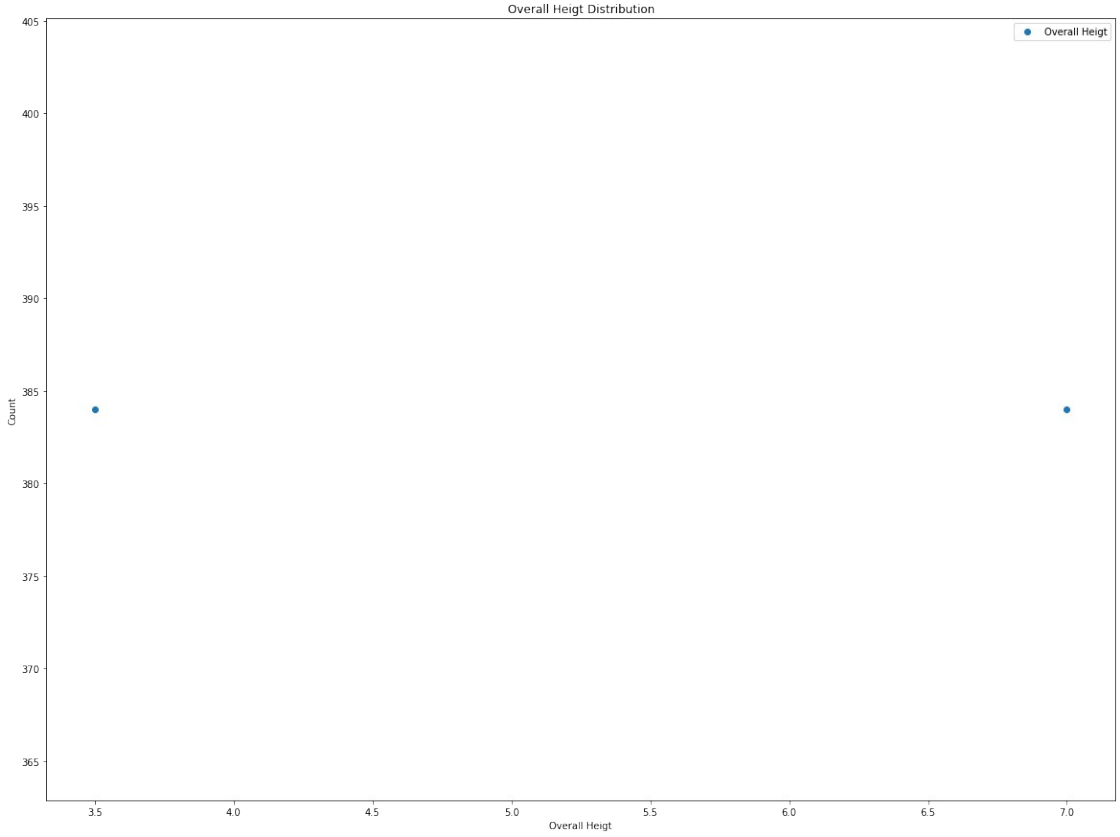
```
# read the data from CSV file into a pandas DataFrame
df = pd.read_csv('dataset.csv')

# iterate through the columns of the DataFrame and create a dot plot
for each column
for col in df.columns:
    fig, ax = plt.subplots(figsize=(20, 15))
    df[col].value_counts().sort_index().plot(style='o', ax=ax)
    ax.set_xlabel(col)
    ax.set_ylabel('Count')
    ax.set_title(f'{col} Distribution')
    ax.legend()
    plt.show()
```

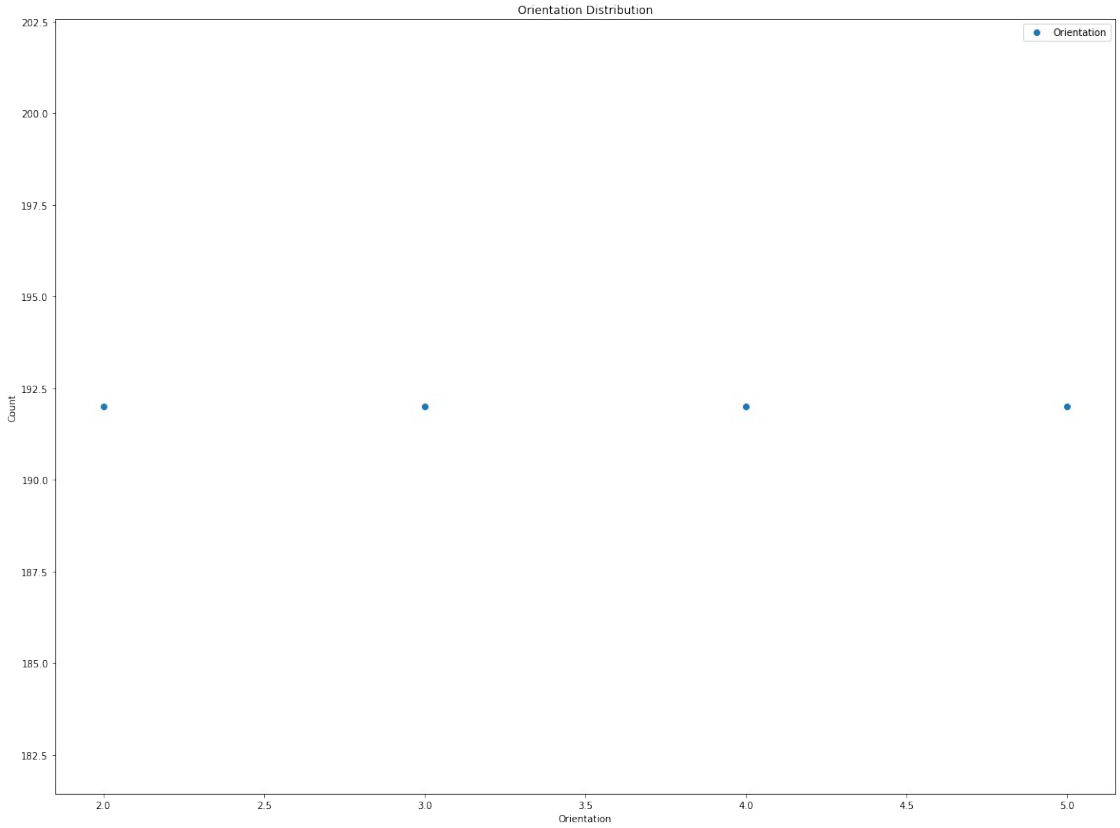

Relative Compactness Distribution
Relative Compactness
67
66
65
64
63
62
61
Count
0.65
0.70
0.75
0.80
0.85
0.90
0.95
Relative Compactness

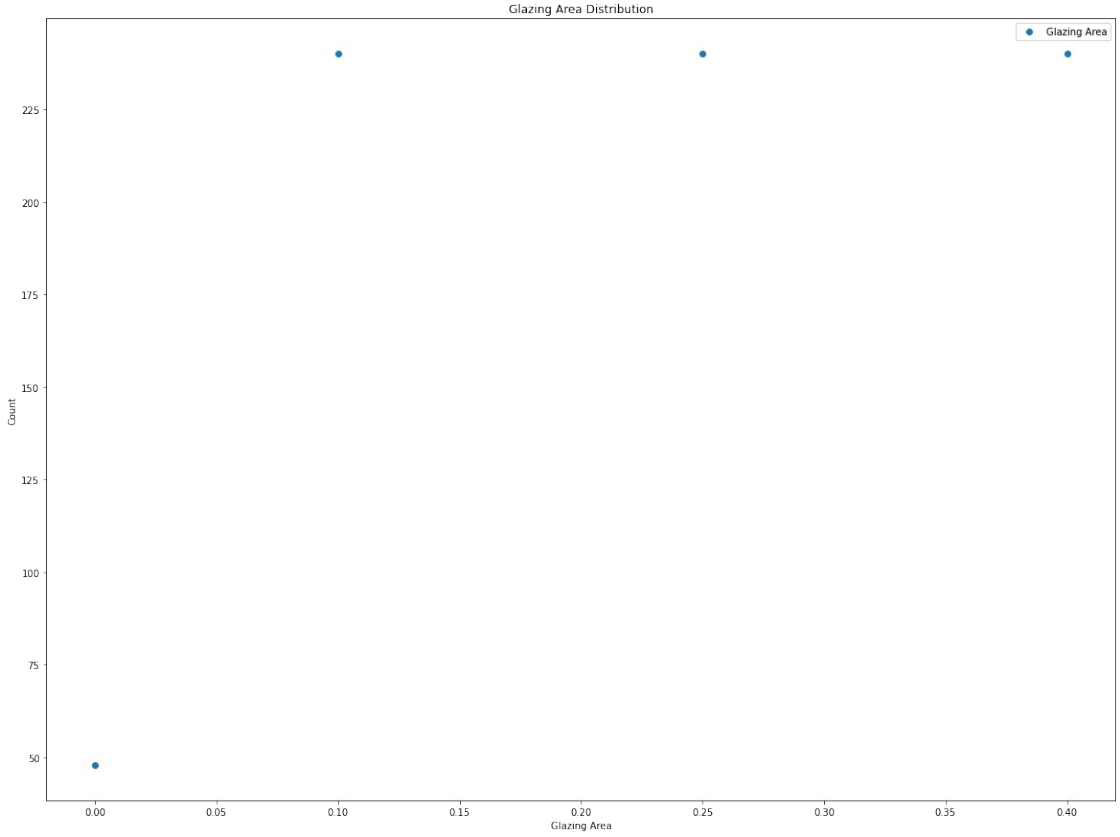

Surface Area Distribution
Surface Area
67
66
65
64
63
62
61
Count
500
550
600
650
700
750
800
Surface Area

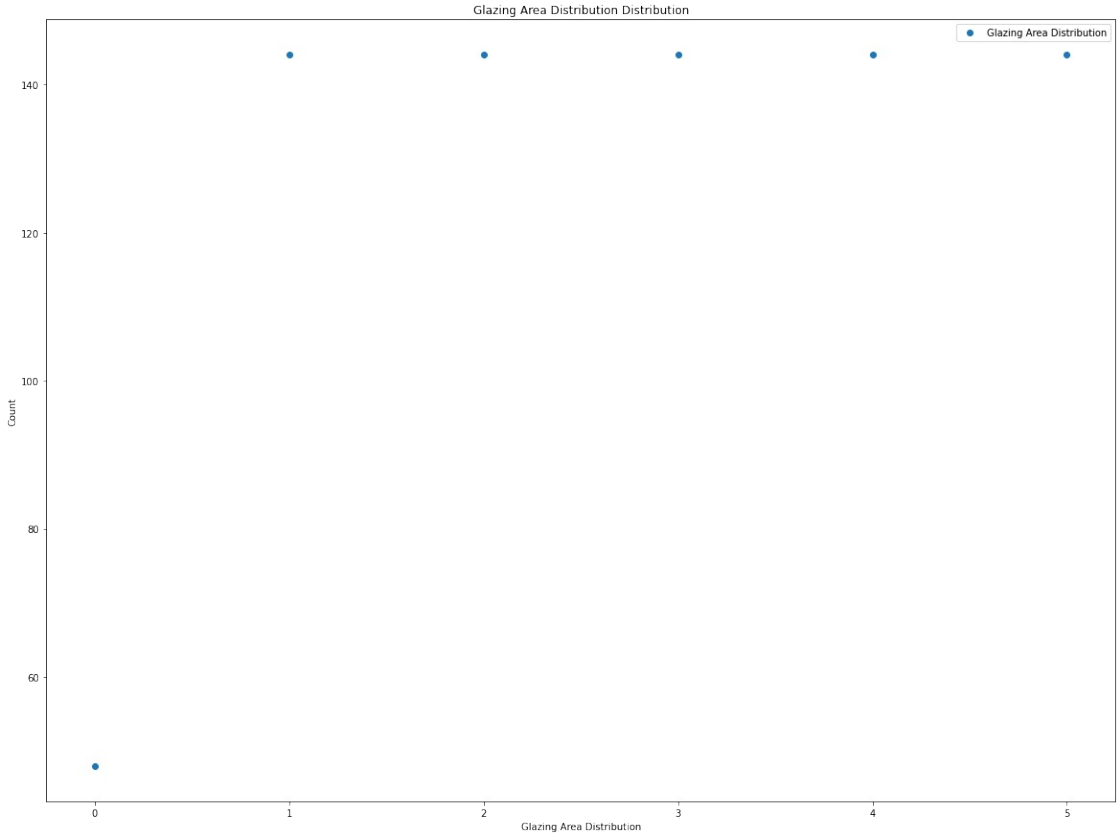

Wall Area Distribution
Wall Area
Count
Wall Area
60
80
100
120
140
160
180
250
275
300
325
350
375
400
425

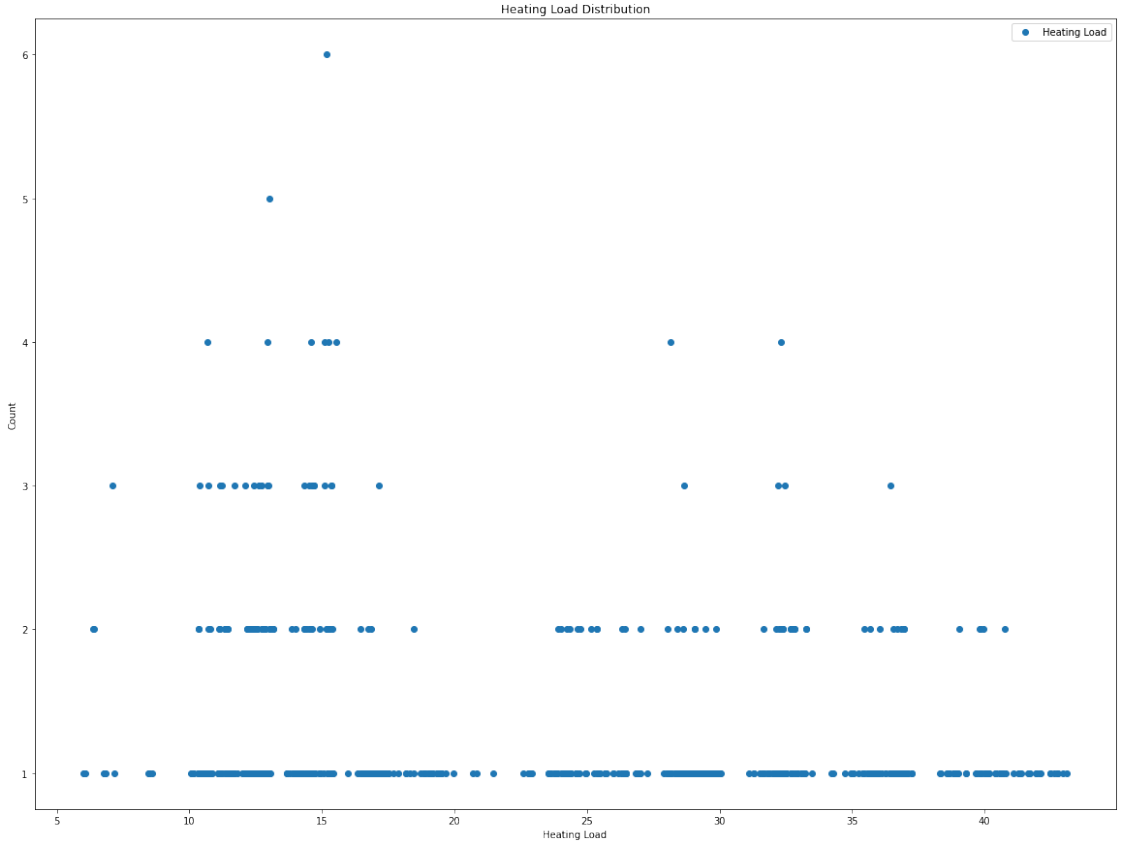

Roof Area Distribution
Roof Area
Count
Roof Area
50
100
150
200
250
300
350
400
120
140
160
180
200
220

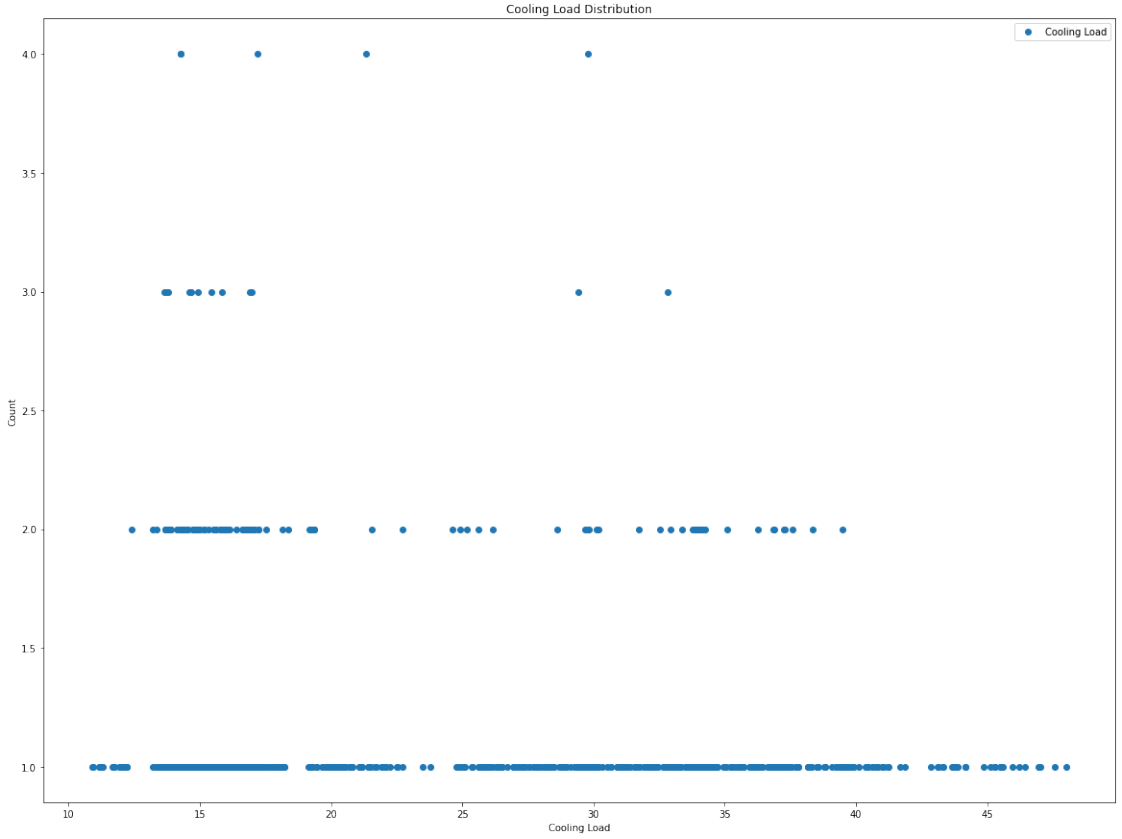

Overall Heigt Distribution
Overall Heigt
Count
Overall Heigt
405
400
395
390
385
380
375
370
365
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0


Orientation Distribution
Orientation
Count
Orientation
202.5
200.0
197.5
195.0
192.5
190.0
187.5
185.0
182.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Glazing Area Distribution
Glazing Area
225
200
175
150
125
100
75
50
Count
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Glazing Area

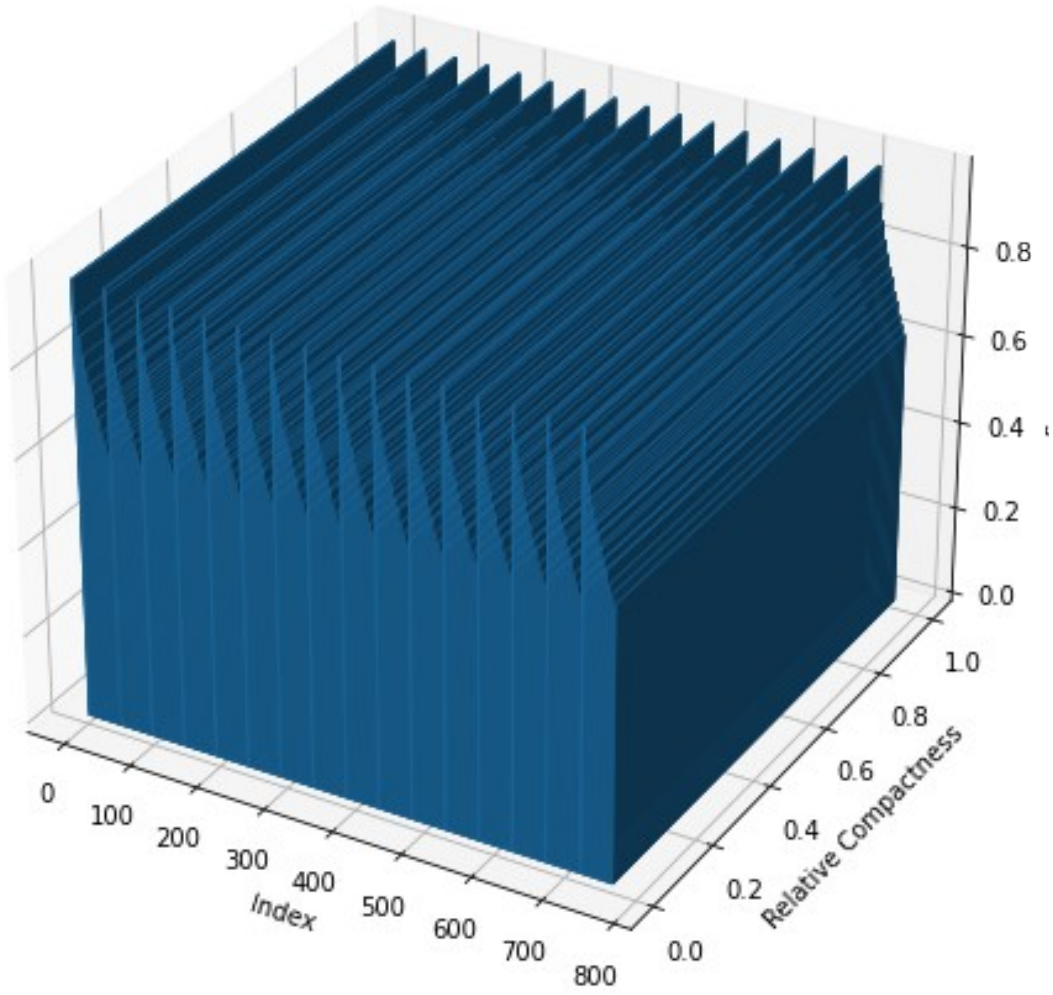

Glazing Area Distribution Distribution
Glazing Area Distribution
140
120
100
80
60
Count
0
1
2
3
4
5
Glazing Area Distribution

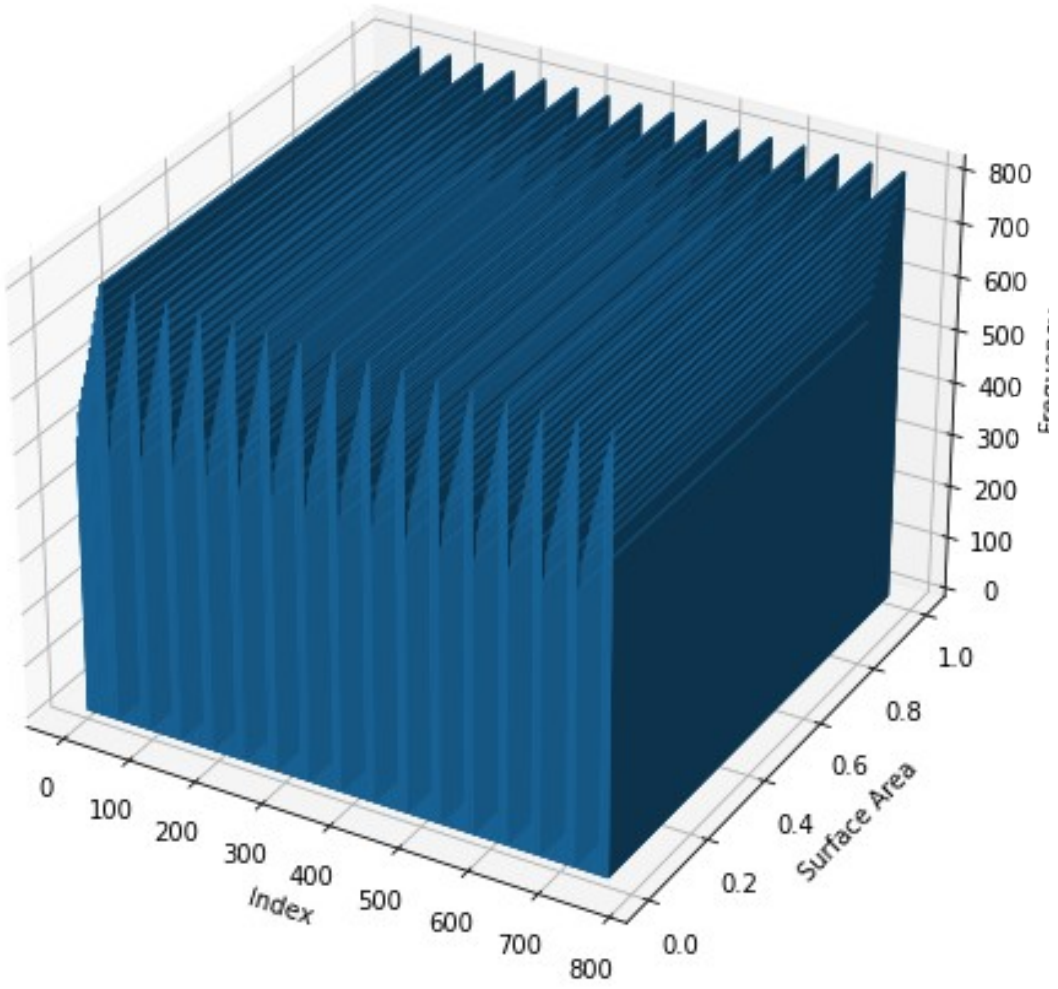

Heating Load Distribution
Heating Load
Count
Heating Load
1
2
3
4
5
6
5
10
15
20
25
30
35
40

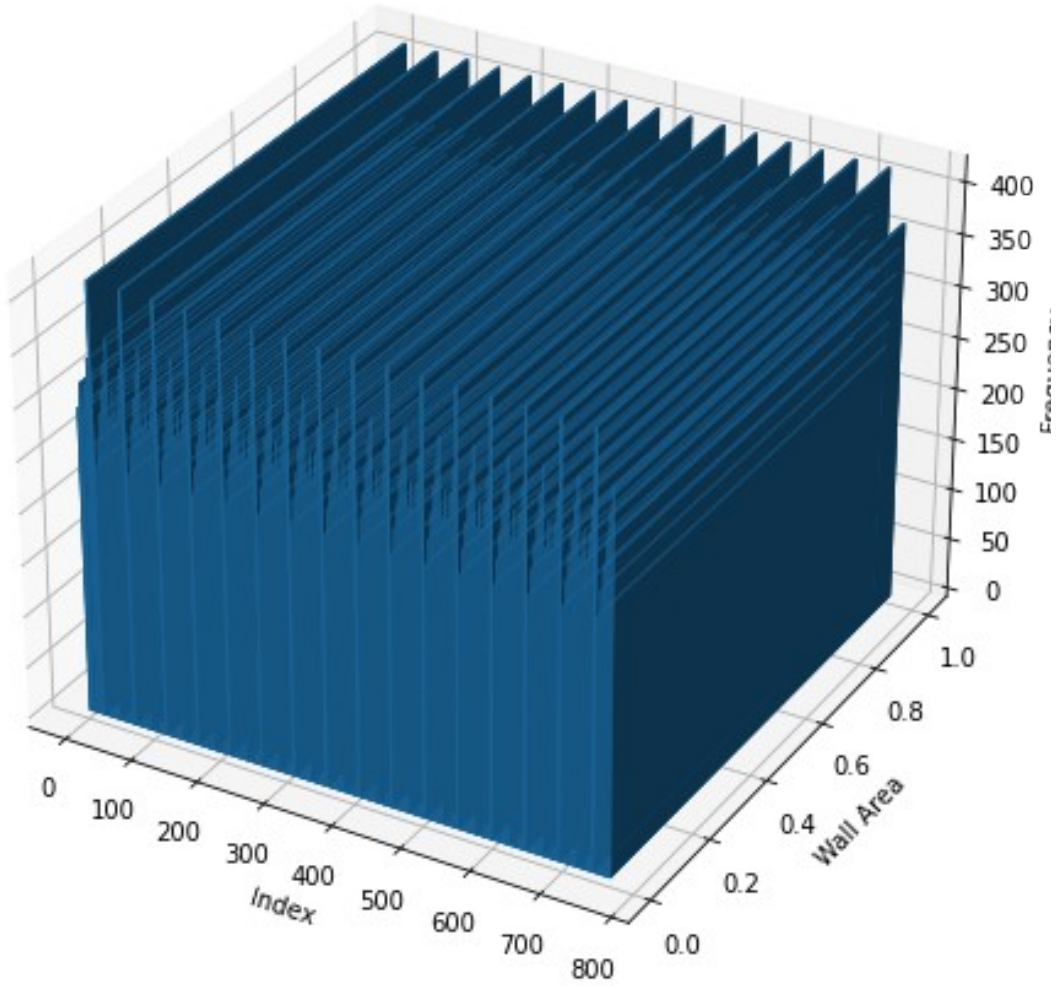

Cooling Load Distribution
Cooling Load
Count
Cooling Load
1.0
1.5
2.0
2.5
3.0
3.5
4.0
10
15
20
25
30
35
40
45

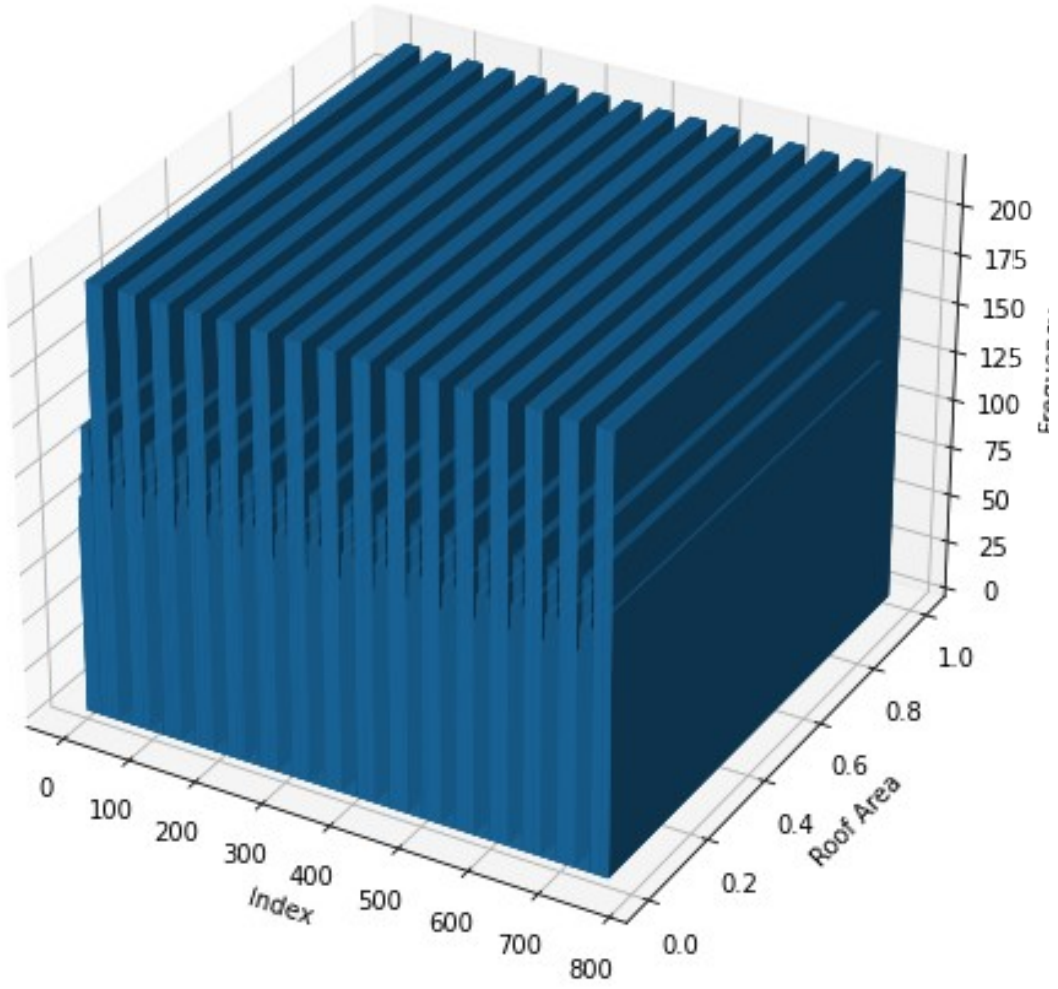
3D BAR CHART

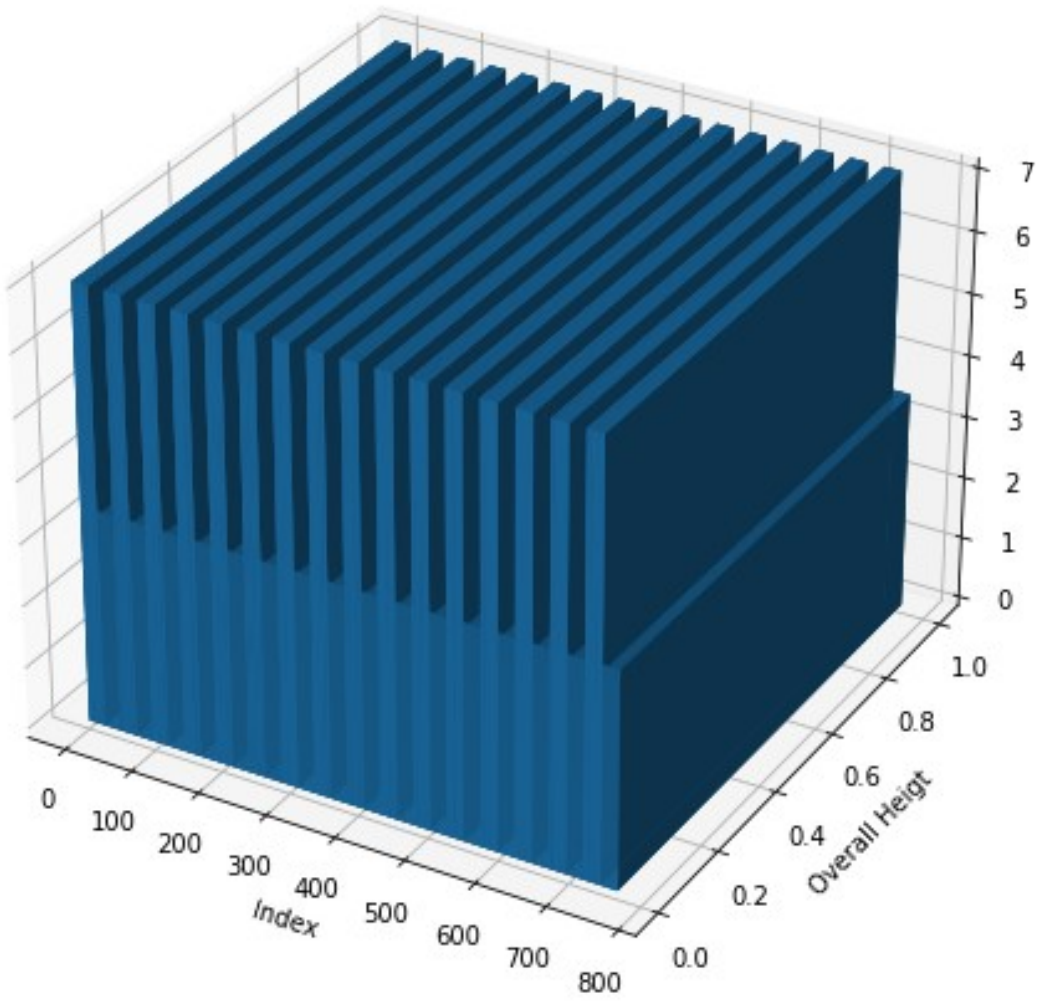
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# load data from csv file
data = pd.read_csv("dataset.csv")

# create a 3D bar chart for each attribute in the data
for column in data.columns:
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')
    x_data = np.arange(len(data[column]))
    y_data = np.zeros(len(data[column]))
    z_data = data[column]
    dx = np.ones(len(data[column]))
    dy = np.ones(len(data[column]))
    dz = z_data
    ax.bar3d(x_data, y_data, np.zeros_like(z_data), dx, dy, dz)
    ax.set_xlabel('Index')
    ax.set_ylabel(column)
    ax.set_zlabel('Frequency')
    plt.show()
```

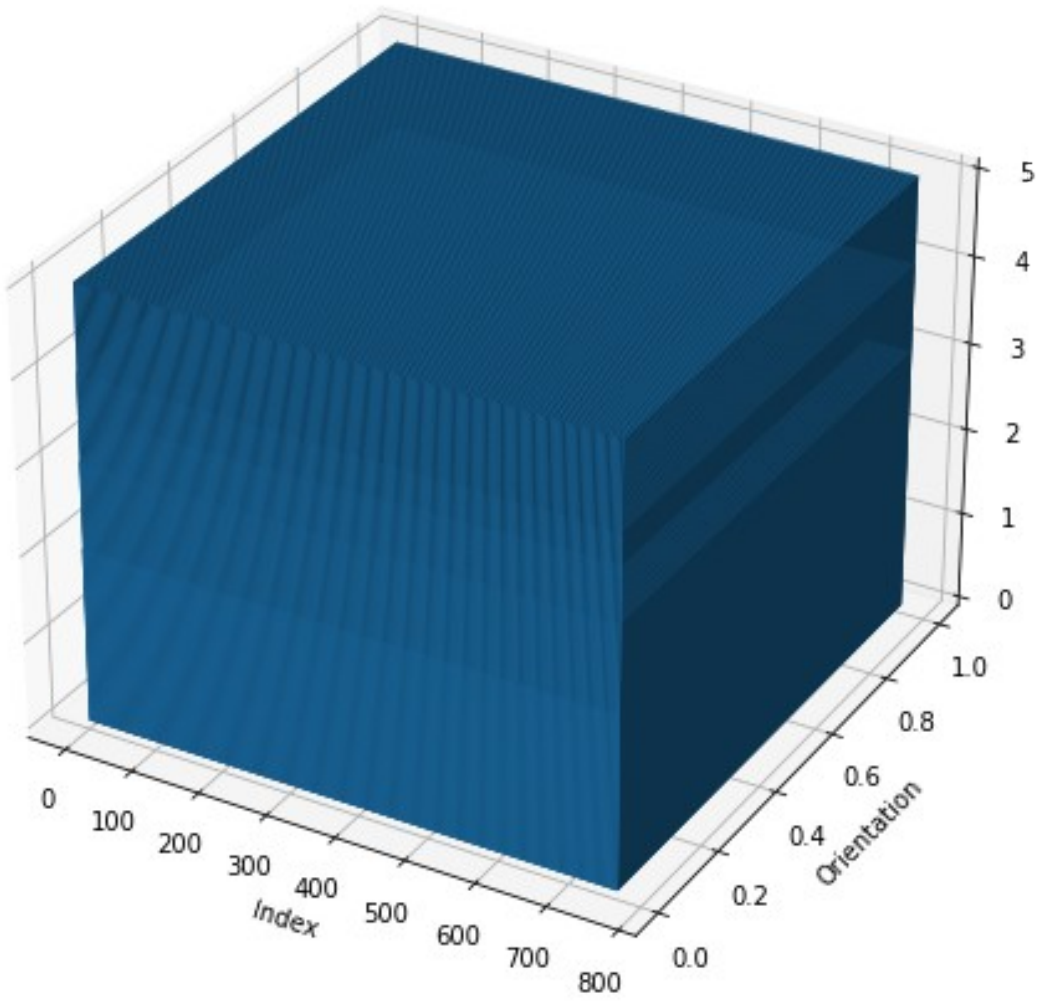

0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
Relative Compactness
0
100
200
300
400
500
600
700
800
Index

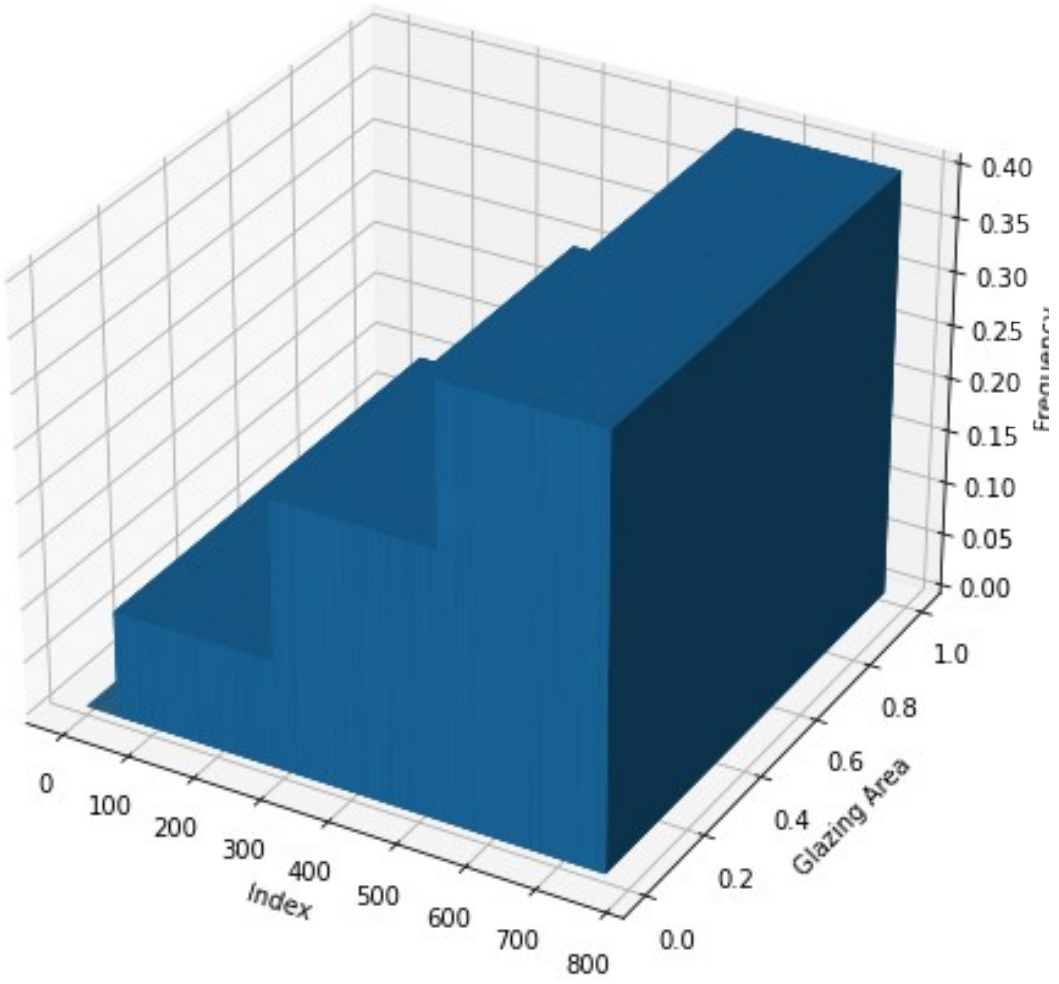

800
700
600
500
400
300
200
100
0
1.0
0.8
0.6
0.4
0.2
0.0
Surface Area
0
100
200
300
400
500
600
700
800
Index

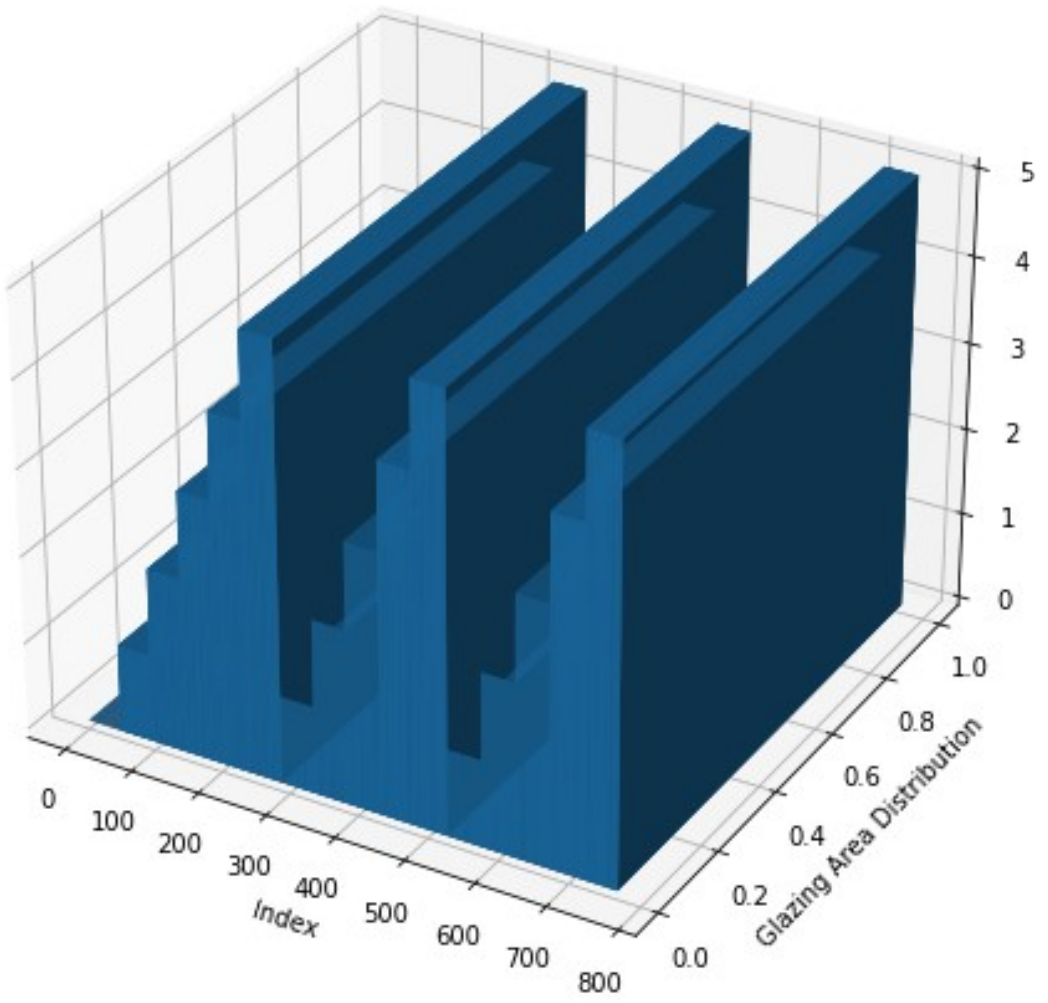

400
350
300
250
200
150
100
50
0
1.0
0.8
0.6
0.4
0.2
0.0
Wall Area
0
100
200
300
400
500
600
700
800
Index

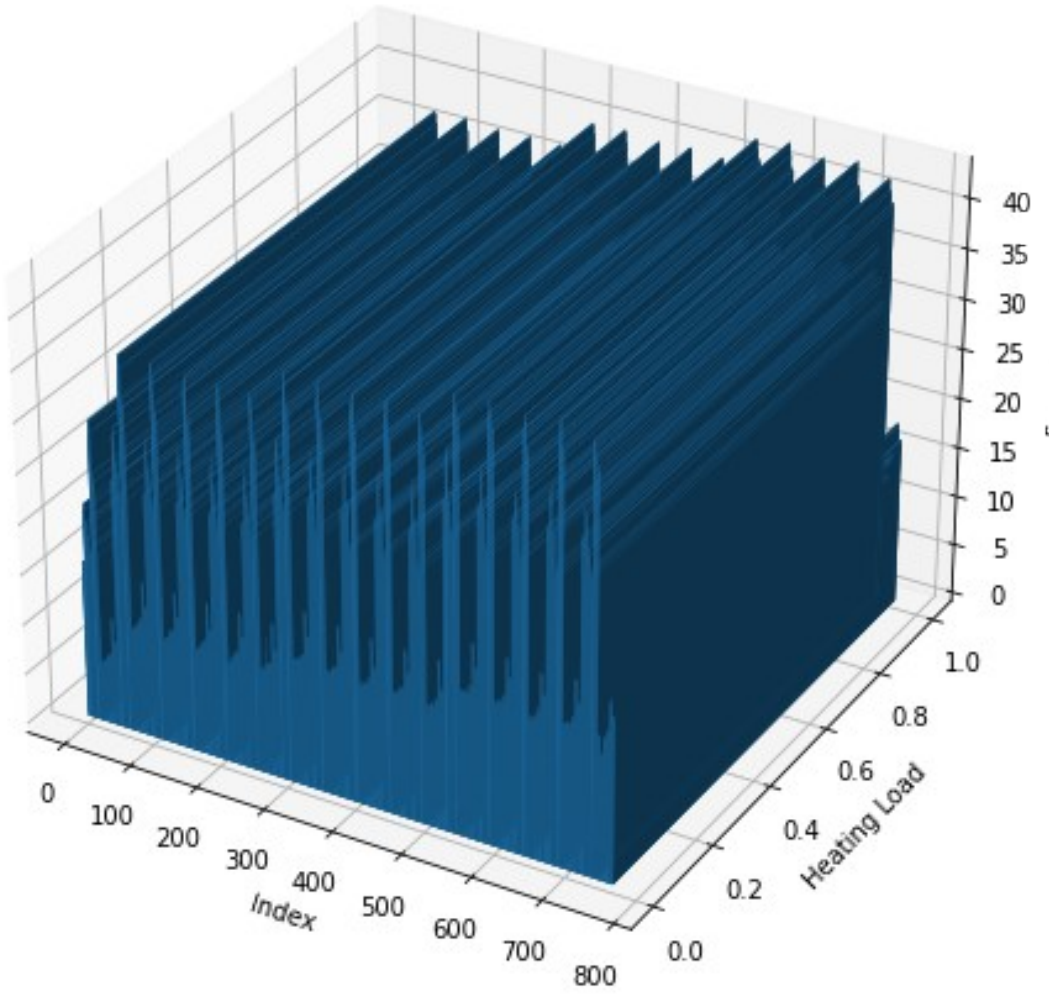

0
100
200
300
400
500
600
700
800
Index
0.0
0.2
0.4
0.6
0.8
1.0
Roof Area
0
25
50
75
100
125
150
175
200

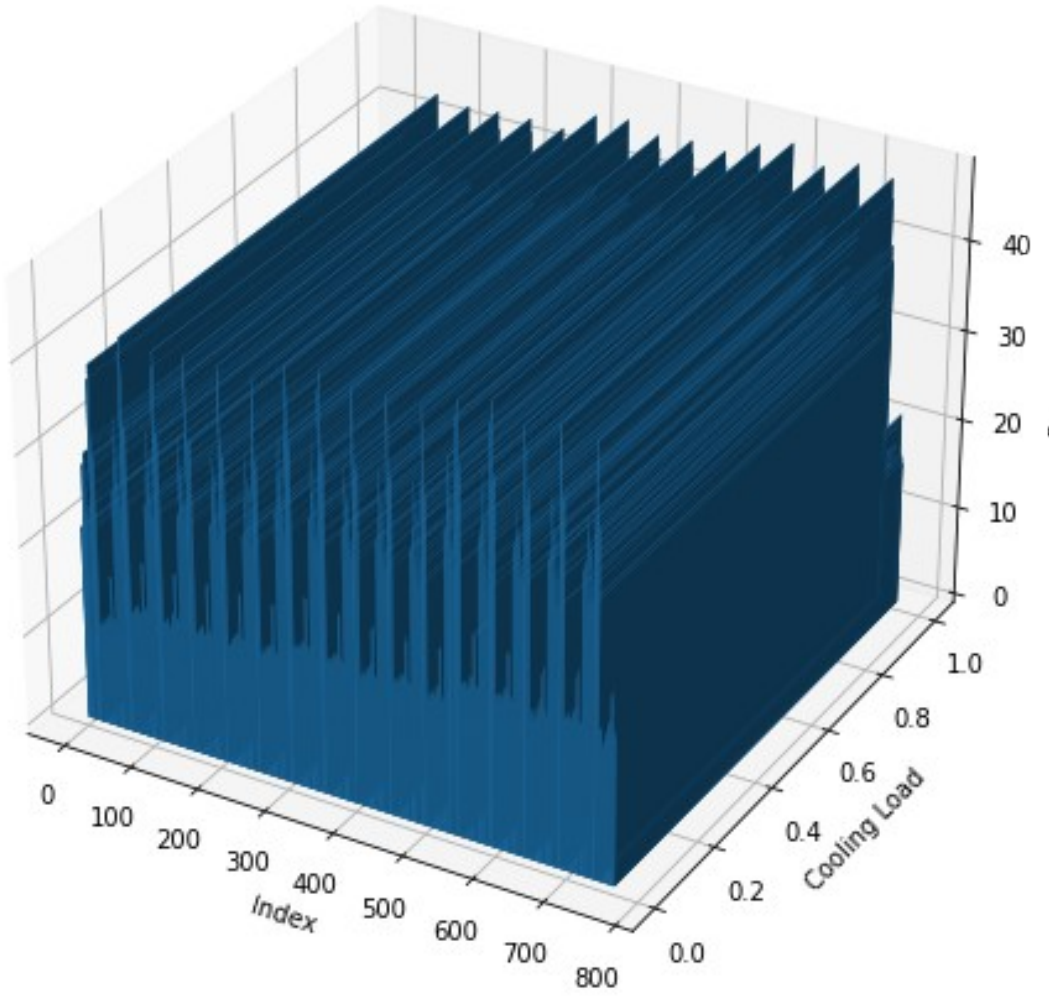

7
6
5
4
3
2
1
0
1.0
0.8
0.6
0.4
0.2
0.0
Overall Heigt
0
100
200
300
400
500
600
700
800
Index


5
4
3
2
1
0
1.0
0.8
0.6
0.4
0.2
0.0
Orientation
0
100
200
300
400
500
600
700
800
Index


0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Frequency
0
100
200
300
400
500
600
700
800
Index
0.0
0.2
0.4
0.6
0.8
1.0
Glazing Area


0
100
200
300
400
500
600
700
800
Index
0.0
0.2
0.4
0.6
0.8
1.0
Glazing Area Distribution
0
1
2
3
4
5

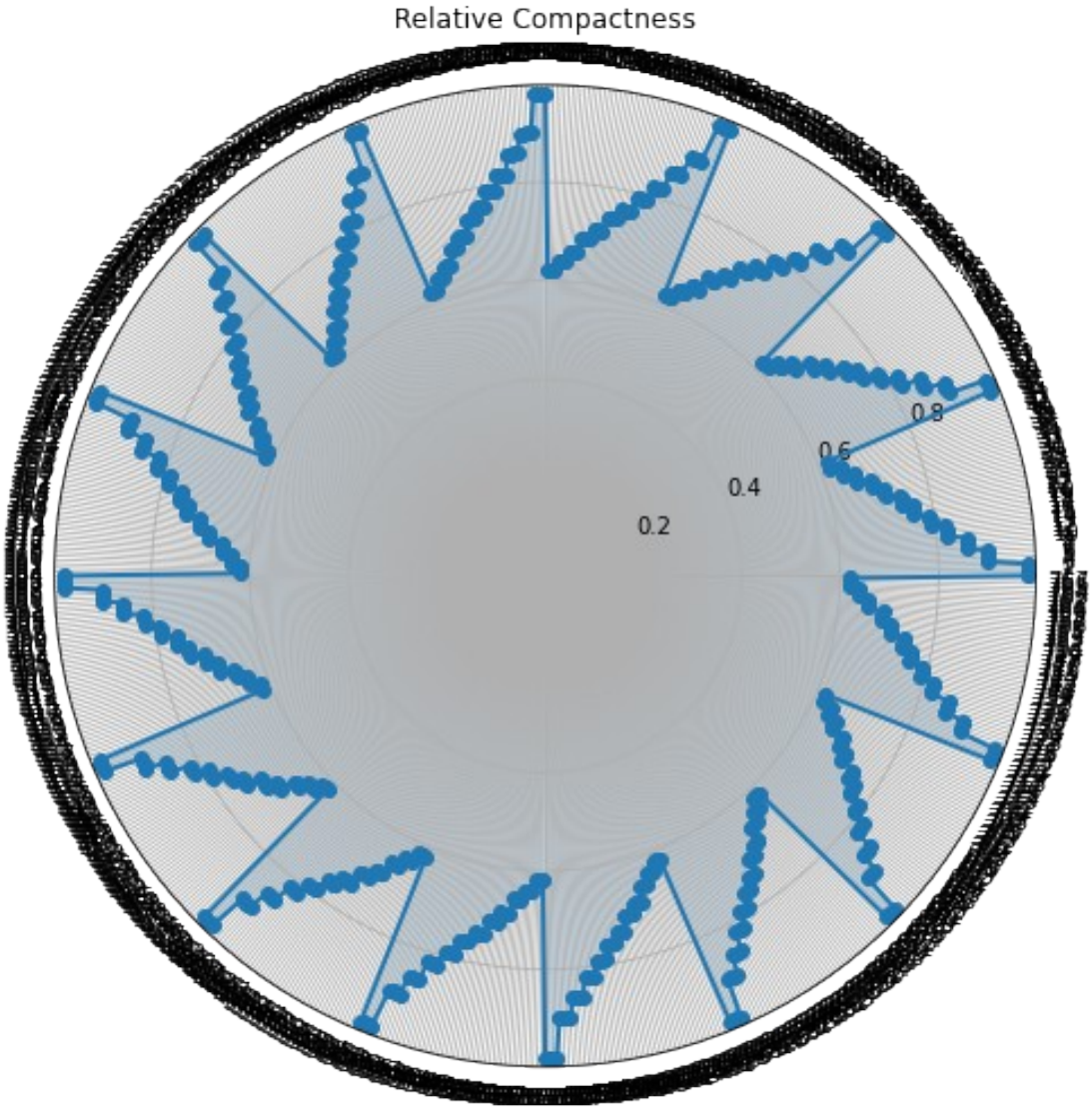

40
35
30
25
20
15
10
5
0
1.0
0.8
0.6
0.4
0.2
0.0
Heating Load
0
100
200
300
400
500
600
700
800
Index

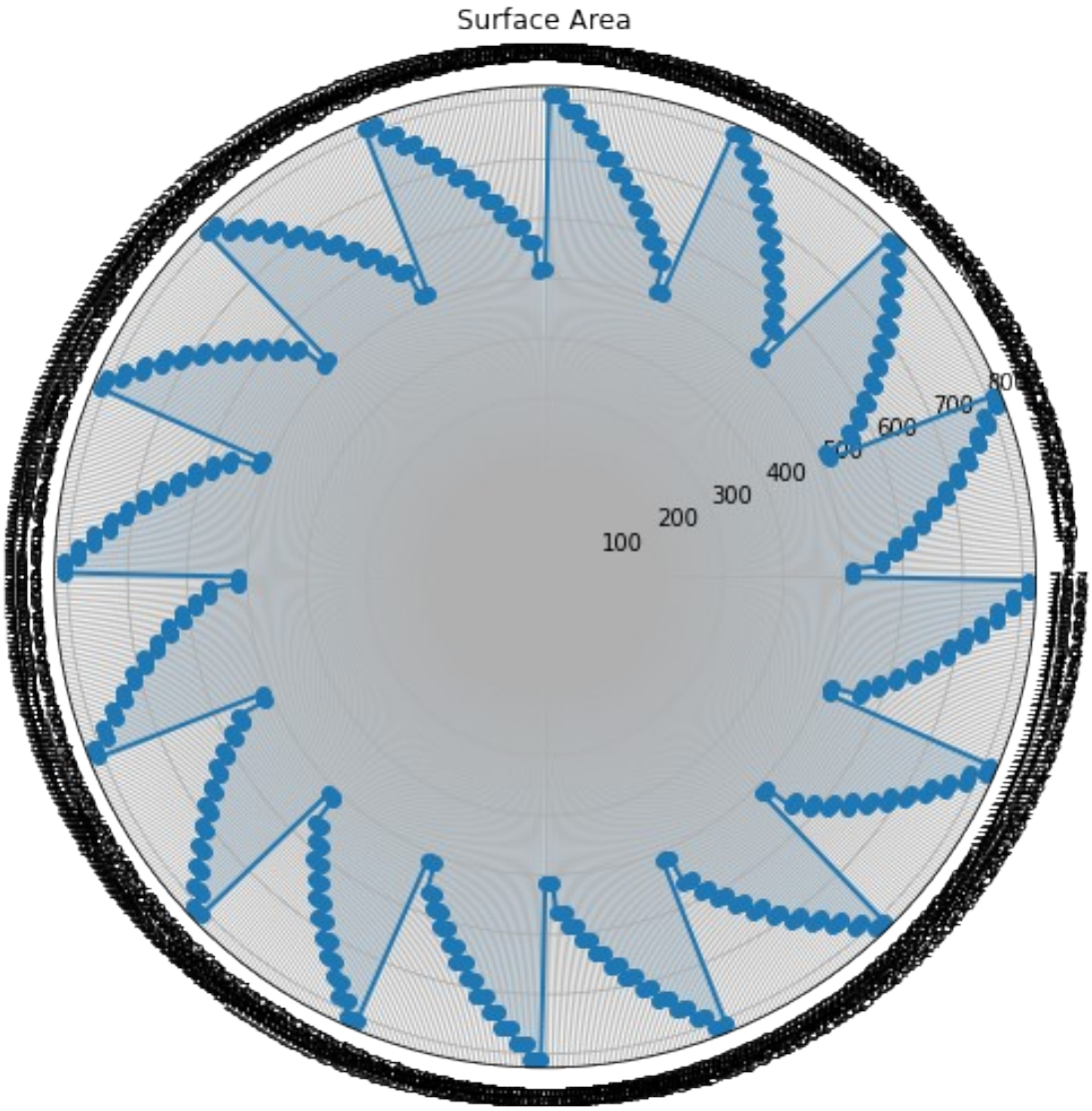


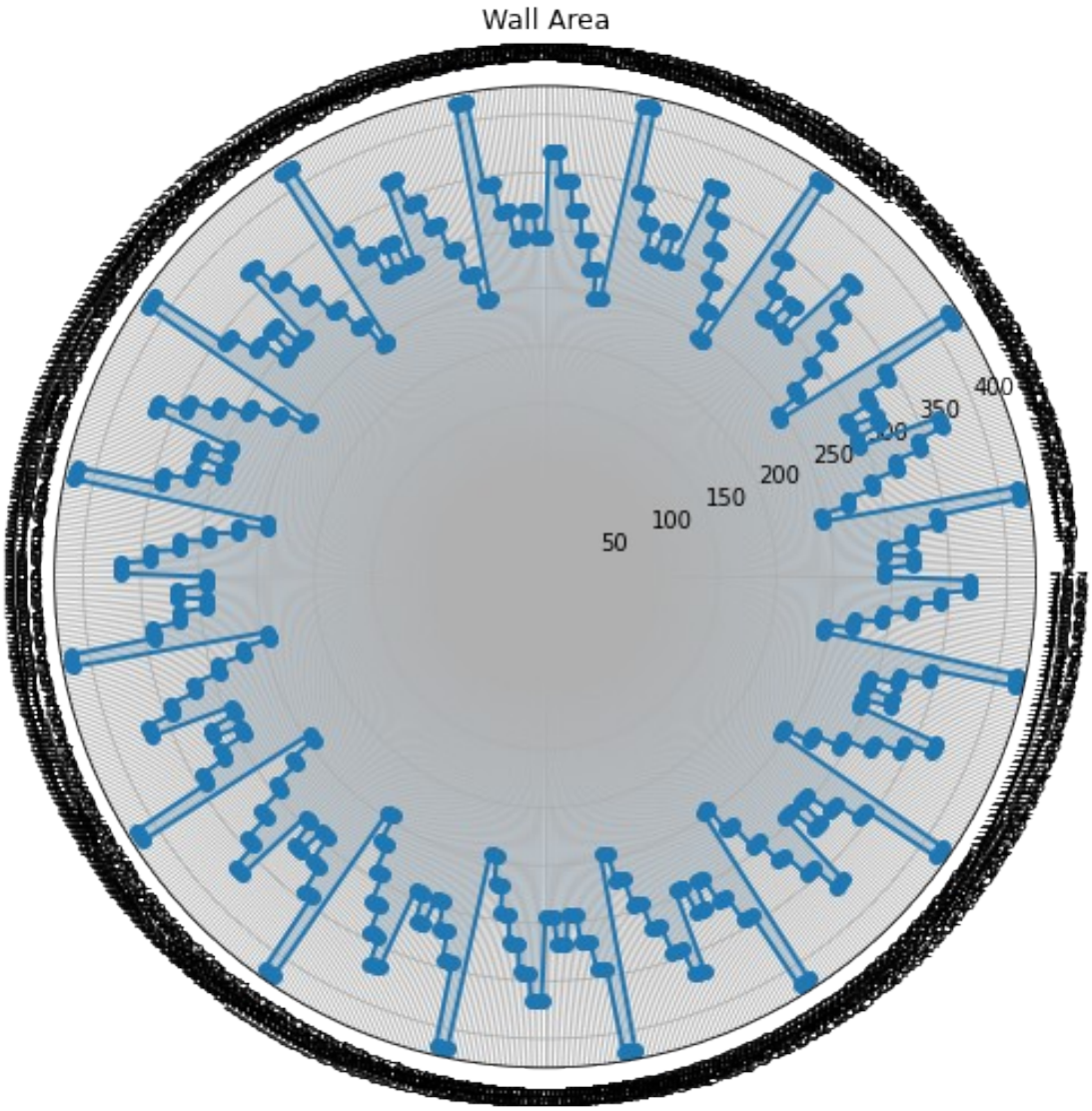
RADIAL BAR PLOT

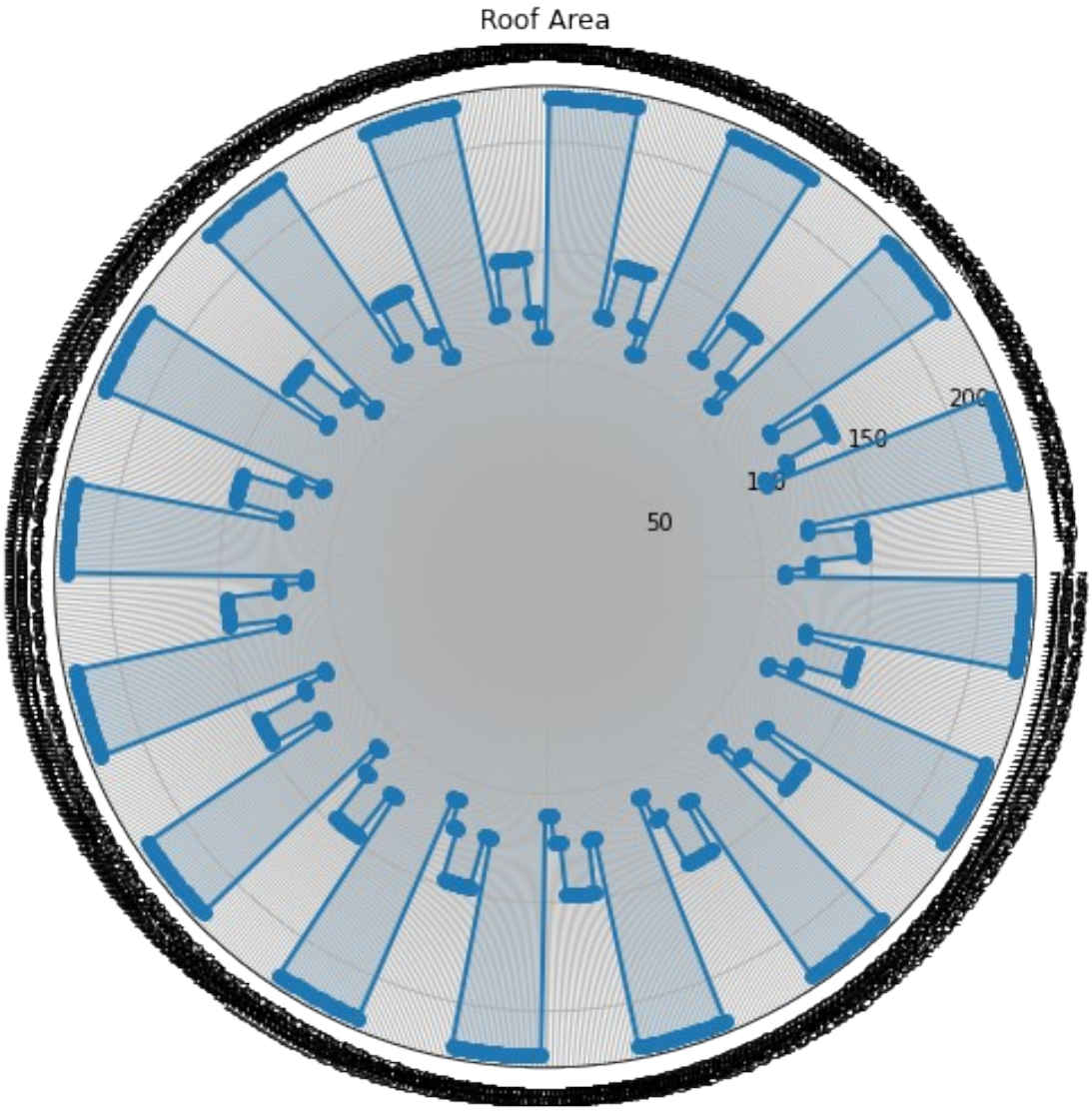
```
# Load data from CSV file
data = pd.read_csv('dataset.csv')

# Define function to create radial bar plot
def radial_bar_plot(data, attribute):
    angles = np.linspace(0, 2*np.pi, len(data[attribute]),
endpoint=False)
    angles = np.concatenate((angles,[angles[0]]))
    values = data[attribute].values.tolist()
    values.append(values[0])

    fig = plt.figure(figsize=(8, 8))
    ax = fig.add_subplot(111, polar=True)
    labels = data.index.tolist()
```

```
    labels.append('')
    ax.plot(angles, values, 'o-', linewidth=2)
    ax.fill(angles, values, alpha=0.25)
    ax.set_thetagrids(angles * 180/np.pi, labels)
    ax.set_title(attribute)
    ax.grid(True)

for attribute in data.columns:
    radial_bar_plot(data, attribute)

plt.show()
```

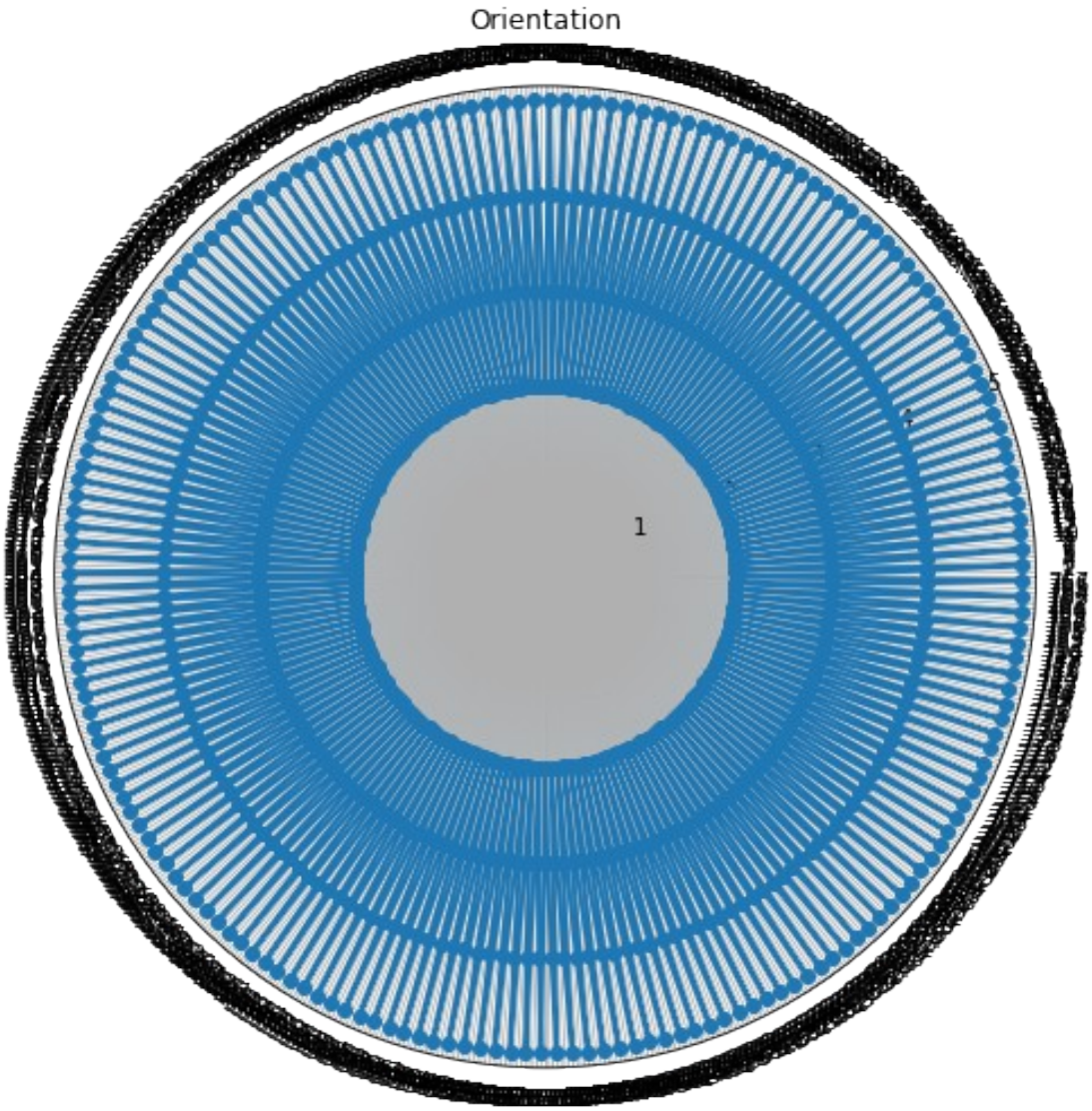



Surface Area
100
200
300
400
500
600
700

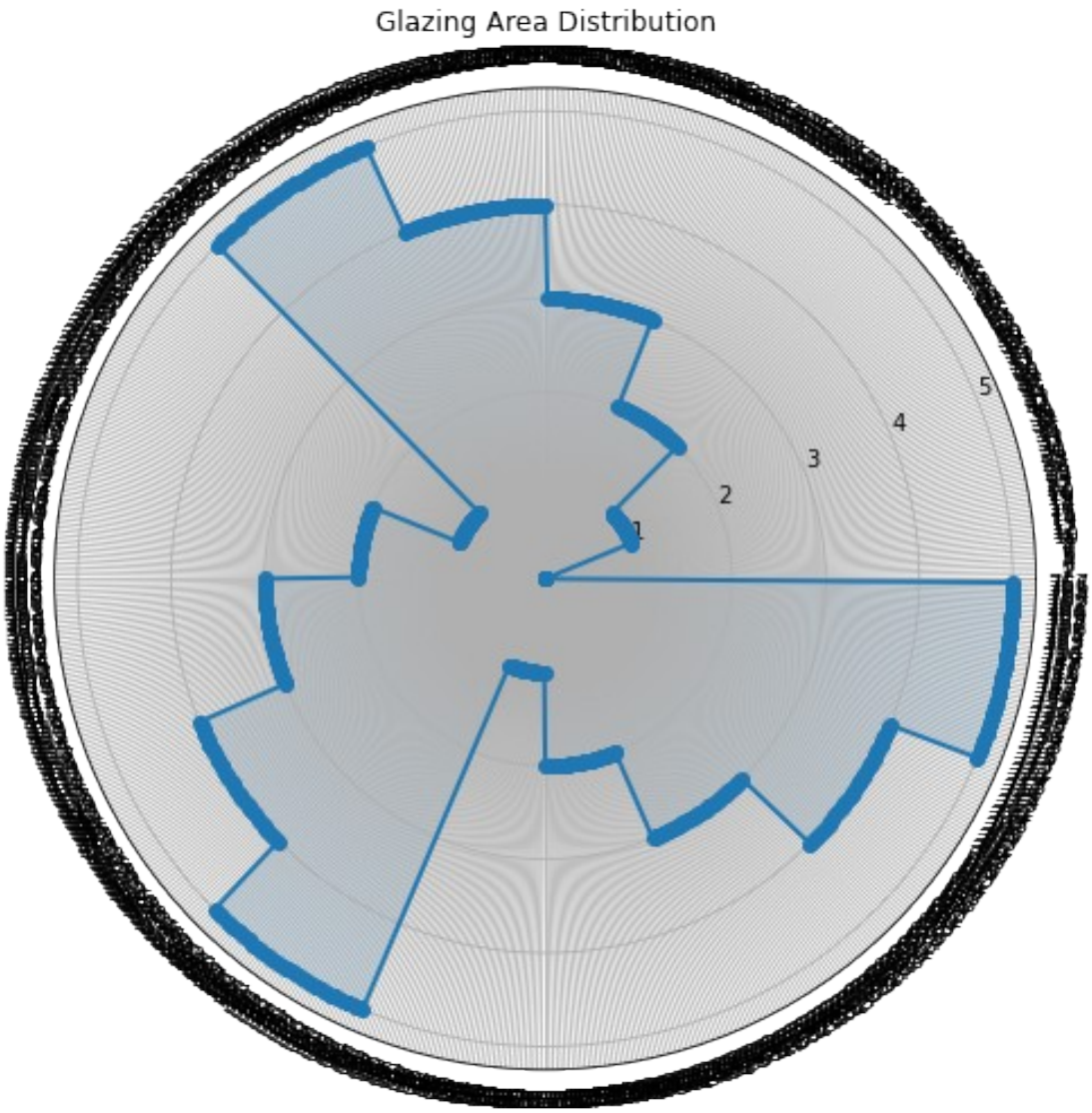

Wall Area
50
100
150
200
250
350
400

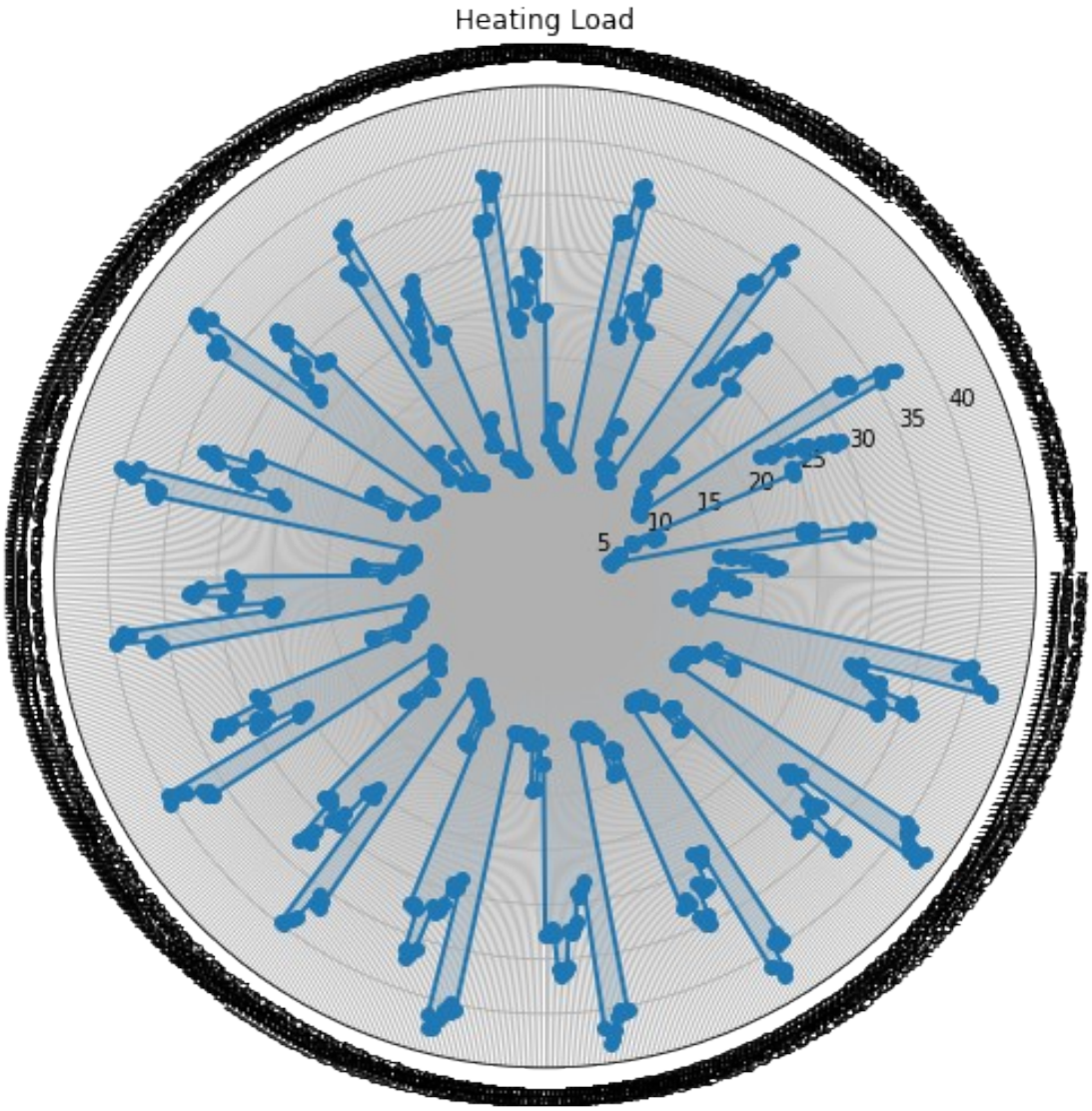

Roof Area
200
150
50

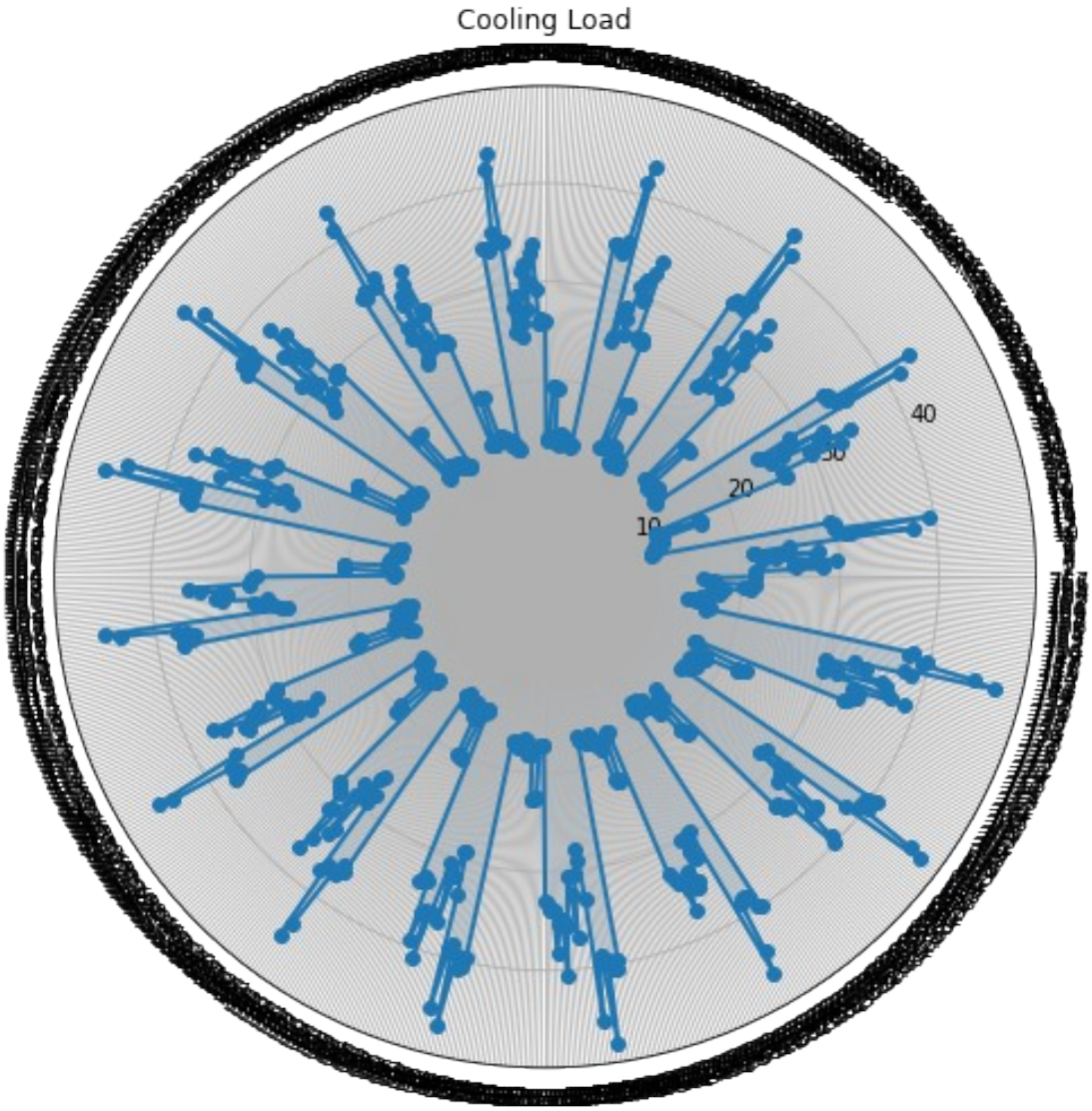

Overall Heigt
1
2
3
4
5
6
7


Orientation
1


Glazing Area
0.05
0.10
0.15
0.20
0.25
0.30
0.35

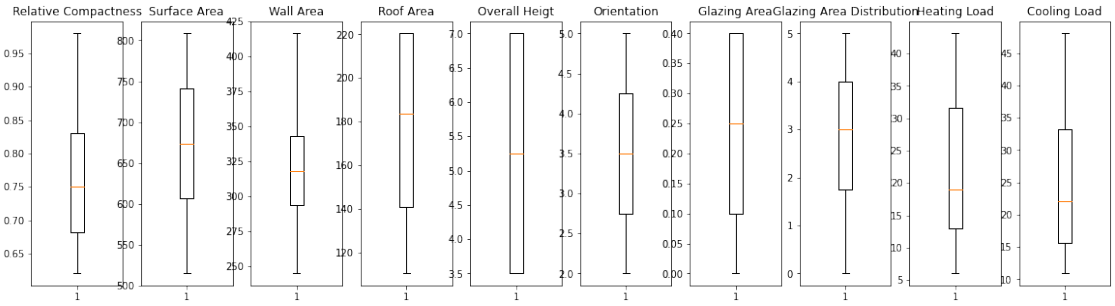

Glazing Area Distribution
1
2
3
4
5


Heating Load
5
10
15
20
25
30
35
40



BOX PLOT

```
# Load data from CSV file
data = pd.read_csv('dataset.csv')

# Create box plot for each attribute in data
fig, axes = plt.subplots(nrows=1, ncols=len(data.columns),
figsize=(20, 5))
for i, attribute in enumerate(data.columns):
    axes[i].boxplot(data[attribute])
    axes[i].set_title(attribute)
plt.show()
```



The command pandas_profiling.ProfileReport(data) generates a report that provides a comprehensive summary of a pandas DataFrame.

The pandas_profiling library generates a HTML report that includes statistics such as the number of missing values, the range of values, the data type of each column, and correlation between columns. The report also includes histograms, scatterplots, and other visualizations that can be useful for exploring the data.

```
pandas_profiling.ProfileReport(data)
```


```
{"model_id":"8eaaf1a162b44828b46b518496824ed3","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"d22f5ee9c2924b53b9abe712d59a569b","version_major":2,"vers
ion_minor":0}
```

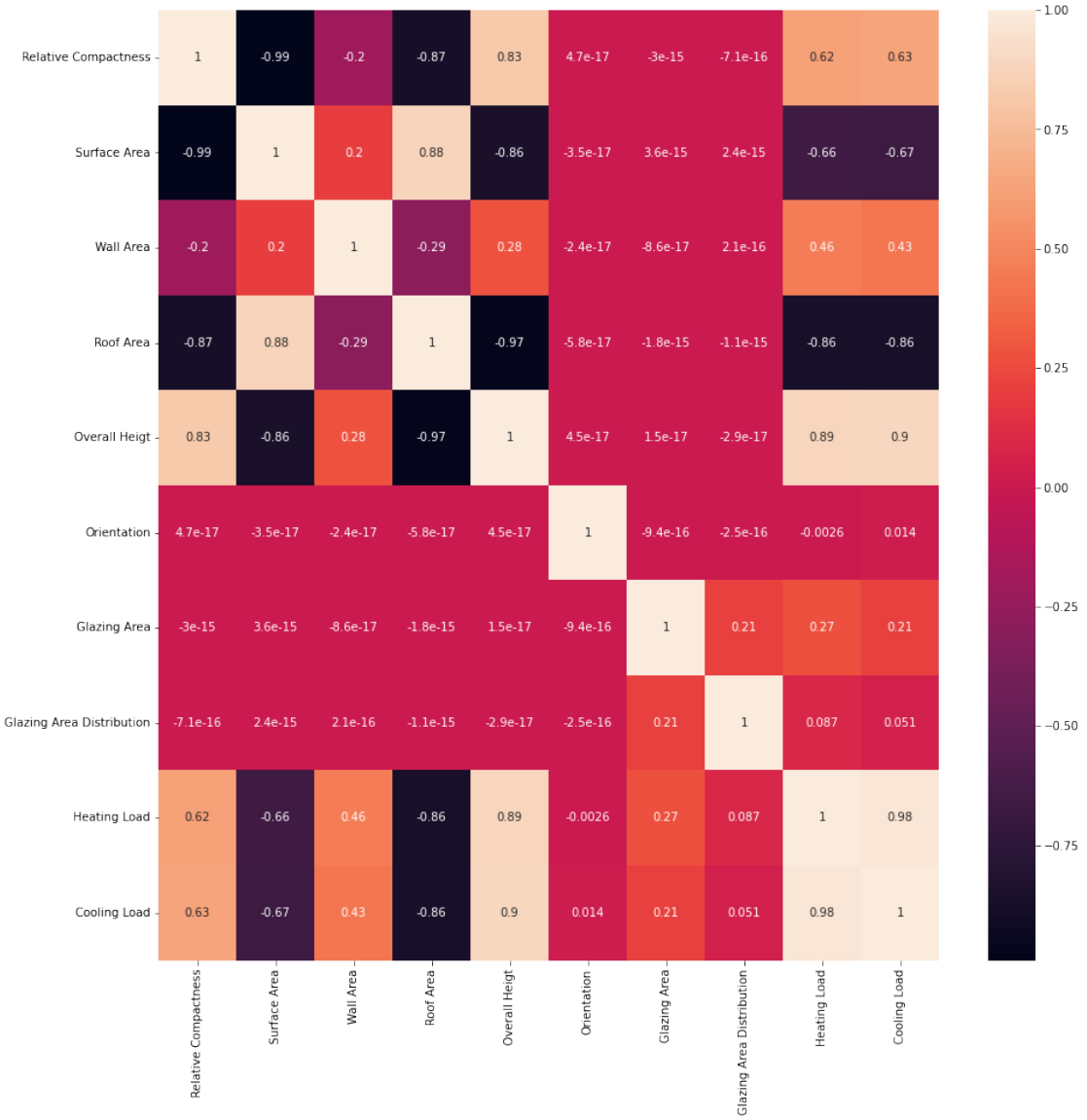
```
{"model_id":"233c76cf8698414bb1be95987b30bc56","version_major":2,"vers
ion_minor":0}
```


```
<IPython.core.display.HTML object>
```

PLOTTING HEATMAP

```
plt.figure(figsize=(15,15))
sns.heatmap(data.corr(), annot=True)
```

```
<Axes: >
```



This code has two parts, let me explain them one by one:

pd.set_option('display.float_format',lambda x: '{:,.2f}'.format(x) if abs(x) < 10000 else '{:,.0f}'.format(x)) This line of code is using the set_option method from the pandas library to set the formatting of float values that are displayed in the output. Specifically, it is setting the float format to display two decimal places if the absolute value of the float is less than 10,000, and to display no decimal places if the absolute value of the float is greater than or equal to 10,000. The lambda function is used to define the formatting behavior for each float value. It takes a single input (x), which is the float value being formatted, and applies the appropriate formatting rule based on the absolute value of x.

data.corr() This line of code is calling the corr() method on a pandas DataFrame named 'data'. The corr() method calculates the pairwise correlation between columns of a DataFrame, returning a new DataFrame of correlation coefficients. The resulting

DataFrame will have the same column and index labels as the original DataFrame, with correlation values ranging from -1 to 1.

```
pd.set_option('display.float_format',lambda x: '{:,.2f}'.format(x) if
abs(x) < 10000 else '{:,.0f}'.format(x))
data.corr()
```

```
                           Relative Compactness  Surface Area  Wall
Area  \
Relative Compactness                       1.00         -0.99      -
0.20
Surface Area                              -0.99          1.00
0.20
Wall Area                                 -0.20          0.20
1.00
Roof Area                                 -0.87          0.88      -
0.29
Overall Heigt                              0.83         -0.86
0.28
Orientation                                0.00         -0.00      -
0.00
Glazing Area                              -0.00          0.00      -
0.00
Glazing Area Distribution                 -0.00          0.00
0.00
Heating Load                               0.62         -0.66
0.46
Cooling Load                               0.63         -0.67
0.43

                           Roof Area  Overall Heigt  Orientation  \
Relative Compactness           -0.87           0.83         0.00
Surface Area                    0.88          -0.86        -0.00
Wall Area                      -0.29           0.28        -0.00
Roof Area                       1.00          -0.97        -0.00
Overall Heigt                  -0.97           1.00         0.00
Orientation                    -0.00           0.00         1.00
Glazing Area                   -0.00           0.00        -0.00
Glazing Area Distribution      -0.00          -0.00        -0.00
Heating Load                   -0.86           0.89        -0.00
Cooling Load                   -0.86           0.90         0.01

                           Glazing Area  Glazing Area Distribution  \
Relative Compactness              -0.00                      -0.00
Surface Area                       0.00                       0.00
Wall Area                         -0.00                       0.00
Roof Area                         -0.00                      -0.00
Overall Heigt                      0.00                      -0.00
Orientation                       -0.00                      -0.00
Glazing Area                       1.00                       0.21
```

Glazing Area Distribution	0.21	1.00
Heating Load	0.27	0.09
Cooling Load	0.21	0.05

	Heating Load	Cooling Load
Relative Compactness	0.62	0.63
Surface Area	-0.66	-0.67
Wall Area	0.46	0.43
Roof Area	-0.86	-0.86
Overall Heigt	0.89	0.90
Orientation	-0.00	0.01
Glazing Area	0.27	0.21
Glazing Area Distribution	0.09	0.05
Heating Load	1.00	0.98
Cooling Load	0.98	1.00