VJC x TP Hackathon

Group 2 from Xinmin Secondary School

Our Team

Andrea, Gareth, Javier

Problem Statement

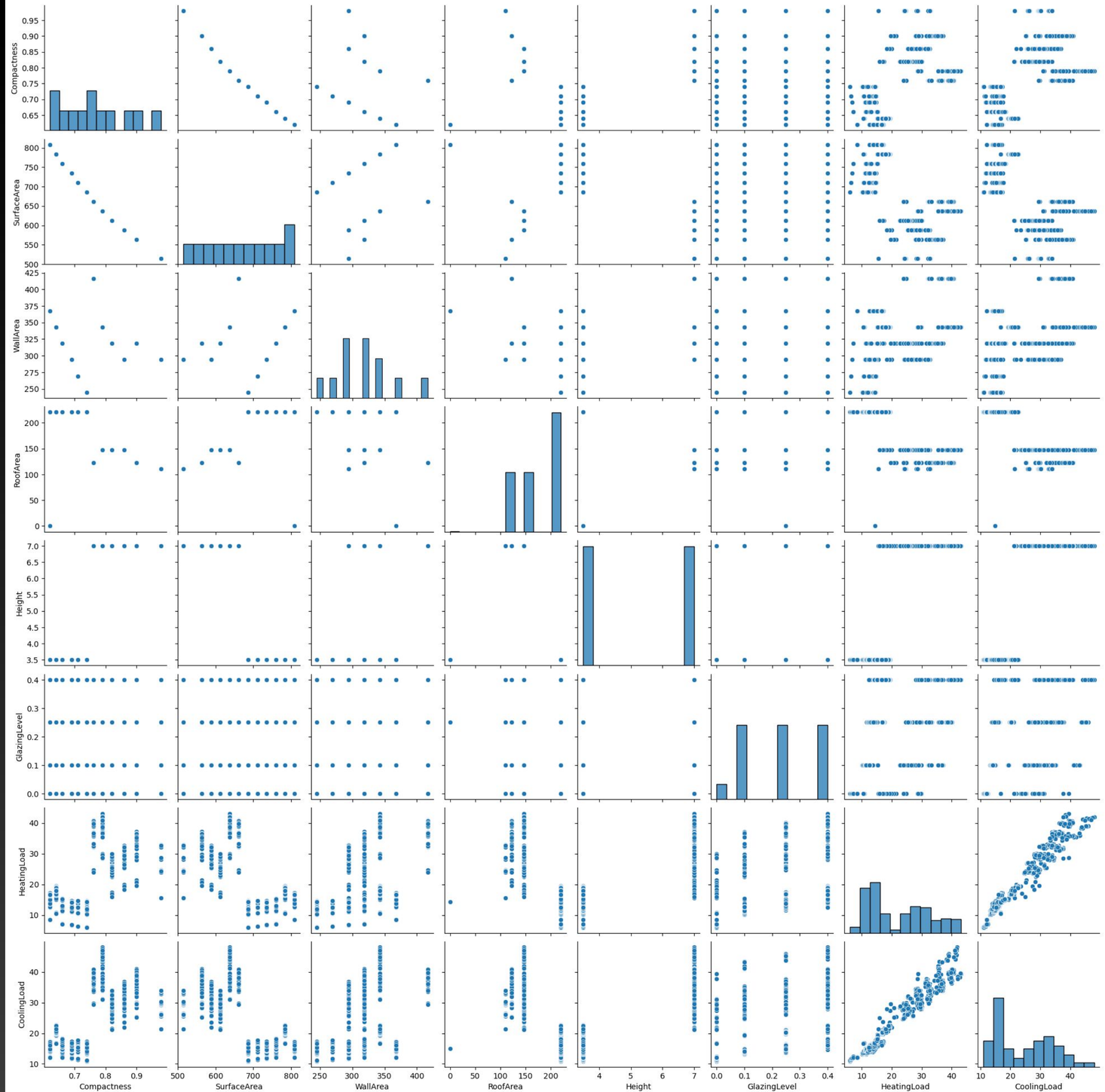
Using the provided dataset, create a data analysis and visualization pipeline to explore the factors influencing energy efficiency in residential buildings. Additionally, build a predictive model to estimate the energy efficiency of a house based on its features.

Data Set

Our data set shows 769 different houses and their compactness, surface area, wall area, roof area, height, orientation, glazing level, glazing area distribution, heating and cooling load.

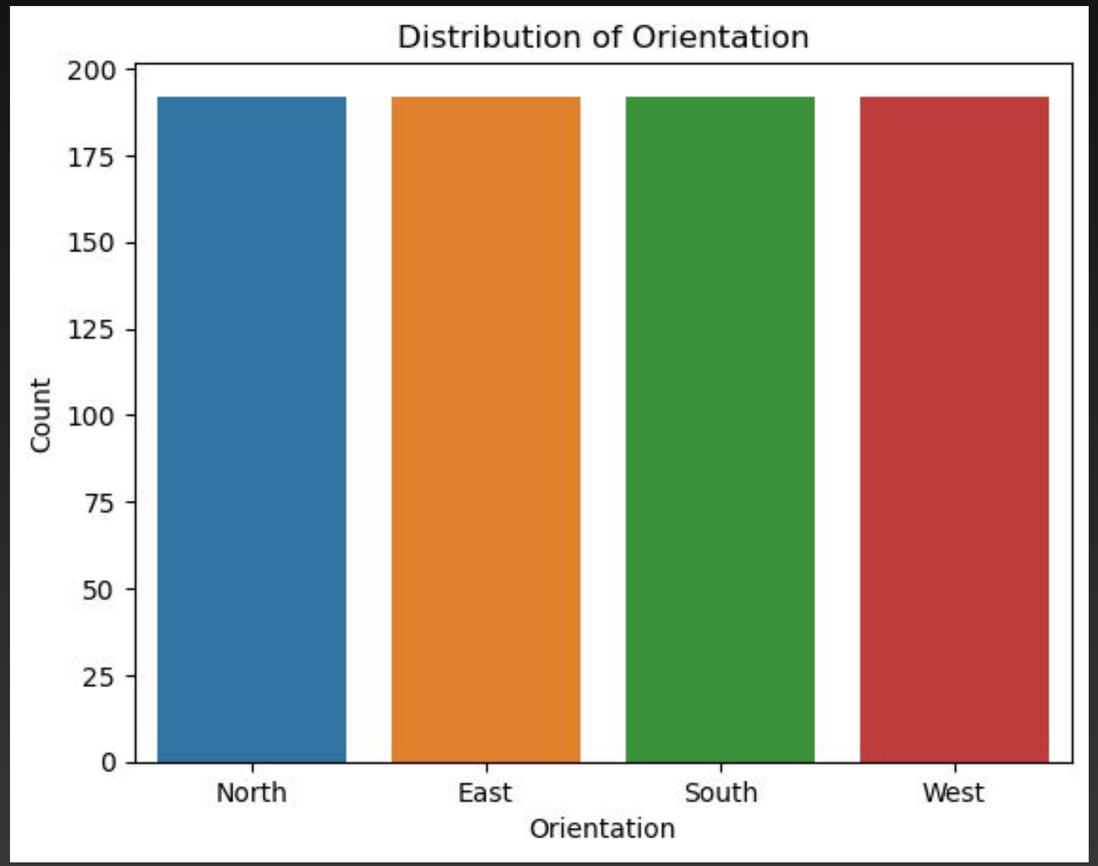
Possible Graphs

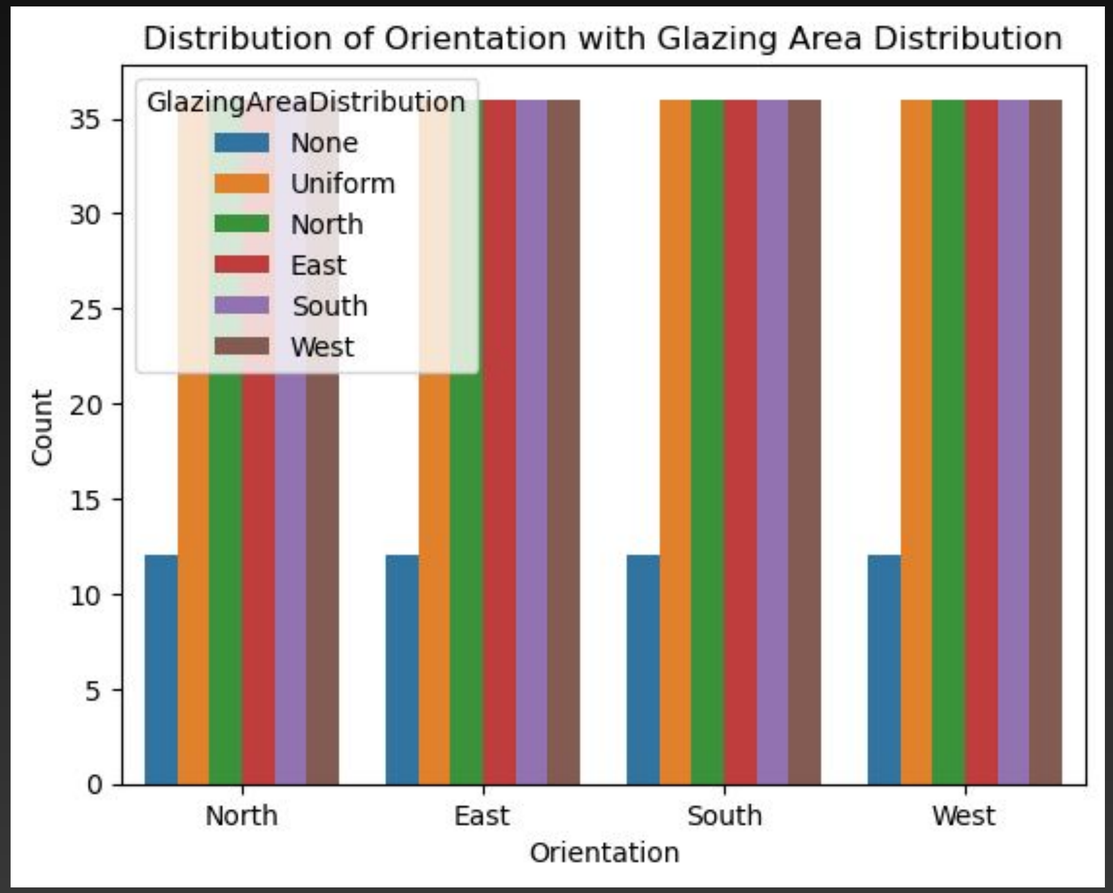
There are 64 possible graphs, but not all of them are useful



Reliability of data

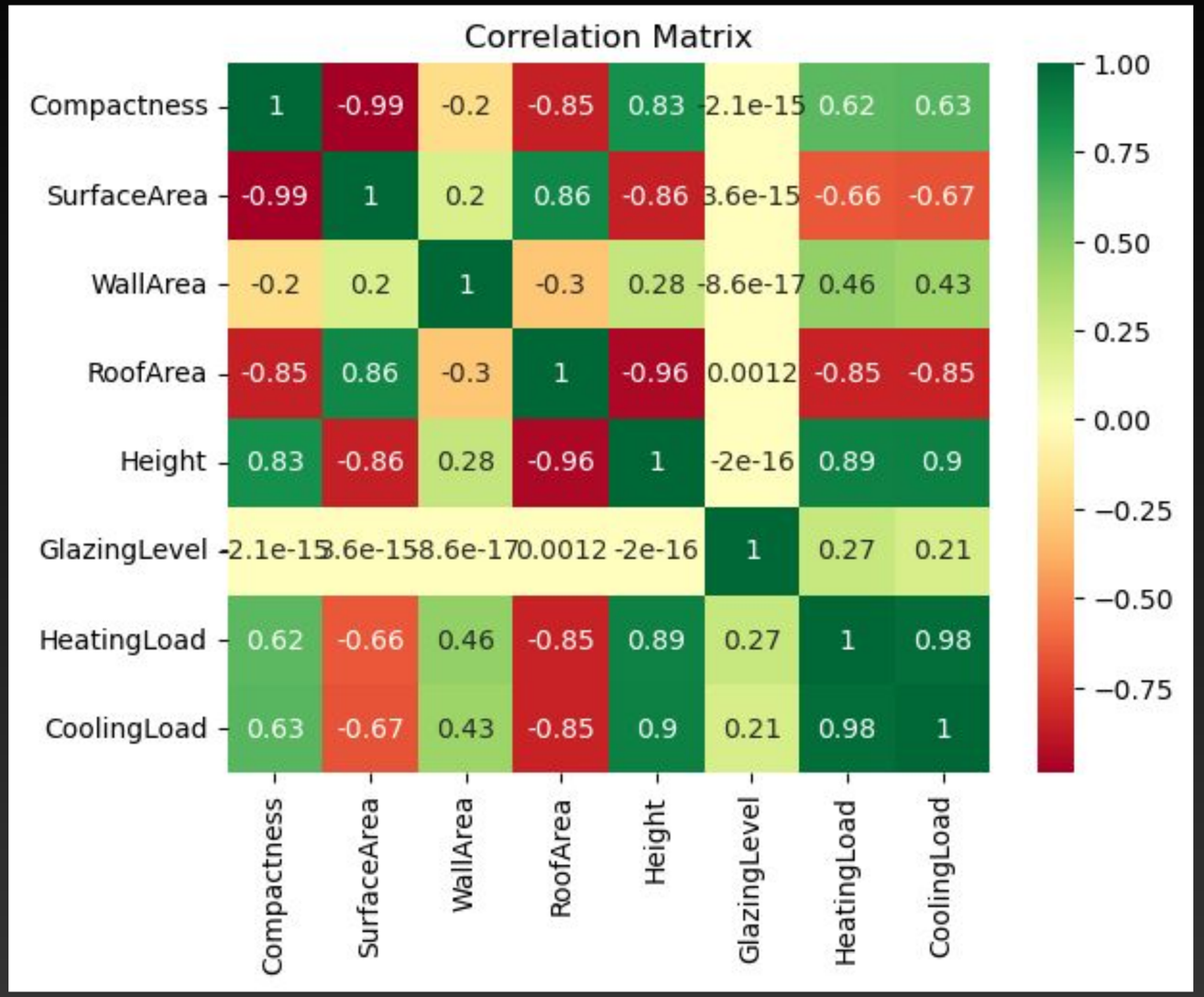
This data is not useful as there are equal number of occurrences for each variable.





Correlation Matrix

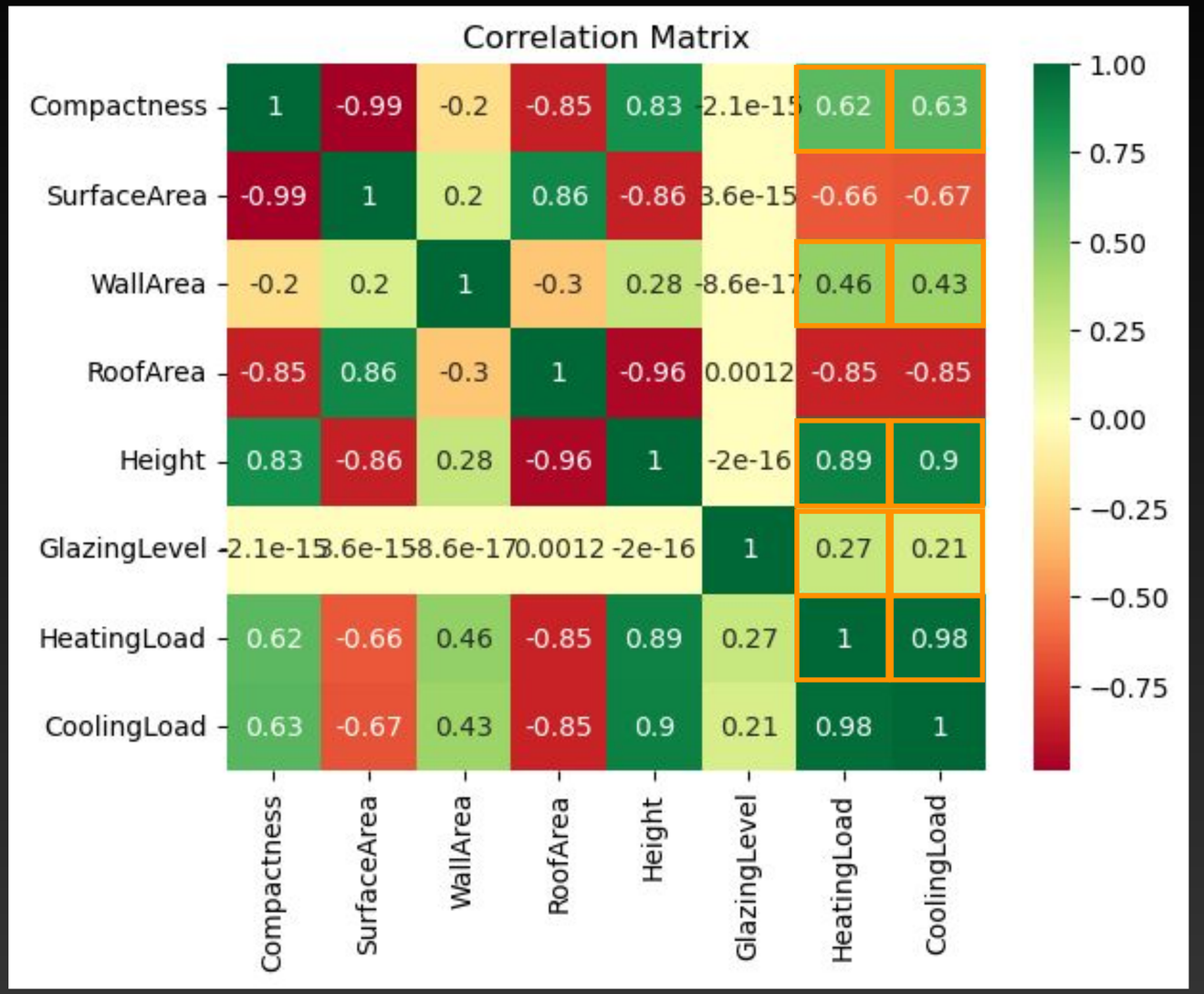
A Correlation Matrix provides a numerical measure of the strength and direction of the linear relationship between pairs of variables. By examining the correlation coefficients in the matrix, you can identify variables that have a strong positive or negative correlation, indicating that they may be good candidates for plotting against each other.



Correlation Matrix of 8 variables to see how related each variable is to each other

Correlation Matrix

We want to compare how the architectural features affect heating and cooling load. Therefore, the values boxed up are those that we can compare.

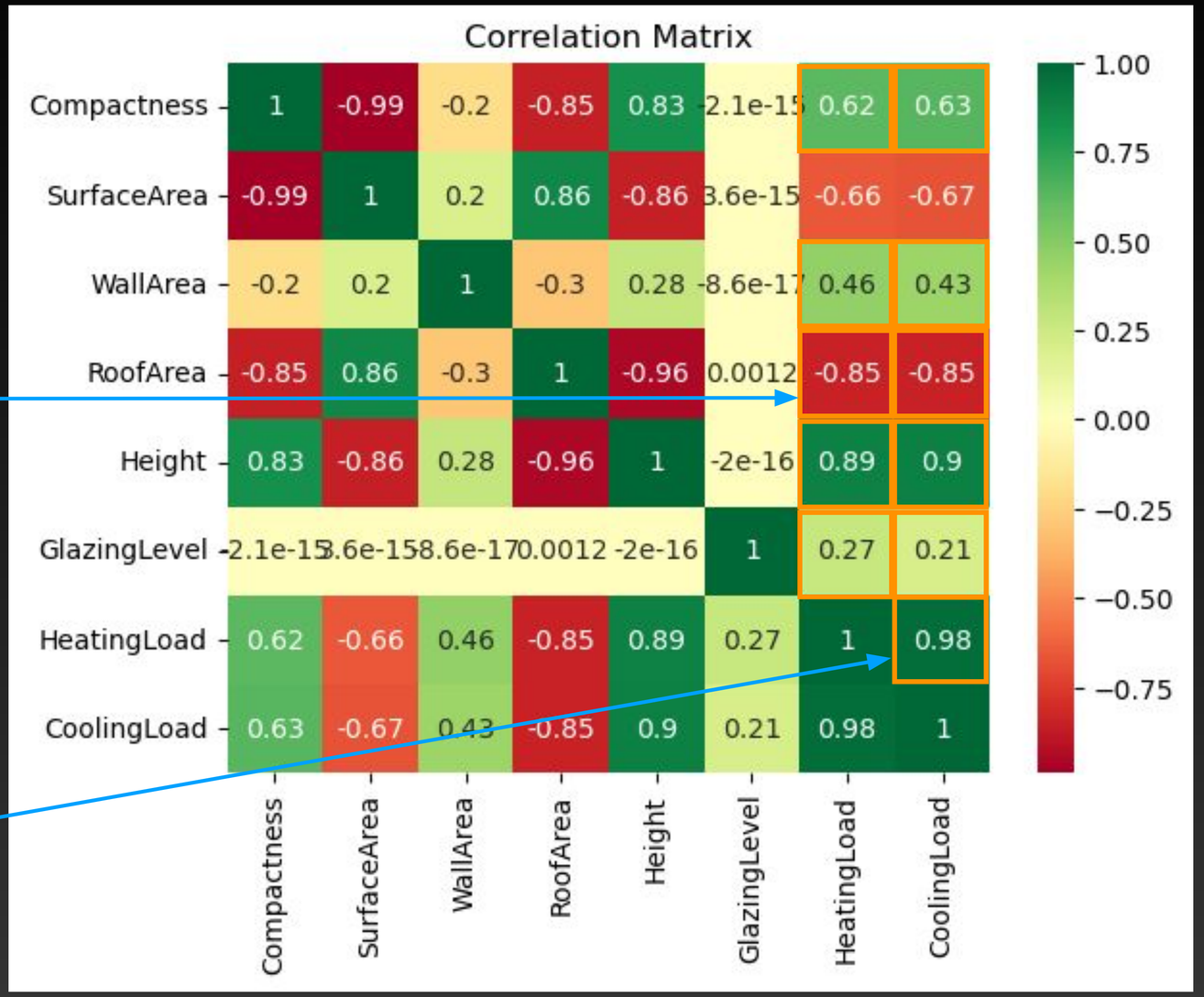


Correlation Matrix of 8 variables to see how related each variable is to each other

Correlation Matrix

Smaller roof area will increase in lower heating and cooling load

Housing and cooling load are as high in a housing estate



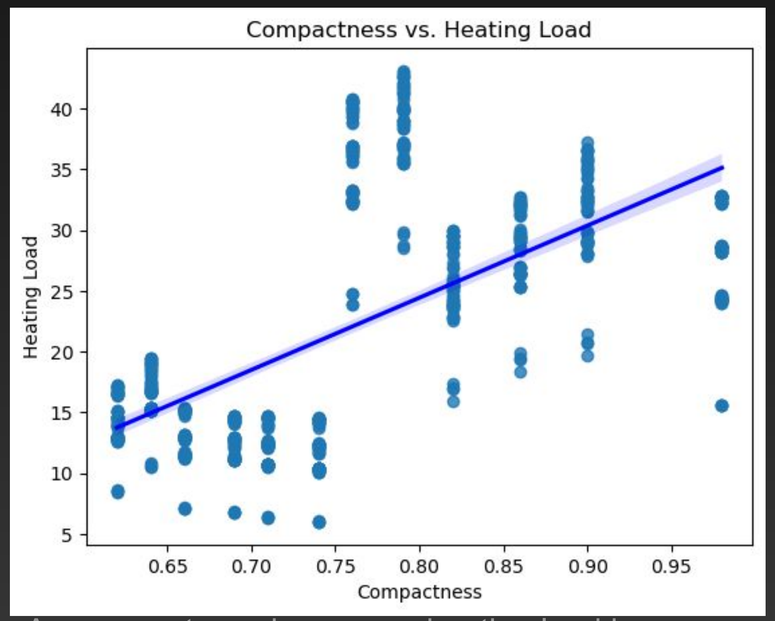
Correlation Matrix of 8 variables to see how related each variable is to each other

Scatter Graphs

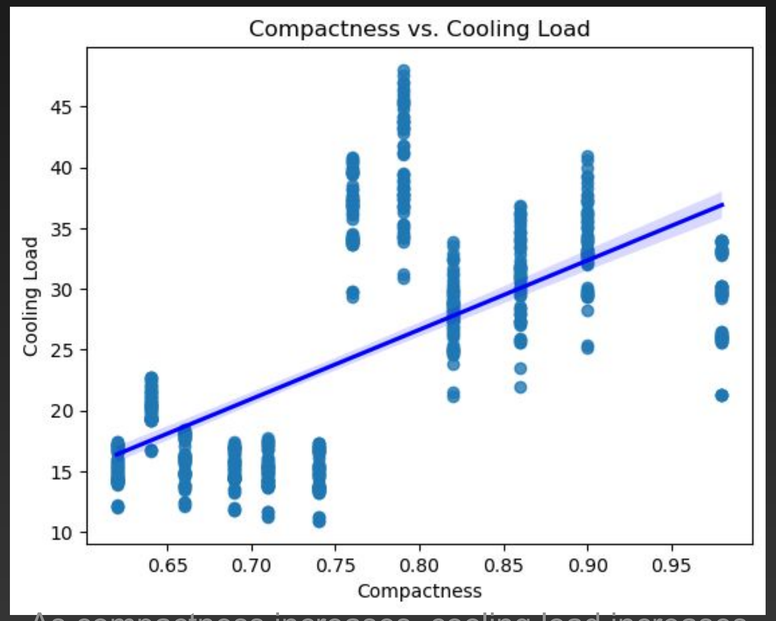
We can use scatter graphs to find the trends of a set of data. This can be achieved through a best fit line, to observe whether the trend is going up or down. In this case, we use architectural features and plot them against the heating and cooling load.

Scatter Graphs

We can use scatter graphs to find the trends of a set of data



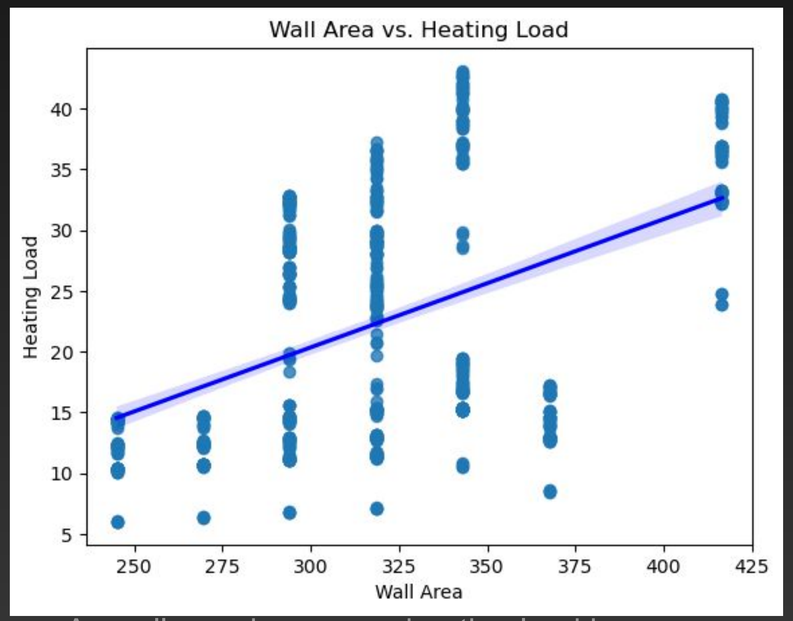
As compactness increases, heating load increases



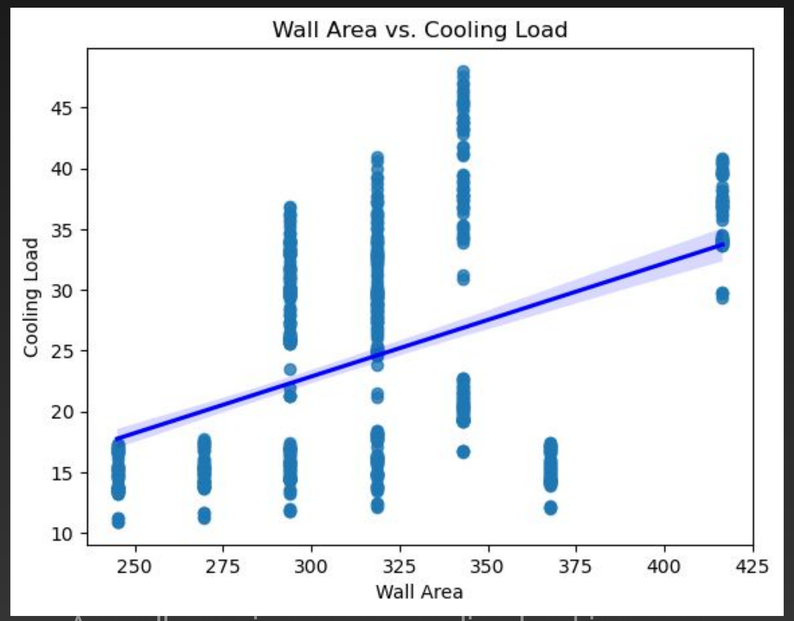
As compactness increases, cooling load increases

Scatter Graphs

We can use scatter graphs to find the trends of a set of data



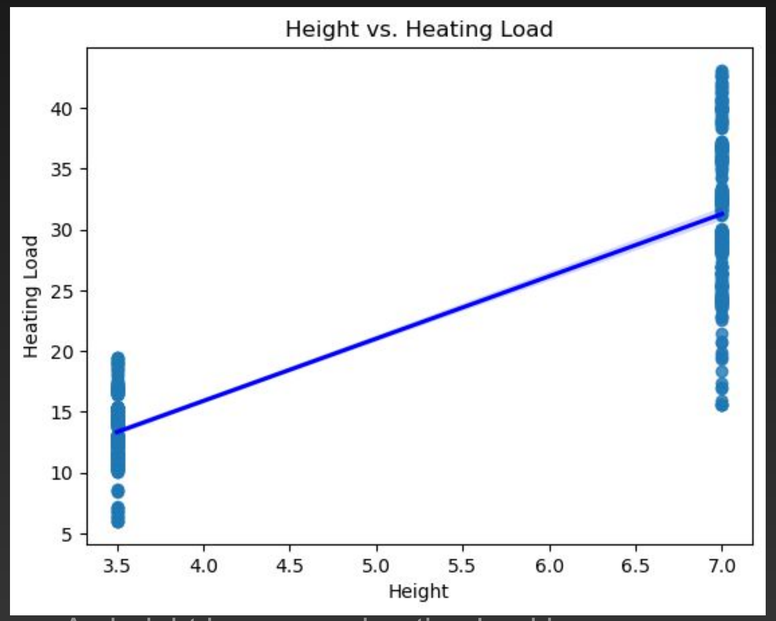
As wall area increases, heating load increases



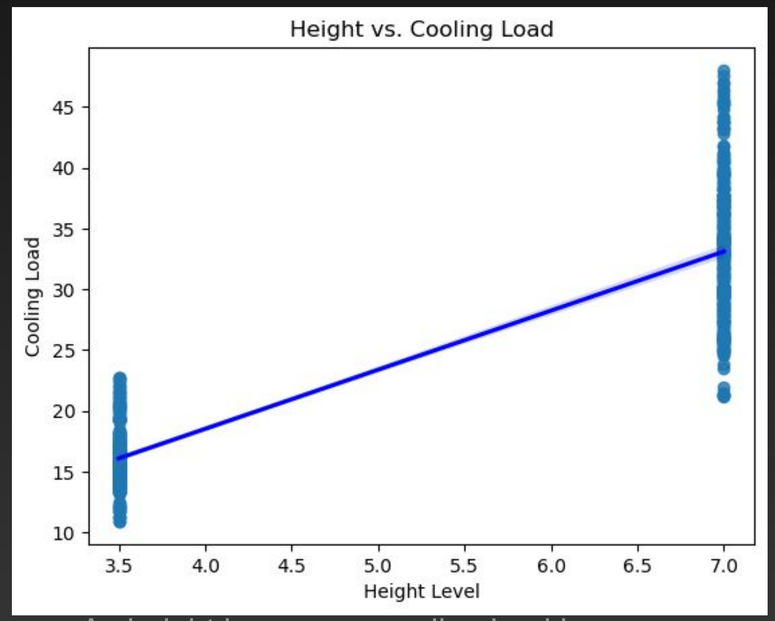
As wall area increases, cooling load increases

Scatter Graphs

We can use scatter graphs to find the trends of a set of data



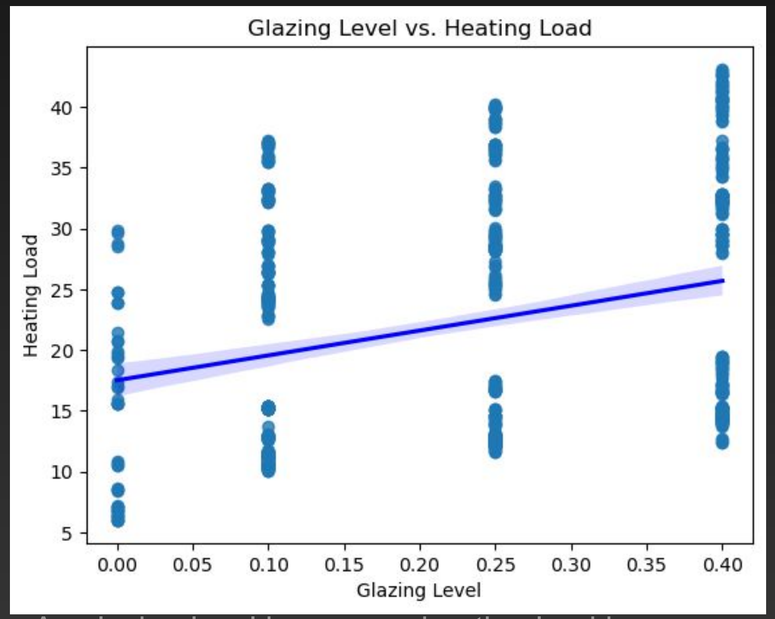
As height increases, heating load increases



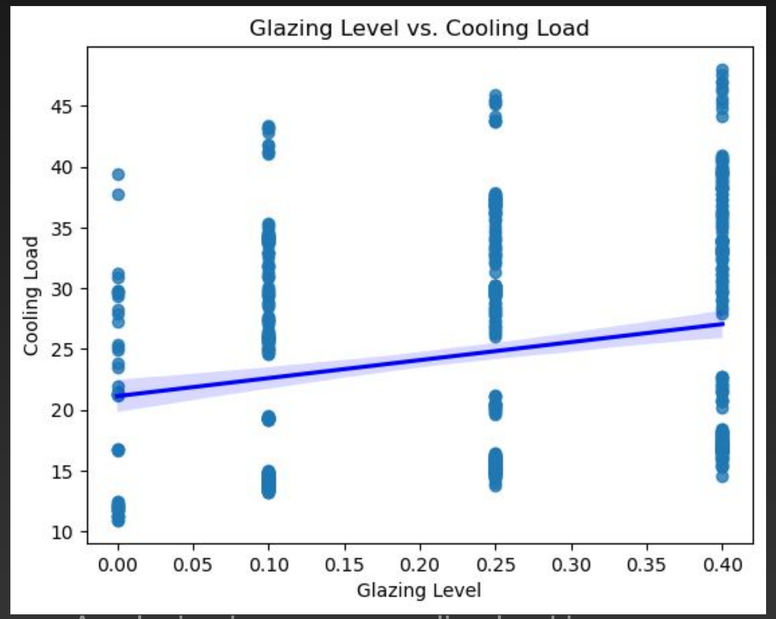
As height increases, cooling load increases

Scatter Graphs

We can use scatter graphs to find the trends of a set of data



As glazing level increases, heating load increases



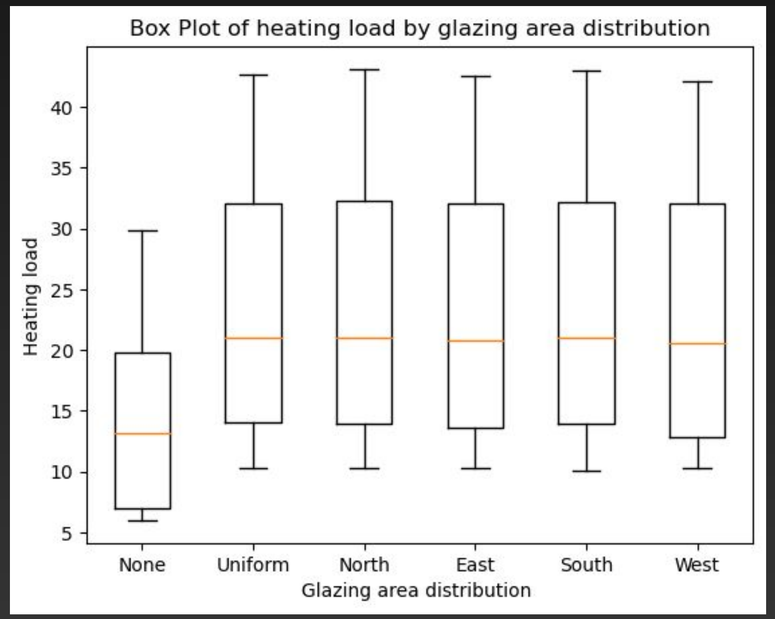
As glazing increases, cooling load increases

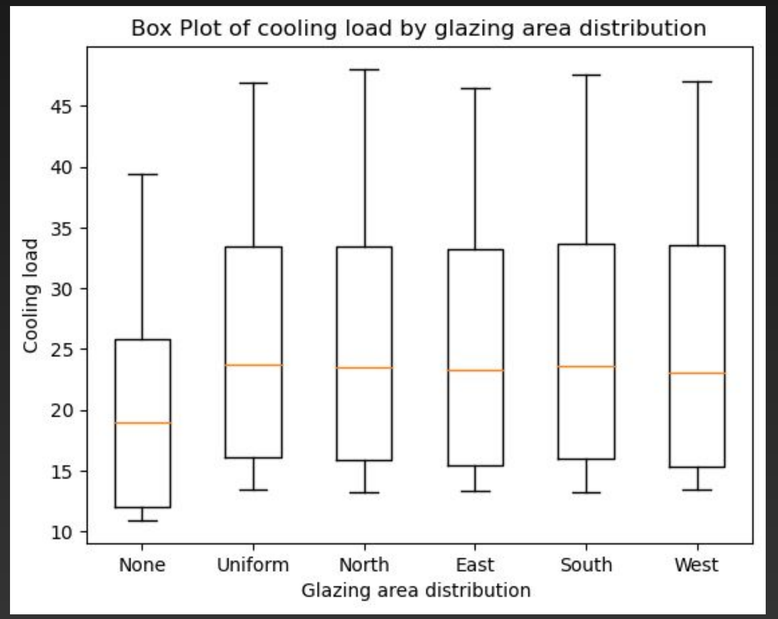
Box Plots

We can use box plots as a standardised way of displaying the distribution of data on a five number summary. It can tell us about our outliers and what their values are.

Box Plots

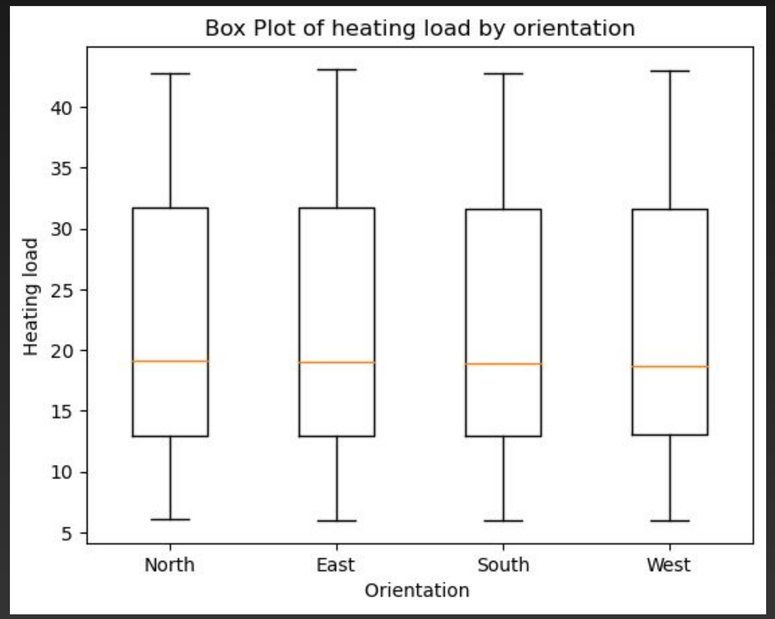
We can use box plots to summarise data

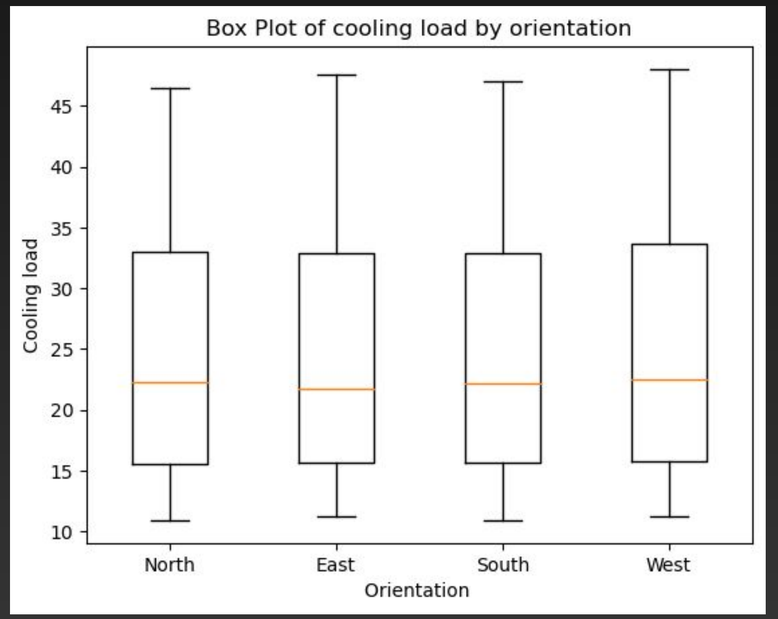




Box Plots

We can use box plots to summarise data





Inference From Graphs

From the scatter plots and box plots, we observe a trend that as architectural features increase in value, the higher the heating and cooling load. To achieve energy efficiency, we have to reduce these factors.

Exploratory Data Analysis

Mean	
Compactness	0.764167
SurfaceArea	671.708333
WallArea	318.500000
RoofArea	176.387223
Height	5.250000
GlazingLevel	0.234375
HeatingLoad	22.307201
CoolingLoad	24.587760

Median	
Compactness	0.75
SurfaceArea	673.75
WallArea	318.50
RoofArea	147.00
Height	5.25
GlazingLevel	0.25
HeatingLoad	18.95
CoolingLoad	22.08

Std. Deviation	
Compactness	0.10577
SurfaceArea	88.086
WallArea	43.626
RoofArea	45.573
Height	1.7511
GlazingLevel	0.13322
HeatingLoad	10.090
CoolingLoad	9.513

Exploratory Data Analysis

Head

	Compactness	SurfaceArea	WallArea	RoofArea	Height	Orientation	GlazingLevel	GlazingAreaDistribution	HeatingLoad	CoolingLoad
0	0.98	514.5	294.0	110.25	7.0	North	0.0	None	15.55	21.33
1	0.98	514.5	294.0	110.25	7.0	East	0.0	None	15.55	21.33
2	0.98	514.5	294.0	110.25	7.0	South	0.0	None	15.55	21.33
3	0.98	514.5	294.0	110.25	7.0	West	0.0	None	15.55	21.33
4	0.90	563.5	318.5	122.50	7.0	North	0.0	None	20.84	28.28

Tail

	Compactness	SurfaceArea	WallArea	RoofArea	Height	Orientation	GlazingLevel	GlazingAreaDistribution	HeatingLoad	CoolingLoad
763	0.64	784.0	343.0	220.5	3.5	West	0.4	West	17.88	21.40
764	0.62	808.5	367.5	220.5	3.5	North	0.4	West	16.54	16.88
765	0.62	808.5	367.5	220.5	3.5	East	0.4	West	16.44	17.11
766	0.62	808.5	367.5	220.5	3.5	South	0.4	West	16.48	16.61
767	0.62	808.5	367.5	220.5	3.5	West	0.4	West	16.64	16.03

Exploratory Data Analysis

Upon closer inspection, we see that values are somewhat sorted because the rows of same values are put together. This can be an issue when a linear regression model is trained on this data, because the model now can predict the output based on order.

Exploratory Data Analysis

Another error there is that columns like ‘GlazingAreaDistribution’ or ‘Orientation’ have same number of occurrences for every value, only exception being ‘None’

GlazingAreaDistribution	
Uniform	144
North	144
East	144
South	144
West	144
None	48

Orientation	
North	192
East	192
South	192
West	192

Data Problem Handling

To solve the first problem, we use Scikit-learn's utility method called 'shuffle'. As the name suggests, the method shuffles the dataframe, making it less predictable by a linear regression model.

	Compactness	SurfaceArea	WallArea	RoofArea	Height	Orientation	GlazingLevel	GlazingAreaDistribution	HeatingLoad	CoolingLoad
566	0.66	759.5	318.5	220.50	3.5	South	0.40	Uniform	15.23	18.14
213	0.76	661.5	416.5	122.50	7.0	East	0.10	South	32.38	33.62
389	0.90	563.5	318.5	122.50	7.0	East	0.25	East	32.40	35.10
2	0.98	514.5	294.0	110.25	7.0	South	0.00	None	15.55	21.33
571	0.64	784.0	343.0	220.50	3.5	West	0.40	Uniform	19.42	22.53
...	...	...	...	...	...	...	...	...	...	...
695	0.76	661.5	416.5	122.50	7.0	West	0.40	South	40.11	40.77
613	0.66	759.5	318.5	220.50	3.5	East	0.40	North	15.29	17.89
325	0.66	759.5	318.5	220.50	3.5	East	0.25	Uniform	13.18	16.27
748	0.71	710.5	269.5	220.50	3.5	North	0.40	West	12.43	15.59
68	0.76	661.5	416.5	122.50	7.0	North	0.10	Uniform	32.96	33.87

Shuffled order of data frame

Data Problem Handling

As for values having the same number of occurrences, there is no fix as of what we know. That said, the person making the dataset should take into consideration factors like occurrences of values.

Linear Regression Model

Our multiple linear regression model was trained and built upon the variables to predict the energy efficiency of a residential building. We then used the data gathered from scatter graphs to decide what values to put into the model to predict. This step can be used to prove that our analysis of the data was indeed correct. We are going to go through the code now. I have written comments on the code so it can be understood better.

Linear Regression Model

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.metrics import r2_score, mean_squared_error

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

Linear Regression Model

```
# Map string values as an integer representation
label_encoder = LabelEncoder()
data['GlazingAreaDistribution'] = label_encoder.fit_transform(data['GlazingAreaDistribution'])

# One-hot encoding
column_transformer = ColumnTransformer([('encoder', OneHotEncoder(), [5])], remainder='passthrough')
data = column_transformer.fit_transform(data)

# Split data into X and y variables. Remove target variables from input variables
X = data[:, :-2]
y_heat = data[:, -2]
y_cool = data[:, -1]

# Handle missing values in X
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)
```

imputer is actually the same as fillna, except it is from the sklearn library, while fillna is from pandas library. It is better to use sklearn's utility when pandas is not involved.

Linear Regression Model

```
# Linear regression models
regressor_heat = LinearRegression()
regressor_cool = LinearRegression()

# Training models
regressor_heat.fit(X, y_heat)
regressor_cool.fit(X, y_cool)
```

```
# Dummy data to be predicted
dummy_data = {
    'Compactness': [0.9],
    'SurfaceArea': [550.0],
    'WallArea': [300.0],
    'RoofArea': [120.0],
    'Height': [6.5],
    'Orientation': ['South'],
    'GlazingLevel': [0.25],
    'GlazingAreaDistribution': ['South'],
    'HeatingLoad': [0.0],
    'CoolingLoad': [0.0]
}

# Make df for dummy data
dummy_df = pd.DataFrame(dummy_data)
```

Linear Regression Model

```
# Map strings values as an integer representation
dummy_df['GlazingAreaDistribution'] = label_encoder.transform(dummy_df['GlazingAreaDistribution'])

# One-hot encoding
dummy_data_transformed = column_transformer.transform(dummy_df)[:, :-2]  # Exclude heating and cooling
load

# Predictions
dummy_pred_heat = regressor_heat.predict(dummy_data_transformed)
dummy_pred_cool = regressor_cool.predict(dummy_data_transformed)
```

Linear Regression Model

```
# Get r^2 and mean squared err (mse) values
r2_heat = r2_score(y_heat, regressor_heat.predict(X))
mse_heat = mean_squared_error(y_heat, regressor_heat.predict(X))
r2_cool = r2_score(y_cool, regressor_cool.predict(X))
mse_cool = mean_squared_error(y_cool, regressor_cool.predict(X))

# Print results
print('Predicted Heating Load:', dummy_pred_heat)
print('Predicted Cooling Load:', dummy_pred_cool)
print('R-squared value (Heating Load):', r2_heat)
print('Mean Squared Error (Heating Load):', mse_heat)
print('R-squared value (Cooling Load):', r2_cool)
print('Mean Squared Error (Cooling Load):', mse_cool)
```

Linear Regression Model

When we plugged in the following values:

Data	
Compactness	0.62
Surface Area	808.5
WallArea	376.5
RoofArea	220.5
Height	3.5
Orientation	East
GlazingLevel	0.1
GlazingAreaDistribution	North

Predicted Heating Load: 12.66956881
Predicted Cooling Load: 15.13694088
R^2 value (Heating Load): 0.9158819989985129
MSE (Heating Load): 8.553074780220324
R^2 value (Cooling Load): 0.8885026832391478
MSE (Cooling Load): 10.077700618142643

Legend:

R^2 — R Squared

R^2 is the measure that provides information about the goodness of fit of a model.

MSE — Mean Squared Error

MSE is a measure of the quality of an estimator.

Red — Heating Load

Green — Cooling Load

Linear Regression Model

Predicted Heating Load: 12.66956881
Predicted Cooling Load: 15.13694088

From our EDA, we know the mean for heating load and cooling load are 22.307201 and 24.587760. When we plugged in data that we think could produce a small heating and cooling load, the model predicted it to be 12.66956881 and 15.13694088 respectively. This proves that our analysis and predictions were accurate.

Mean	
Mean:	
Compactness	0.764167
SurfaceArea	671.708333
WallArea	318.500000
RoofArea	176.387223
Height	5.250000
GlazingLevel	0.234375
HeatingLoad	22.307201
CoolingLoad	24.587760

Key features

Earlier, from our scatter graphs and box plots, we observed a trend that as the values of architectural features increase, the higher the heating and cooling load. This can be proved with the multiple linear regression model.

Key features

From our data, we can infer that height, compactness, glazing level and surface area affect energy efficiency the most. We recommend that potential home buyers should look out for these factors when finding energy efficient homes.

Key features

With the specific architectural features in mind, we asked ChatGPT to find some places in Singapore that suit our descriptions

ChatGPT, find housing estates in Singapore that suits the following:
Height: 3.5m
Orientation: East
Surface Area: Below 800㎡
Glazing Level: 0.1
Glazing Area Distribution: North

Key features

What ChatGPT suggested:

Bedok Estates

Bedok offers HDB flats with a height of around 3.5m, an eastern orientation, and a surface area below 800m². HDB flats in Bedok typically have windows on multiple sides, including the north side, which aligns with the desired glazing area distribution.

Key features

What ChatGPT suggested:

Tampines Estates

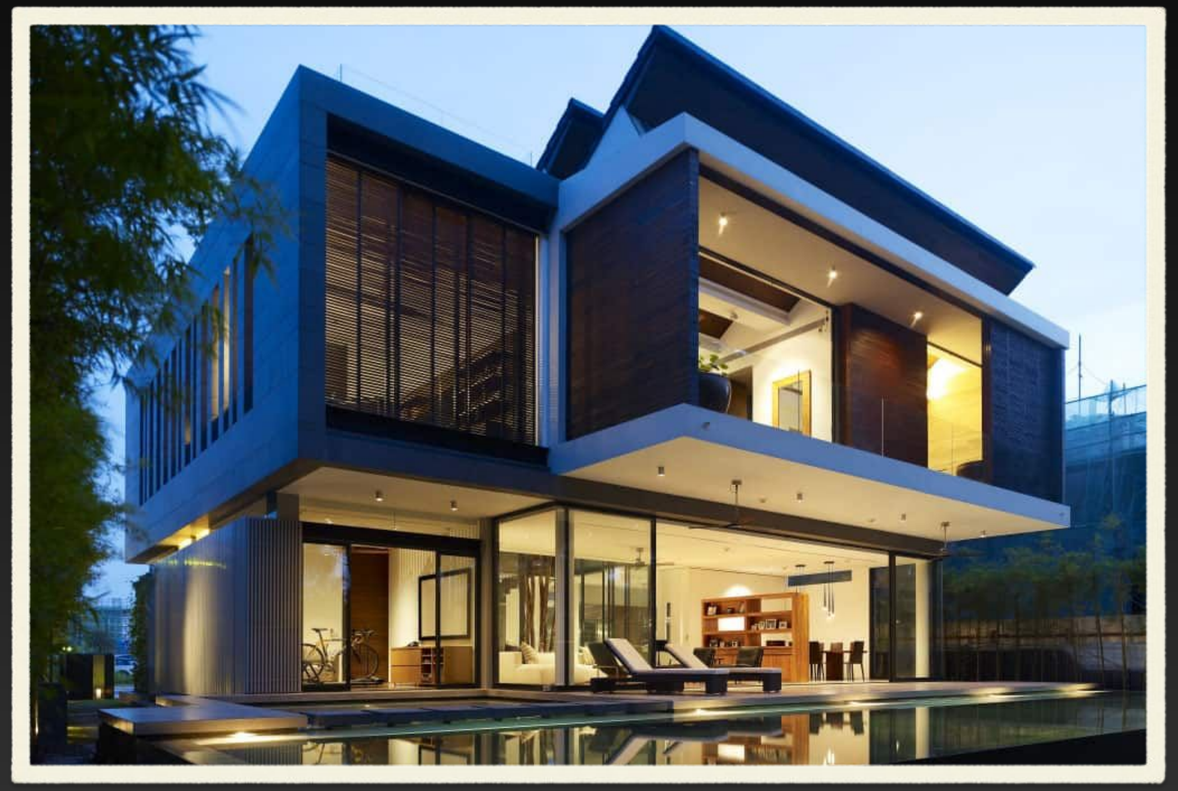
Tampines is another residential area in the eastern part of Singapore that offers a variety of housing options, including HDB flats. Similar to Bedok, Tampines has HDB flats with heights around 3.5 meters and an eastern orientation. You can find HDB flats with a surface area below 800m² and some units with significant glazing on the north side.

Key features

What ChatGPT suggested:

Sentosa Cove

It is located on Sentosa Island, Sentosa Cove is a luxurious waterfront residential area. It comprises primarily of high-end bungalows, villas, and houses. While the properties here are generally larger, it's worth exploring listings to see if any meet your specifications.

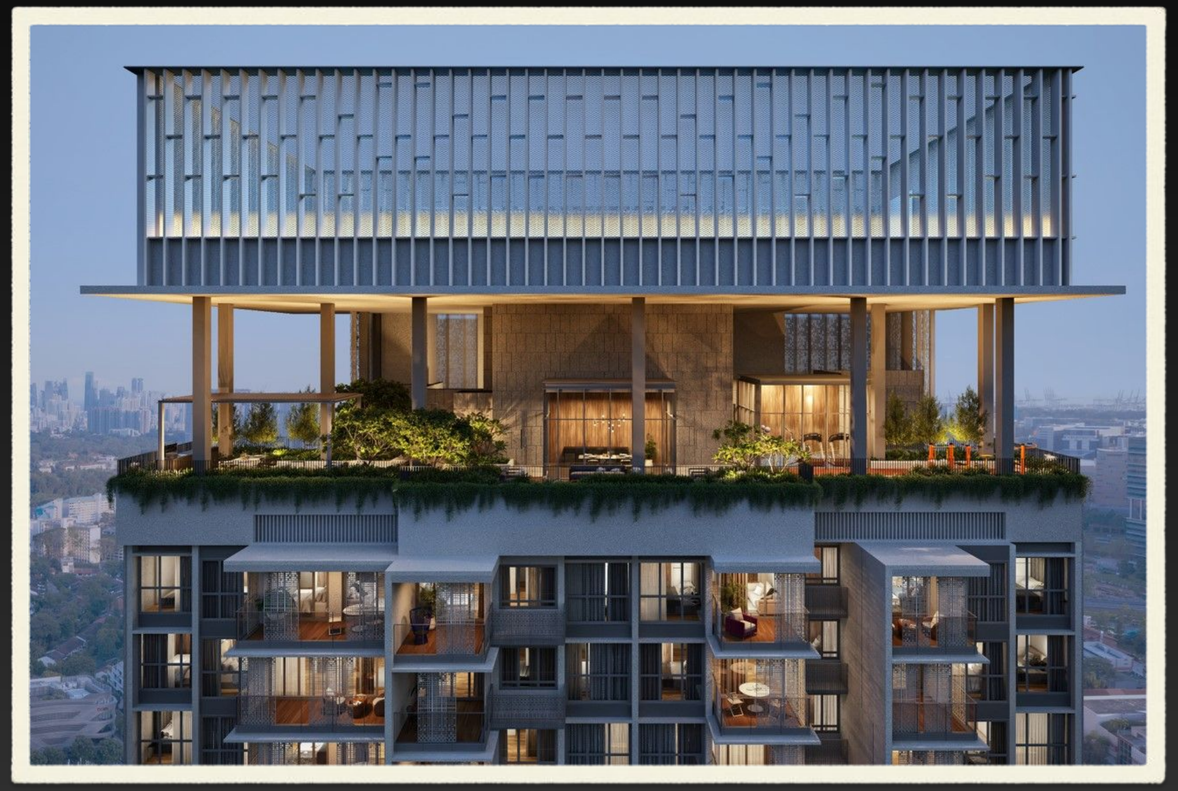


Key features

What ChatGPT suggested:

Holland Road

It is situated near the vibrant Holland Village, this area is known for its landed properties. Some houses in this neighbourhood may offer suitable heights, orientations, and surface areas.



Key features

In Conclusion

Height of the house should be less than 3m for HDB or condo, less than 7m for landed property.

Relative compactness of the house (0 to 1) should be around 0.6.

Glazing area expressed in percentage of floor area should be lower, around 0.3.

Surface area of the house (m^2) should be around 800.

Thank you for listening

To three misfits present their stupid idea

VJC x TP Hackathon

Group 2 from Xinmin Secondary School