D-214

Task-2

Data Analytics Report and Executive Summary (Attempt 2)

20th February 2022

Presented By : Harshal Parikh
Student ID: 007091749
Program : Masters in Data Analytics
Program Mentor : Jared Knepper
Contact : *hparikh@wgu.edu*

`

Section A: Research Question

A.1 Justification of Research Question

"Could we identify the principal components that have the greatest impact on the Building Energy Efficiency?" was the research question identified in Task-1. Since we have 8 different components that potentially impact the Energy Efficiency of a building it is crucial to identify which component/feature impacts the Energy Efficiency of the building the most.

A.2 Description of Context

Energy Efficient Buildings are defined as buildings that are designed to reduce the energy consumption required for heating and cooling, irrespective of the energy and of the components that are chosen to heat or cool the building. This could be achieved through bioclimatic architecture, high performing building envelopes and high-performance controlled ventilation.

With efficient energy buildings, the calculation of heating load and cooling load helps identify the equipment needed to maintain indoor air conditions with the requisite equipment. To better understand and estimate required cooling and heating capacity, building designers would need information about characteristics of the buildings and spaces (including the occupancy and activity level).

To evaluate the performance of the model, R-Square score will be used. R-Square is a statistical measure of how the data is to be fitted to the regression line. This is very important to create predictions that are close to real values.

A.3 Discussion of Hypothesis

The orientation of a building statistically significantly impacts the Heating and cooling of residential buildings.

We anticipate that we will be able to create a statistical model with a moderate to high success within a certain confidence interval that the prediction of output of Heating and Cooling Load when provided with the 8 features. The model will predict the most important principal components of interest relevant to the Heating and Cooling Load of the building. The model will also provide coefficients (Weights) for all the features in the regression model. We intend to identify the component / components which have a large coefficient value.

Section B: Data Collection

The dataset contains eight attributes (features X1 to X8) and two responses (Y1 and Y2). The intention would be to use the eight attributes to generate the two responses.

Information of Variables

The variables X1 to X8 along with y1 and y2 have been renamed as follows:-

1. X1 represents relative compactness
   The Relative Compactness is a good tool to assess the impact of shape on the energy efficiency of a building. This is an important component which indicates the compactness of its thermal envelope. Lower the relative compactness better the energy efficiency of the building.

2. X2 represents surface area
   The total Surface Area is measured in meter square. The surface area to volume ratio(S/V) help design energy efficient buildings(Expressed as meter^2)

3. X3 represents the Wall Area.
   This is the total area of the wall (Expressed as meter^2)

4. X4 represents the Roof Area.
   The area of the Roof is expressed as length square(meter^2)

5. X5 represents Overall Height
   The Total Height is expressed in units of length (meter)

6. X6 represents Orientation
   The building orientation which would be represented as North, South, East and West

7. X7 represents Glazing Area
   Part of Wall / Window which is made of glass
   This is represented as 0%, 10%, 25% and 40% of the floor area

8. X8 represents Glazing Area Distribution
   The distribution of the Glazing Area, the variance is explained as a Uniform, North, East, South and West.

9. y1 represents Heating Load
   Amount of Heat Energy to be added to a place, it is expressed in kWh/m^2

10. y2 represents Cooling Load
    Amount of Cooling Energy to be added to a place, it is expressed in kWh/m^2

Data Gathering

This dataset was created by Angeliki Xifara who was a Civil/ Structural Engineer Student. The dataset is maintained by Center for Machine Learning and Intelligent Systems hosted by UCI Machine Learning Repository. Since this is a public dataset and the citation policy allows using this data, I can confirm that I am able to use this data for my capstone project.

Since this dataset has been collected through a public repository, the advantages are the ease of data collection and no need of searching and motivating individuals to participate. The disadvantages of this data collection method include that the data may not be current or could be out of date.

The challenge involved in collecting a dataset was to pick a dataset which was not proprietary. Screening a dataset and ensuring its relevance and accuracy also poses to be a challenge in obtaining data.

Data Analytics Tools, Techniques and Justification

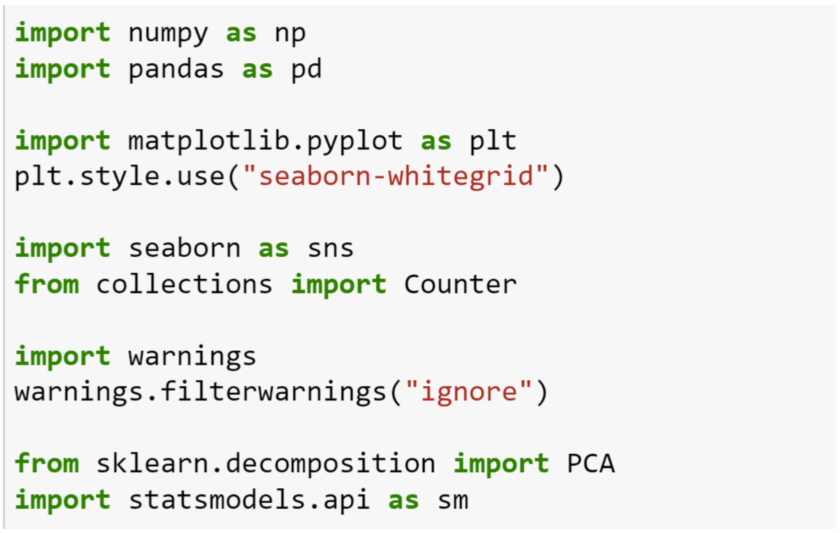
Among the data analytics tools and techniques would involve performing descriptive statistics which include identifying outliers, missing values and conducting pair-wise correlations. Further Principal Component Analysis would be performed to find the features that contribute most to the variance of the model. We shall use Ordinary Least Square model (OLS) to estimate coefficients of linear regression equation which describe the relationship between independent, predictor variable and output variable.

SECTION C: Data Extraction and Preparation Process

While cleaning the dataset, we identify 10 different variables

1. Relative Compactness
2. Surface Area
3. Wall Area
4. Roof Area
5. Overall Height
6. Orientation
7. Glazing Area
8. Glazing Area Distribution
9. Heating Load and
10. Cooling Load

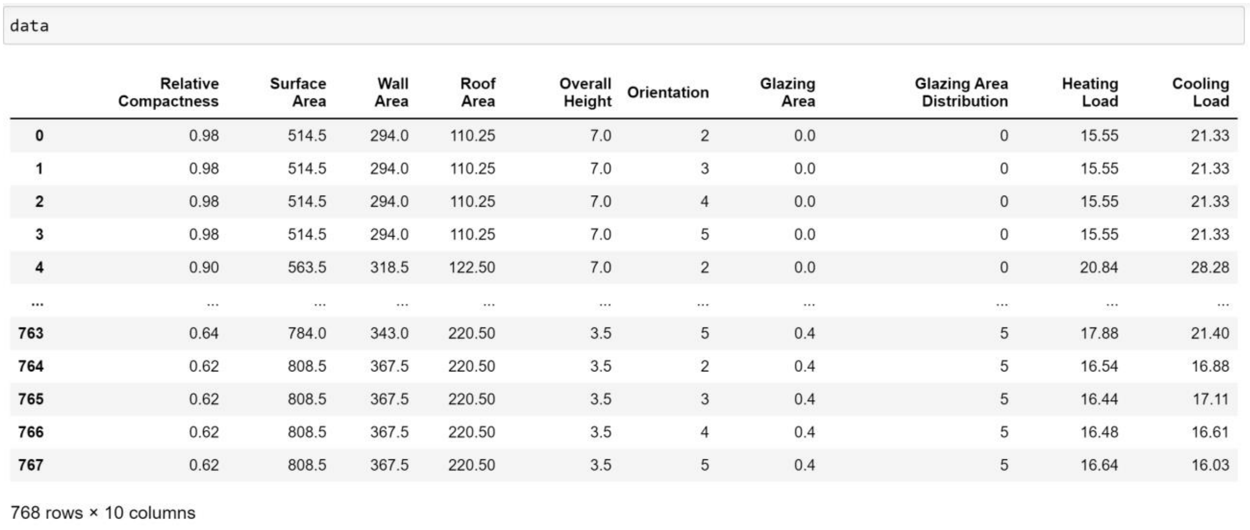
Our target variable being the orientation of the building.



```python
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
plt.style.use("seaborn-whitegrid")

import seaborn as sns
from collections import Counter

import warnings
warnings.filterwarnings("ignore")

from sklearn.decomposition import PCA
import statsmodels.api as sm
```

# **Reading the Building Energy Efficiency File**
data = pd.read_csv(r'C:/Users/harsh/OneDrive/Desktop/2021 WGU Masters in Data Analytics/D-214/Building Energy Efficiency.csv')

# **Renaming all the columns to be a correct representation**
data.rename(columns = {'X1' : 'Relative Compactness', 'X2' : 'Surface Area',
'X3' : 'Wall Area', 'X4' : 'Roof Area',
'X5' : 'Overall Height', 'X6' : 'Orientation',
'X7' : 'Glazing Area', 'X8' : 'Glazing Area Distribution',
'Y1' : 'Heating Load', 'Y2' : 'Cooling Load'}, inplace = True)



```
data
```

	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load	Cooling Load
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	15.55	21.33
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.55	21.33
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.55	21.33
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	15.55	21.33
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	20.84	28.28
...	...	...	...	...	...	...	...	...	...	...
763	0.64	784.0	343.0	220.50	3.5	5	0.4	5	17.88	21.40
764	0.62	808.5	367.5	220.50	3.5	2	0.4	5	16.54	16.88
765	0.62	808.5	367.5	220.50	3.5	3	0.4	5	16.44	17.11
766	0.62	808.5	367.5	220.50	3.5	4	0.4	5	16.48	16.61
767	0.62	808.5	367.5	220.50	3.5	5	0.4	5	16.64	16.03

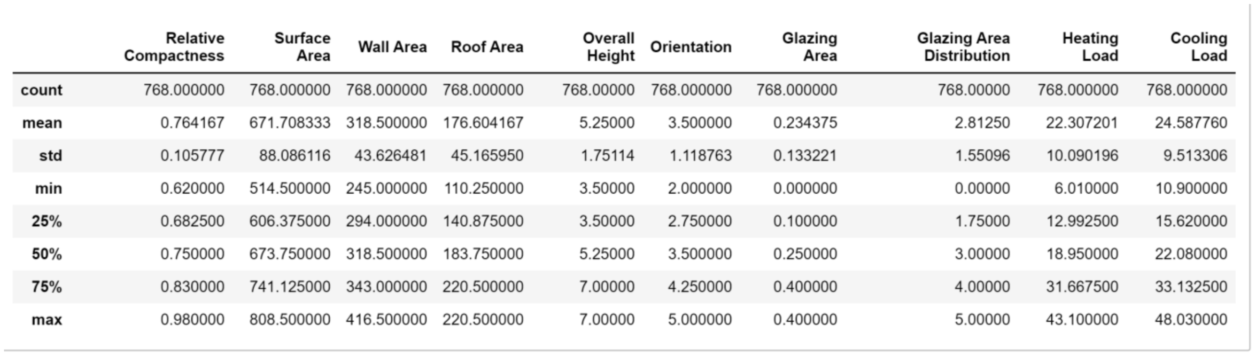
768 rows × 10 columns

# **Looking for empty rows or columns in the dataframe**

```
np.where(pd.isnull(data))
(array([], dtype=int64), array([], dtype=int64))
```

# **Generate a descriptive Statistics Summary**

```
data.describe()
```



	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load	Cooling Load
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.00000	768.000000	768.000000
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.81250	22.307201	24.587760
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.55096	10.090196	9.513306
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	0.000000	0.00000	6.010000	10.900000
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	0.100000	1.75000	12.992500	15.620000
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	0.250000	3.00000	18.950000	22.080000
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	0.400000	4.00000	31.667500	33.132500
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	0.400000	5.00000	43.100000	48.030000

# **Looking for any duplicates**

```
data_dup = data.loc[data.duplicated()]
print(data_dup)
Empty DataFrame
Columns: [Relative Compactness, Surface Area, Wall Area, Roof Area, Overal
l Height, Orientation, Glazing Area, Glazing Area Distribution, Heating Lo
ad, Cooling Load]
Index: []
```

At this juncture this dataset does not have any duplicate data, Unique data could be identified using this format.

```
dataframe ['Column name'].unique()
```

Since particularly looking for unique data does not contribute to our dataset. Only Unique data has been looked for across 2 columns

# **Relative Compactness and Surface Area**

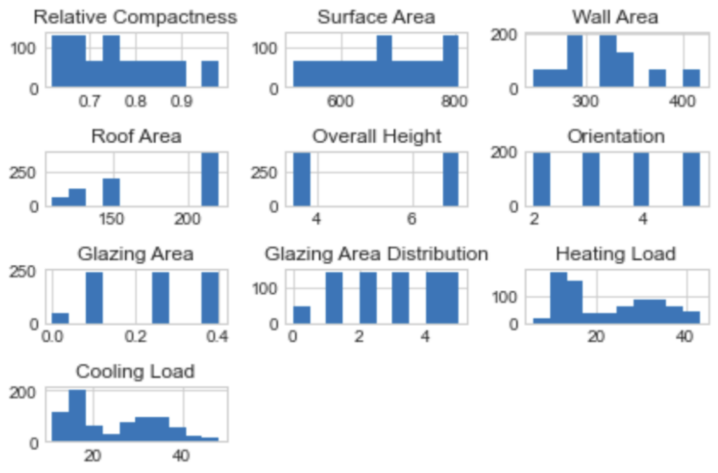
```
data['Relative Compactness'].unique()
array([0.98, 0.9 , 0.86, 0.82, 0.79, 0.76, 0.74, 0.71, 0.69, 0.66, 0.64,
    0.62])
data['Surface Area'].unique()
array([514.5, 563.5, 588. , 612.5, 637. , 661.5, 686. , 710.5, 735. ,
    759.5, 784. , 808.5])

data.head()
```

	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load	Cooling Load
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	15.55	21.33
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.55	21.33
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.55	21.33
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	15.55	21.33
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	20.84	28.28

```
data.shape
```

```
(768, 10)
```

```
data[['Relative Compactness', 'Surface Area', 'Wall Area', 'Roof Area', 'Overall Height',
'Orientation', 'Glazing Area', 'Glazing Area Distribution', 'Heating Load', 'Cooling Load']].hist()
plt.savefig('Buildinghistograms.jpg')
plt.tight_layout()
plt.grid(True)
```



# Exporting the clean dataset

```
data.to_csv('Energy_Efficiency.csv')
```

Among the data analytics tools and techniques would involve performing descriptive statistics which include identifying outliers, missing values and conducting pair-wise correlations. Further Principal Component Analysis would be performed to find the features that contribute most to the variance of the model. We shall use Ordinary Least Square model (OLS) to estimate coefficients of linear regression equation which describe the relationship between independent, predictor variable and output variable.

The tools and techniques being used for data extraction and data preparation would include:

- **Application type (select one):** Web
- **Programming/development language(s) you will use:** Python 3.8.8
- **Operating System(s)/Platform(s) you will use:** Windows 10
- **Database Management System you will use:** PostGre SQL

The tools and techniques for data extraction and preparation would include using Python Libraries: Numpy, Pandas, Matplotlib, Seaborn and Sklearn.

- **Numpy** is a math library that supports many operations of arrays from simple to complex. An advantage of Numpy is it lets us store large amounts of dense data effectively. An disadvantage of Numpy is using “Nan” in Numpy. Nan represents not a number. It was designed to address the problem of missing values. Lack of cross-platform support within python makes it difficult for the user. [Pedamkar, P]

- **Pandas** - An advantage of Pandas is that they provide extremely streamlined forms of data representation. An disadvantage of Pandas library is they have a pretty poor compatibility with 3D Matrices

- **Matplotlib** – It is a Python 2D Plotting library which provides charts and figures. It has been around since 2002 and helps creating statistical interfaces which is an advantage. It lacks beautiful themes which are an advantage when Seaborn is used.

- **Scikit-learn** (or in short, sklearn) helps us greatly simplify Machine Learning algorithms and related functions. Data preprocessing are those complex tasks, which require complex algorithms and coding, can be done using sklearn.

- **Seaborn** - It is used for high-level interface and attractive visualizations like univariate and bivariate analysis. It is built using python, numpy and Matplotlib which makes it a superset of the Matplotlib library.

SECTION D: Analysis

Prior to performing the data analysis, it was essential to import libraries, packages, clean the data and perform a descriptive statistics summary.

The data analysis process included performing a univariate analysis on the data. This was followed by performing Principal Component Analysis(PCA), and getting the total variance captured by components.

The variable being used for the PCA and regression analysis was Orientation. This analysis used Principal Component Analysis (PCA). PCA was used for feature extraction. PCA involves performing linear algebra operations to transform the dataset into a more compliant format with lesser data points and more important variables. The steps involved with PCA are

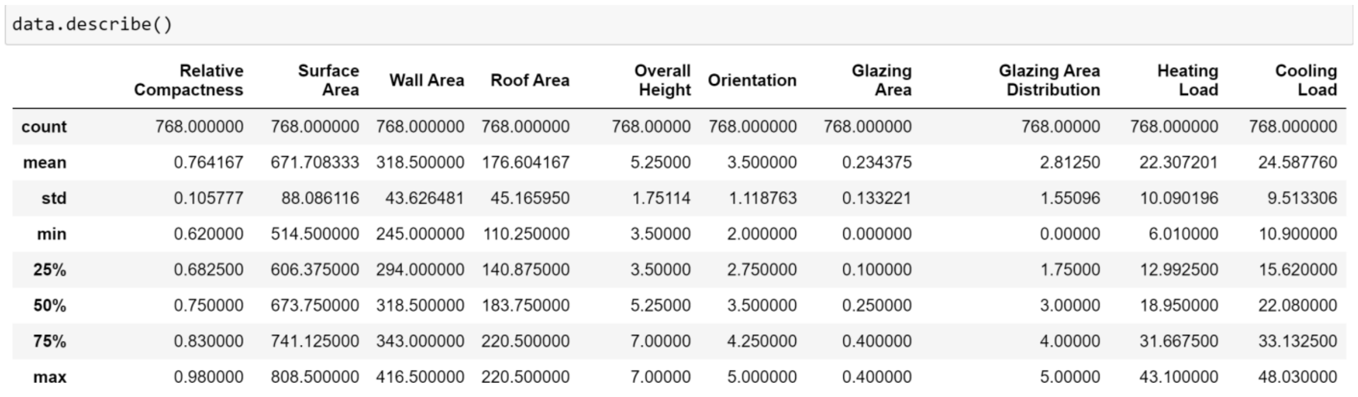
1) Standardize the dataset. This would involve subtracting the mean of the data points from the data point and dividing by standard deviation
2) Obtain the Eigenvalues and Eigenvectors from the co-variance matrix

One assumption of this approach is we would reduce the dimensions (Building Energy Efficiency features) of this n-dimensional dataset by projecting it into a another dataset. The intention is to ensure the features of the projected dataset are less than the n-dataset.

The advantage of Principal Component Analysis(PCA) is it removes correlated features, improves the algorithm performance, reduces overfitting and improves visualization. The disadvantage of PCA is independent variables become less interpretable, data standardization is needed before PCA. The target variable being used for the PCA and the regression analysis is “Orientation”.

PCA does help us to identify the patterns in the data based on the correlation between features. Since PCA is a dimensionality reduction tool, it wouldn’t make sense to use it if we have just 2 variables. In this case PCA was used to reduce the dimensionality of the data. It would yield principal components that are linear combinations of the variables.

The OLS regression (Linear regression) helps us identify the linear relationship between dependent variables and independent variable. It provides a formula indicating the components which have the most impact. The reason of using both PCA and Linear regression was to establish dimensionality reduction and achieve a relationship which would quantify the impact of various features on our target variable being (Orientation)



```
data.describe()
```

	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load	Cooling Load
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.00000	768.000000	768.000000
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.81250	22.307201	24.587760
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.55096	10.090196	9.513306
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	0.000000	0.00000	6.010000	10.900000
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	0.100000	1.75000	12.992500	15.620000
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	0.250000	3.00000	18.950000	22.080000
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	0.400000	4.00000	31.667500	33.132500
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	0.400000	5.00000	43.100000	48.030000

```
# Defining the x and y variables
        X = data.drop(["Orientation"], axis=1)
        y = data[["Orientation"]]

# Importing Standard Scaler from Scikit learn
    from sklearn.preprocessing import StandardScaler

# Standardize the data
        X_standardized = StandardScaler().fit_transform(X)

# Creating a covariance matrix
        mean_vec = np.mean(X_standardized, axis=0)
```

```
covariance_matrix = (X_standardized – mean_vec).T.dot((X_standardized))
```

# **Print Covariance Matrix**

```
print(‘Covariance matrix: \n%s’ %covariance_matrix)
Covariance matrix:
[[ 7.68000000e+02 -7.61780323e+02 -1.56504330e+02 -6.67256377e+02
   6.35709939e+02  8.30217605e-16 -2.38616339e-13 -4.33567041e-14]
 [-7.61780323e+02  7.68000000e+02  1.50145254e+02  6.76392589e+02
  -6.59057413e+02  6.24941573e-16 -6.13038245e-14 -4.80691837e-15]
 [-1.56504330e+02  1.50145254e+02  7.68000000e+02 -2.24499046e+02
   2.15789371e+02 -3.15185644e-15 -8.30319657e-15  1.57865985e-15]
 [-6.67256377e+02  6.76392589e+02 -2.24499046e+02  7.68000000e+02
  -7.46889398e+02 -5.52307528e-16  2.06786447e-14  1.60175715e-14]
 [ 6.35709939e+02 -6.59057413e+02  2.15789371e+02 -7.46889398e+02
   7.68000000e+02  0.00000000e+00  5.27355937e-16  1.66533454e-15]
 [ 8.30217605e-16  7.12349298e-15 -3.15185644e-15  6.39308866e-16
   0.00000000e+00  7.68000000e+02 -2.26697629e-16 -8.97977145e-16]
 [ 6.83897383e-14 -1.13242749e-14 -2.66453526e-15 -1.43218770e-14
  -8.93729535e-15 -7.10542736e-15  7.68000000e+02  1.63556522e+02]
 [ 3.88578059e-15 -4.74065232e-14 -3.33066907e-15  5.00155473e-14
  -1.08801856e-14  0.00000000e+00  1.63556522e+02  7.68000000e+02]]
```

#**Visualization of feature Scaling**

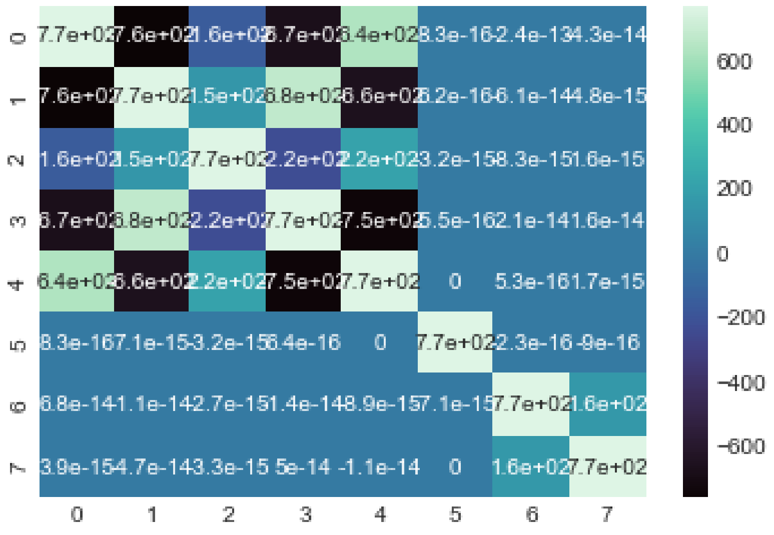
```
sns.heatmap(covariance_matrix, annot=True, cmap=”mako”, linecolor=’blue’)
plt.show()
```

# **Performing Eigendecomposition on Covariance matrix**

```
covariance_matrix =
np.cov(X_standardized.T)
eigen_values, eigen_vectors =
np.linalg.eig(covariance_matrix)
```

# **Print Eigenvectors and Eigenvalues**

```
print(‘Eigenvectors: \n%s’
%eigen_vectors)
print(‘Eigenvalues: \n%s’
%eigen_values)
```



Eigenvectors:

```
Eigenvectors:
[[-3.78238891e-01  3.78008796e-01 -9.61756718e-02 -5.65975635e-03
   3.08771994e-01  3.80927280e-01  6.81102821e-01  2.03987675e-02
   2.05876399e-13]
 [ 3.87644306e-01 -3.62159507e-01  9.09029957e-02  3.86157248e-03
  -1.31366114e-01 -3.21241533e-02  5.06594278e-01  2.08321525e-03
   6.59820428e-01]
 [-1.06274796e-01 -6.84293168e-01  3.72838836e-01 -1.38846195e-01
   4.78927381e-01  1.28886857e-01  8.44742358e-02 -4.74419654e-02
  -3.26789793e-01]
 [ 4.29333121e-01 -2.26710736e-02 -9.14223513e-02  7.08223988e-02
  -3.59401796e-01 -9.35723918e-02  4.53202119e-01  2.49438831e-02
  -6.76642766e-01]
 [-4.29144273e-01  1.91169148e-03  9.25574383e-02 -6.28059389e-02
  -1.12509443e-02 -8.50069942e-01  2.57287694e-01 -1.19906738e-01
   8.64216554e-14]
 [-4.63028039e-02 -3.04188990e-01 -6.46571998e-01  6.36338360e-01
   2.60730139e-01 -8.96227273e-02 -8.60850562e-03 -7.90993459e-02
  -4.83654154e-16]
 [-1.54822634e-02 -1.88262265e-01 -6.38291660e-01 -7.45930994e-01
   7.55139127e-03 -4.46553510e-03 -7.60963296e-04 -2.02678249e-02
  -4.94350258e-17]
 [-4.02139293e-01 -2.71861682e-01 -3.03969121e-02  7.74368829e-02
  -3.84351322e-01  9.56770181e-02  2.14429069e-02  7.74670861e-01
  -1.14187441e-15]
 [-4.03461900e-01 -2.35143943e-01  1.25538189e-02  6.61692085e-02
  -5.59438969e-01  2.97774511e-01  1.42016219e-02 -6.12818532e-01
   4.43266937e-15]]
Eigenvalues:
[5.22968969e+00 1.53537186e+00 1.22048303e+00 8.05822095e-01
 1.63362983e-01 3.31424290e-02 4.35960229e-03 1.95023494e-02
 1.24358263e-15]
```

Principal Components

# **List Descending sorted Eigenvalues**

```
eigen_pairs = [(np.abs(eigen_values[i]), eigen_vectors[:,i]) for I in
range(len(eigen_values))]
eigen_pairs.sort(key=lambda x: x[0], reverse = True)
```

# **Print List of Eigenvalues, descending**

```
print('Eigenvalues:')
for I in eigen_pairs:
   print(i[0])
```

Eigen Values:

```
3.707767110598057
1.2414678056060922
1.2145456604191944
1.0013037809647978
0.7880619015104022
0.0528226666098101
0.00446132201003707
1.2480332916375745e-15
```

# **Fitting a standardized matrix of features to a PCA class**

```
pca = PCA().fit(X_standardized)
```

# **Print Explained variance ratio**

```
print(pca.explained_variance_ratio_)
[4.62867411e-01 1.54981414e-01 1.51620528e-01 1.25000000e-01
 9.83794724e-02 6.59423589e-03 5.56939125e-04 1.37070577e-33]
```

# **Identification of Total Number of Components**

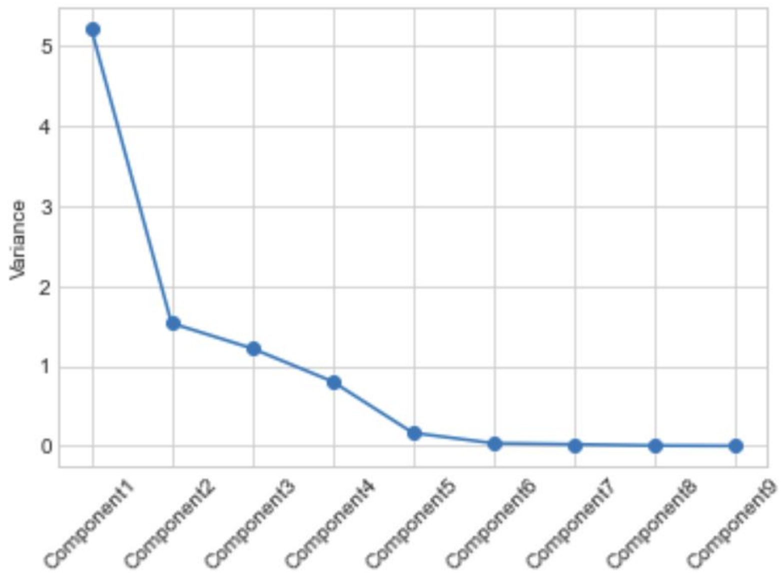
Perform the scree plot

```
def screeplot(pca, standardized_values):
   y=np.std(pca.transform(standardized_values), axis=0)**2
   x=np.arange(len(y)) + 1
   plt.plot(x, y, "o-")
   plt.xticks(x, ['Component' + str(i) for i in x], rotation=45)
   plt.ylabel('Variance')
   plt.show()
```

# **Visualize Scree plot**

```
screeplot(pca, X_standardized)
```

The Scree Plot indicates that 3 components have a variance above 1



Total Variance of Components

# **List importance of components**

```
def pca_summary(pca, standardized_data, out = True):
     names = ['PC' + str(j) for j in range(1, len(pca.explained_variance_ratio_)+1)]
    a = list(np.std(pca.transform(standardized_data), axis=0))
    b = list(pca.explained_variance_ratio_)
     c = [np.sum(pca.explained_variance_ratio_[:j]) for j in range(1,
len(pca.explained_variance_ratio_)+1)]
   columns = pd.MultiIndex.from_tuples([('standard_deviation','Standard Deviation'),
                    ('proportion_of_variance', 'Proportion of Variation'),
                    ('cumulative_proportion', 'Cumulative Proportion')])
   summary = pd.DataFrame(zip(a, b, c), index = names, columns= columns)
   if out:
      print('Component importance:')
   return summary
```

# **Display Summary**

```
summary = pca_summary(pca, X_standardized)
summary.standard_deviation**2
```

## Total Variance Captured by Components

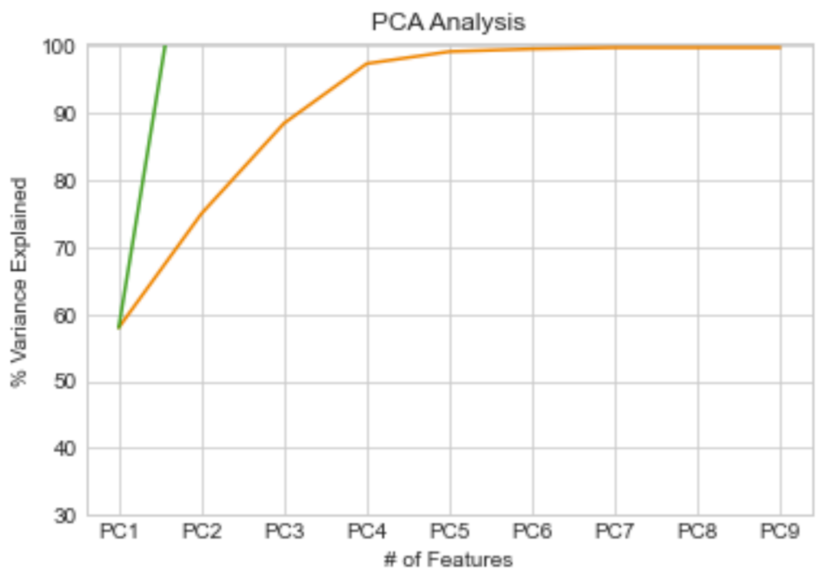
# Identify total variance captured by principal component

```
np.sum(summary.standard_deviation**2)
Standard Deviation    9.0
dtype: float64
```

	Standard Deviation
PC1	5.222880e+00
PC2	1.533373e+00
PC3	1.218894e+00
PC4	8.047728e-01
PC5	1.631503e-01
PC6	3.309927e-02
PC7	1.947696e-02
PC8	4.353926e-03
PC9	1.365120e-30

# Total Variance Captured by Components

```
var = np.cumsum(np.round(summary, decimals=3)*100)
plt.ylabel('% Variance Explained')
plt.xlabel('# of Features')
plt.title('PCA Analysis')
plt.ylim(30,100.5)
#plt.style.context('seaborn-whitegrid')
plt.plot(var)
```



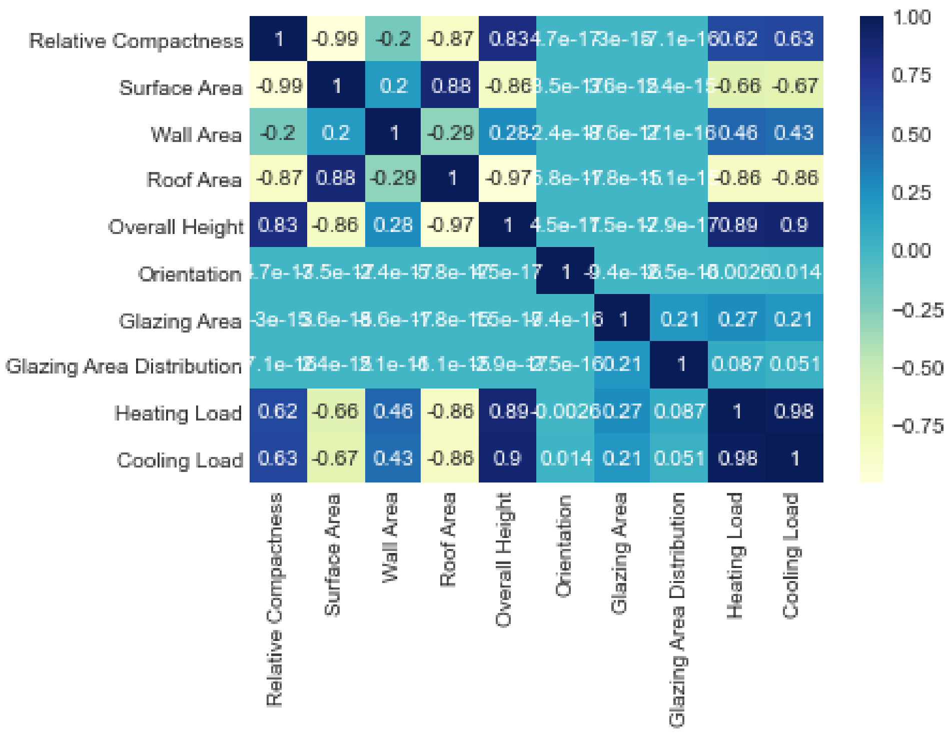
# Bivariate Analysis for heatmap

```
df_bivariate1 = data[['Relative Compactness', 'Surface Area', 'Wall Area', 'Roof Area',
'Overall Height', 'Orientation', 'Glazing Area', 'Glazing Area Distribution', 'Heating Load',
'Cooling Load']]
```

This indicates that the first 5 Components explain the principal component analysis.

# Generating a heatmap with annotations for variables

sns.heatmap(df_bivariate1.corr(), cmap="YlGnBu", annot=True)
    plt.show()



Develop the initial estimated regression equation that could be used to predict the Orientation, given the only continuous variables

    data['intercept'] = 1
    lm_Orientation = sm.OLS(data['Orientation'], data[['Relative Compactness', 'Surface Area', 'Wall Area', 'Roof Area', 'Overall Height', 'Glazing Area', 'Glazing Area Distribution', 'Heating Load', 'Cooling Load']]).fit()

print(lm_Orientation.summary())

```
OLS Regression Results
==============================================================================
=============
Dep. Variable:            Orientation   R-squared (uncentered):
0.908
Model:                            OLS   Adj. R-squared (uncentered):
0.907
Method:                 Least Squares   F-statistic:
937.8
Date:                Mon, 14 Feb 2022   Prob (F-statistic):
0.00
```

```
Time:                        13:04:46   Log-Likelihood:
-1172.9
No. Observations:                 768   AIC:
2362.
Df Residuals:                     760   BIC:
2399.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
===================
                              coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
-------------------
Relative Compactness        1.8766      0.732      2.565      0.011
0.440       3.313
Surface Area                0.0022      0.000      4.472      0.000
0.001       0.003
Wall Area                   0.0004      0.002      0.261      0.794
-0.003       0.003
Roof Area                   0.0009      0.001      0.922      0.357
-0.001       0.003
Overall Height              0.0131      0.113      0.116      0.908
-0.209       0.235
Glazing Area                0.1748      0.417      0.419      0.675
-0.644       0.994
Glazing Area Distribution   0.0071      0.027      0.262      0.793
-0.046       0.060
Heating Load               -0.0438      0.023     -1.929      0.054
-0.088       0.001
Cooling Load                0.0476      0.021      2.289      0.022
0.007       0.088
==============================================================================
====
Omnibus:                     2874.973   Durbin-Watson:                       2
.389
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                   57
.195
Skew:                          -0.019   Prob(JB):                         3.80
e-13
Kurtosis:                       1.664   Cond. No.                         4.48
e+16
==============================================================================
====

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
[3] The smallest eigenvalue is 2.27e-25. This might indicate that there
are strong multicollinearity problems or that the design matrix is singula
r.
```

<u>**Initial multiple regression model**</u>

Y = 1.8766*Relative Compactness + 0.0022*Surface Area + 0.0004*Wall Area + 0.0009*Roof Area + 0.0131*Overall Height + 0.1748*Glazing Area + 0.0071*Glazing Area Distribution - 0.0438* Heating Load + 0.0476*Cooling Load

Part E: Data Summary and Implications

From the above visualization, more than 70% of the total variance can be explained by the two principal components which have a variance greater than equal to 1.0 from the previous scree plot above. A total of 100% total variance cab be explained from the 5 principal components and approximately 95% of the variance from 4 principal components

Summary of Data Analysis

The dataset is being reduced to a much more manageable size which places greater importance on the significant numerical variable important to Building Energy Efficiency, using PCA Analysis. The context of the research question has been explained in detail in section A2. We avoid overfitting as we have less variables and understand their relationships lesser than before the analysis

Steps followed above include:

- Selecting a variable “Orientation” and standardizing the variable
- Creating a covariance matrix and perform an Eigen decomposition on Covariance Matrix
- Identifying the components of the Scree Plot
- Listing the variance order

On visualizing the Scree Plot we get a total of 5 components which indicate 100% of the variance. The OLS regression gave us R-squared (uncentered) value of 0.908 and an adjusted R-squared value of 0.907. We got an Alkaline Information Criteria (AIC) of 2362 and an Bayesian information criteria of 2399. The variance vs component plot indicate that a total of 3 components have a variance of 1.0. Since we get the smallest eigenvalue of 2.27 e-25, we get an indication of strong multicollinearity problem or the design matrix being singular.

The initial multiple regression model gives Y = 1.8766*Relative Compactness + 0.0022*Surface Area + 0.0004*Wall Area + 0.0009*Roof Area + 0.0131*Overall Height + 0.1748*Glazing Area + 0.0071*Glazing Area Distribution -0.0438* Heating Load + 0.0476*Cooling Load.

Summarizing the findings we get an understanding that the first 5 components explain 100% of the variance.

The seaborn heatmap gives the following correlation findings with respect to the heating load:-

- Heating Load and Relative Compactness is 0.62
- Heating Load and Wall Area is 0.46
- Heating Load and Overall Height is 0.89
- Heating Load and Orientation is 0.0026

This gives a clear indication that the orientation of the building does not statistically significantly affect the heating load of the building.

With respect to the Cooling Load we observe the following correlation findings:-

- Cooling load and Relative Compactness is 0.63
- Cooling load and Wall Area is 0.43
- Cooling load and Overall Height is 0.9
- Cooling load and Orientation is 0.014

This also gives an indication that the orientation of the building does not statistically significantly affect the cooling load of the building.

We also observe that the Overall height has a strong correlation of 0.89 and 0.9 with the heating and cooling load of the building which is a direct indicator of the building energy efficiency.

The research question was "Could we identify the principal components that have the biggest impact on the Building Energy Efficiency?" The limitation from the analysis is that the independent variables are less interpretable and potential loss of information. To study the dataset further, it would be ideal to have a more comprehensive research question for instance a) What component proves to be the most statistically significant? b) Potentially run an analysis to see if other parameters like glazing area, glazing area distribution among other features impact the Heating and Cooling load of the building. An additional course of action would involve performing a Time Series Analysis.

Section F: In-Text Citations and References

1. Bonilauri, E (October, 26th, 2015) The compactness ratio of a building. Retrieved from: https://emu.systems/2015/10/26/the-compactness-ratio-of-a-building

2. Data Collection Methods and Tools: Advantages and Tools. Retrieved from: https://www.intellspot.com/data-collection-methods-advantages/#:~:text=%E2%80%93%20High%20costs%20as%20this%20method,%2C%20organization%2C%20reporting%2C%20etc

3. Pedamkar, P Introduction to Numpy. Retrieved from: https://www.educba.com/introduction-to-numpy

4. What are the Pros and Cons of PCA. Retrieved from: https://www.i2tutorials.com/what-are-the-pros-and-cons-of-the-pca

5. Difference between Matplotlib vs Seaborn. Retrieved from: https://www.geeksforgeeks.org/difference-between-matplotlib-vs-seaborn