This notebook is for the REGRESSION model of the final project paper

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn.linear_model import LinearRegression as LR, Ridge, RidgeCV as RCV
from sklearn.preprocessing import PolynomialFeatures as pf
from sklearn.model_selection import train_test_split as tts
```

In [3]:
```
from sklearn.metrics import mean_squared_error as MSE, r2_score as r2s, mean_absolute
from sklearn.model_selection import cross_val_score as cvs
```

In [4]:
```
# nrgdf for energy dataframe
nrgdf = pd.read_excel('ENB2012_data.xlsx')
print(nrgdf)
```

```
       X1     X2     X3      X4   X5  X6   X7  X8     Y1     Y2
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 x 10 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

In [5]:
```
heat = nrgdf['Y1'].to_numpy()
cool = nrgdf['Y2'].to_numpy()
X = nrgdf.drop(['Y1','Y2'], axis = 1)
print(heat)
print(cool)
print(X)
```

```
[15.55  15.55  15.55  15.55  20.84  21.46  20.71  19.68  19.5   19.95
 19.34  18.31  17.05  17.41  16.95  15.98  28.52  29.9   29.63  28.75
 24.77  23.93  24.77  23.93   6.07   6.05   6.01   6.04   6.37   6.4
  6.366  6.4    6.85   6.79   6.77   6.81   7.18   7.1    7.1    7.1
 10.85  10.54  10.77  10.56   8.6    8.49   8.45   8.5   24.58  24.63
 24.63  24.59  29.03  29.87  29.14  28.09  26.28  26.91  26.37  25.27
 23.53  24.03  23.54  22.58  35.56  37.12  36.9   35.94  32.96  32.12
 32.94  32.21  10.36  10.43  10.36  10.39  10.71  10.8   10.7   10.75
 11.11  11.13  11.09  11.16  11.68  11.69  11.7   11.69  15.41  15.2
 15.42  15.21  12.96  12.97  12.93  13.02  24.29  24.31  24.13  24.25
```

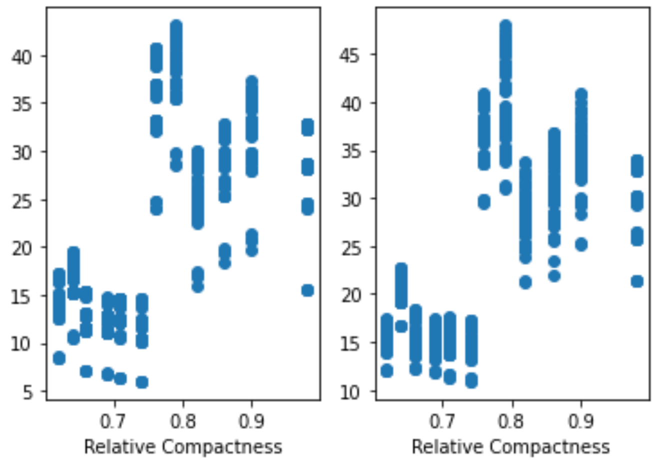
```
 28.88  29.68  28.83  27.9   26.48  27.02  26.33  25.36  23.75  24.23
 23.67  22.79  35.65  37.26  36.97  36.03  33.16  32.4   33.12  32.41
 10.42  10.46  10.32  10.45  10.64  10.72  10.55  10.68  11.45  11.46
 11.32  11.49  11.45  11.42  11.33  11.43  15.41  15.18  15.34  15.19
 12.88  13.    12.97  13.04  24.28  24.4   24.11  24.35  28.07  29.01
 29.62  29.05  25.41  26.47  26.89  26.46  22.93  23.84  24.17  23.87
 35.78  35.48  36.97  36.7   32.52  33.28  32.33  33.24  10.39  10.34
 10.35  10.38  10.77  10.68  10.68  10.7   11.22  11.16  11.1   11.14
 11.59  11.6   11.53  11.61  15.16  15.36  15.12  15.36  12.68  12.63
 12.71  12.73  24.38  24.23  24.04  24.32  29.06  28.05  28.86  29.79
 26.44  25.37  26.33  27.03  23.8   22.8   23.59  24.24  36.86  35.89
 35.45  37.1   33.08  32.38  33.09  32.31  10.08  10.15  10.07  10.14
 10.66  10.68  10.53  10.72  11.18  11.22  11.07  11.2   11.44  11.42
 11.33  11.43  15.4   15.19  15.32  15.16  12.85  13.04  13.    13.
 24.35  24.33  24.03  24.26  29.83  29.08  28.03  29.02  27.03  26.45
 25.36  26.45  24.37  23.89  22.89  23.86  37.03  36.71  36.77  35.48
 32.31  33.21  32.46  33.27  10.47  10.37  10.34  10.39  10.78  10.7
 10.67  13.69  11.21  11.14  11.11  11.16  11.38  11.34  11.22  11.34
 15.16  15.37  15.12  15.36  12.59  12.74  12.8   12.62  28.15  28.15
 28.37  28.41  32.68  33.48  32.84  32.    29.54  30.05  29.6   28.66
 26.84  27.27  26.97  26.19  38.67  40.03  39.86  39.04  36.96  36.13
 36.91  36.43  12.43  12.5   12.41  12.45  12.57  12.65  12.57  12.63
 12.78  12.93  12.73  12.72  13.17  13.18  13.17  13.18  17.5   17.35
 17.52  17.37  15.09  15.12  15.08  15.16  28.67  28.57  28.18  28.6
 32.46  33.27  32.33  31.66  29.34  29.87  29.27  28.4   25.74  25.98
 25.38  24.94  38.57  40.19  39.97  38.98  36.95  36.28  36.86  36.45
 12.35  12.45  12.16  12.3   12.33  12.29  12.2   12.49  12.85  12.87
 12.73  12.95  13.05  12.93  12.77  13.    17.14  16.84  17.02  17.11
 14.34  14.66  14.6   14.6   28.67  28.56  28.17  28.63  31.63  32.4
 32.68  32.29  28.4   29.4   29.43  29.07  24.7   25.48  25.37  25.17
 39.04  38.35  39.81  39.83  35.99  36.59  35.64  36.52  11.8   12.03
 11.98  11.69  12.41  12.28  12.1   12.19  12.34  12.46  12.31  12.12
 12.97  13.01  12.74  12.84  16.83  16.93  16.66  16.86  13.91  14.34
 13.95  13.99  28.7   28.55  28.15  28.62  32.67  31.69  32.07  33.28
 29.47  28.42  29.08  29.88  25.66  24.96  25.43  26.    40.    38.84
 38.33  40.12  36.95  36.45  36.81  36.26  12.32  12.3   12.18  12.43
 12.36  12.49  12.17  12.28  12.91  12.95  12.67  12.86  12.95  13.
 12.86  12.92  16.99  16.69  16.56  16.62  14.33  14.61  14.61  14.65
 28.69  28.58  28.15  28.61  33.13  32.31  31.53  32.46  29.71  29.09
 28.31  29.39  25.7   25.17  24.6   25.49  39.89  39.83  39.01  38.65
 35.69  36.64  36.06  36.7   12.12  11.67  11.64  12.02  12.27  12.19
 12.25  12.27  12.47  12.12  12.18  12.47  12.93  12.82  12.78  13.02
 16.73  16.86  16.76  16.92  13.68  13.99  14.16  13.86  32.26  32.26
 32.49  32.53  36.47  37.24  36.66  35.96  31.89  32.39  32.09  31.29
 29.22  29.91  29.53  28.65  41.4   42.62  42.5   41.67  40.78  39.97
 40.71  40.43  14.52  14.61  14.5   14.55  14.51  14.6   14.5   14.58
 14.51  14.7   14.42  14.42  15.23  15.23  15.23  15.23  19.52  19.36
 19.48  19.42  15.09  17.17  17.14  17.14  32.82  32.71  32.24  32.72
 35.84  36.57  36.06  35.69  32.48  32.74  32.13  31.64  28.95  29.49
 28.64  28.01  41.64  43.1   42.74  41.92  40.78  40.15  40.57  40.42
 14.54  14.45  14.18  14.5   14.7   14.66  14.4   14.71  14.75  14.71
 14.33  14.62  15.34  15.29  15.09  15.3   19.2   18.88  18.9   19.12
 16.76  17.23  17.26  17.15  32.82  32.69  32.23  32.75  34.24  34.95
 35.05  34.29  31.28  32.12  32.05  31.84  28.67  29.67  29.47  28.91
 41.26  41.3   42.49  42.08  39.32  39.84  38.89  39.68  13.97  14.22
 14.1   13.78  14.07  14.03  13.94  13.86  14.32  14.56  14.33  14.08
 15.16  15.18  14.72  14.9   18.48  18.71  18.48  18.46  16.47  16.35
 16.55  16.74  32.85  32.67  32.21  32.74  36.45  35.73  35.4   36.57
 32.38  31.66  32.15  32.75  28.93  28.05  28.64  29.52  42.77  41.73
 41.32  42.96  40.68  40.4   40.6   40.11  14.37  14.48  14.32  14.44
 14.6   14.7   14.47  14.66  14.54  14.62  14.53  14.71  15.34  15.29
 15.09  15.3   19.06  19.13  19.    18.84  16.44  16.9   16.94  16.77
 32.84  32.72  32.21  32.73  35.67  35.01  34.72  35.24  32.31  31.81
 31.12  32.06  30.    29.5   29.06  29.92  42.11  41.96  41.09  40.79
 38.82  39.72  39.31  39.86  14.41  14.19  14.17  14.39  12.43  12.63
 12.76  12.42  14.12  14.28  14.37  14.21  14.96  14.92  14.92  15.16
 17.69  18.19  18.16  17.88  16.54  16.44  16.48  16.64 ]
[21.33 21.33 21.33 21.33 28.28 25.38 25.16 29.6  27.3  21.97 23.49 27.87
 23.77 21.46 21.16 24.93 37.73 31.27 30.93 39.44 29.79 29.68 29.79 29.4
 10.9  11.19 10.94 11.17 11.27 11.72 11.29 11.67 11.74 12.05 11.73 11.93
```

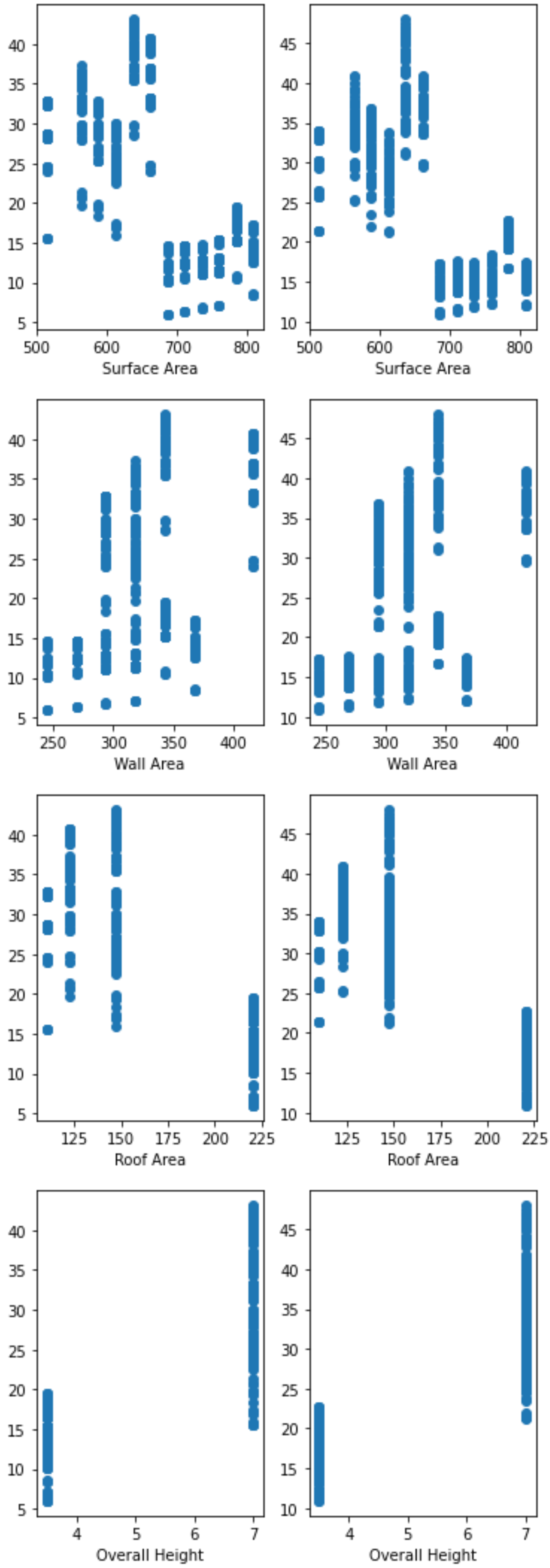
```
 12.4  12.23 12.4  12.14 16.78 16.8  16.75 16.67 12.07 12.22 12.08 12.04
 26.47 26.37 26.44 26.29 32.92 29.87 29.58 34.33 30.89 25.6  27.03 31.73
 27.31 24.91 24.61 28.51 41.68 35.28 34.43 43.33 33.87 34.07 34.14 33.67
 13.43 13.71 13.48 13.7  13.8  14.28 13.87 14.27 14.28 14.61 14.3  14.45
 13.9  13.72 13.88 13.65 19.37 19.43 19.34 19.32 14.34 14.5  14.33 14.27
 25.95 25.63 26.13 25.89 32.54 29.44 29.36 34.2  30.91 25.63 27.36 31.9
 27.38 25.02 24.8  28.79 41.07 34.62 33.87 42.86 33.91 34.07 34.17 33.78
 13.39 13.72 13.57 13.79 13.67 14.11 13.8  14.21 13.2  13.54 13.32 13.51
 14.86 14.75 15.   14.74 19.23 19.34 19.32 19.3  14.37 14.57 14.27 14.24
 25.68 26.02 25.84 26.14 34.14 32.85 30.08 29.67 31.73 31.01 25.9  27.4
 28.68 27.54 25.35 24.93 43.12 41.22 35.1  34.29 33.85 34.11 34.48 34.5
 13.6  13.36 13.65 13.49 14.14 13.77 14.3  13.87 14.44 14.27 14.67 14.4
 13.46 13.7  13.59 13.83 19.14 19.18 19.37 19.29 14.09 14.23 14.14 13.89
 25.91 25.72 26.18 25.87 29.34 33.91 32.83 29.92 27.17 31.76 31.06 25.81
 24.61 28.61 27.57 25.16 34.25 43.3  41.86 35.29 34.11 33.62 33.89 34.05
 13.2  13.36 13.21 13.53 13.67 14.12 13.79 14.2  14.29 14.49 14.42 14.73
 14.86 14.67 15.   14.83 19.24 19.25 19.42 19.48 14.37 14.34 14.28 14.47
 25.64 25.98 25.88 26.18 29.82 29.52 34.45 33.01 25.82 27.33 32.04 31.28
 25.11 24.77 28.88 27.69 34.99 34.18 43.14 41.26 34.25 34.35 33.64 33.88
 13.65 13.44 13.72 13.5  14.18 13.75 14.26 13.89 14.55 14.28 14.46 14.39
 14.54 14.81 14.65 14.87 19.24 19.18 19.26 19.29 14.24 13.97 13.99 14.15
 29.79 29.79 29.28 29.49 36.12 33.17 32.71 37.58 33.98 28.61 30.12 34.73
 30.17 27.84 27.25 31.39 43.8  37.81 36.85 45.52 36.85 37.58 37.45 36.62
 15.19 15.5  15.28 15.5  15.42 15.85 15.44 15.81 15.21 15.63 15.48 15.78
 16.39 16.27 16.39 16.19 21.13 21.19 21.09 21.08 15.77 15.95 15.77 15.76
 29.62 29.69 30.18 30.02 35.56 32.64 32.77 37.72 33.37 27.89 29.9  34.52
 28.27 26.96 26.72 29.88 43.86 37.41 36.77 45.97 36.87 37.35 37.28 36.81
 14.73 15.1  15.18 15.44 14.91 15.4  14.94 15.32 15.52 15.85 15.66 15.99
 15.89 15.85 16.22 15.87 20.47 20.56 20.48 20.43 15.32 15.64 15.14 15.3
 29.43 29.78 30.1  30.19 36.35 35.1  32.83 32.46 33.52 32.93 28.38 29.82
 28.77 27.76 26.95 26.41 45.13 43.66 37.76 36.87 36.07 36.44 37.28 37.29
 14.49 13.79 14.72 14.76 14.92 14.74 15.57 14.94 14.92 14.38 15.44 15.17
 15.53 15.8  16.14 16.26 19.87 20.03 20.46 20.28 14.89 14.96 14.89 14.35
 29.61 29.59 30.19 30.12 32.12 37.12 36.16 33.16 29.45 34.19 33.93 28.31
 26.3  29.43 28.76 27.34 36.26 45.48 44.16 37.26 37.2  36.76 37.05 37.51
 14.92 15.24 15.03 15.35 14.67 15.09 15.2  15.64 15.37 15.73 15.83 16.13
 15.95 15.59 16.17 16.14 19.65 19.76 20.37 19.9  15.41 15.56 15.07 15.38
 29.53 29.77 30.   30.2  32.25 32.   37.19 35.62 28.02 29.43 34.15 33.47
 26.53 26.08 29.31 28.14 37.54 36.66 45.28 43.73 36.93 37.01 35.73 36.15
 14.48 14.58 14.81 14.03 15.27 14.71 15.23 14.97 15.14 14.97 15.22 14.6
 15.83 16.03 15.8  16.06 20.13 20.01 20.19 20.29 15.19 14.61 14.61 14.75
 33.37 33.34 32.83 33.04 39.28 36.38 35.92 40.99 35.99 30.66 31.7  36.73
 31.71 29.13 28.99 33.54 45.29 39.07 38.35 46.94 39.55 40.85 40.63 39.48
 16.94 17.25 17.03 17.25 17.1  17.51 17.12 17.47 16.5  17.   16.87 17.2
 18.14 18.03 18.14 17.95 22.72 22.73 22.72 22.53 17.2  17.21 17.15 17.2
 32.96 33.13 33.94 33.78 38.35 35.39 34.94 40.66 35.48 30.53 32.28 36.86
 30.34 27.93 28.95 32.92 45.59 39.41 38.84 48.03 39.48 40.4  40.47 39.7
 16.43 16.93 16.99 17.03 16.77 17.37 17.27 17.51 16.44 17.01 17.23 17.22
 17.85 17.89 18.36 18.15 21.72 22.07 22.09 21.93 17.36 17.38 16.86 16.99
 32.78 33.24 33.86 34.   37.26 35.04 33.82 33.31 35.22 34.7  30.11 31.6
 32.43 30.65 29.77 29.64 46.44 44.18 38.81 38.23 38.17 38.48 39.66 40.1
 16.08 15.39 16.57 16.6  16.11 15.47 16.7  16.1  16.35 15.84 16.99 17.02
 17.04 17.63 18.1  18.22 20.78 20.72 21.54 21.53 16.9  17.14 16.56 16.
 32.95 33.06 33.95 33.88 33.98 39.92 39.22 36.1  31.53 36.2  36.21 31.
 28.2  32.35 31.14 28.43 38.33 47.59 46.23 39.56 40.36 39.67 39.85 40.77
 16.61 16.74 16.9  17.32 16.85 17.2  17.23 17.74 16.81 16.88 16.9  17.39
 17.86 17.82 18.36 18.24 21.68 21.54 22.25 22.49 17.1  16.79 16.58 16.79
 32.88 33.23 33.76 34.01 33.94 33.14 38.79 37.27 29.69 31.2  36.26 35.71
 29.93 29.56 33.84 32.54 38.56 37.7  47.01 44.87 39.37 39.8  37.79 38.18
 16.69 16.62 16.94 16.7  15.59 14.58 15.33 15.31 16.63 15.87 16.54 16.74
 17.64 17.79 17.55 18.06 20.82 20.21 20.71 21.4  16.88 17.11 16.61 16.03]
       X1     X2     X3      X4   X5  X6   X7  X8
0    0.98  514.5  294.0  110.25  7.0   2  0.0   0
1    0.98  514.5  294.0  110.25  7.0   3  0.0   0
2    0.98  514.5  294.0  110.25  7.0   4  0.0   0
3    0.98  514.5  294.0  110.25  7.0   5  0.0   0
4    0.90  563.5  318.5  122.50  7.0   2  0.0   0
..    ...    ...    ...     ...  ...  ..  ...  ..
763  0.64  784.0  343.0  220.50  3.5   5  0.4   5
764  0.62  808.5  367.5  220.50  3.5   2  0.4   5
```

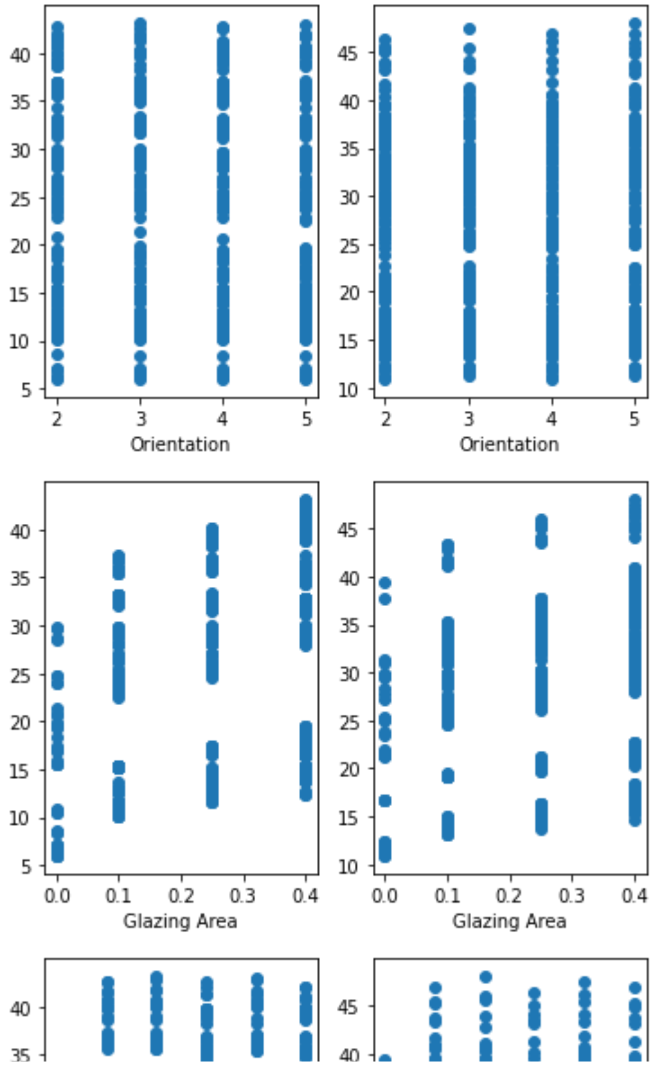
```
765  0.62  808.5  367.5  220.50  3.5   3  0.4   5
766  0.62  808.5  367.5  220.50  3.5   4  0.4   5
767  0.62  808.5  367.5  220.50  3.5   5  0.4   5
```

In [6]:

```
hl_rc = plt.subplot(1,2,1, xlabel = "Relative Compactness")
hl_rc = plt.scatter(X['X1'], heat)
cl_rc = plt.subplot(1,2,2, xlabel = "Relative Compactness")
cl_rc = plt.scatter(X['X1'], cool)
plt.show()
hl_rc = plt.subplot(1,2,1, xlabel = "Surface Area")
hl_rc = plt.scatter(X['X2'], heat)
cl_rc = plt.subplot(1,2,2, xlabel = "Surface Area")
cl_rc = plt.scatter(X['X2'], cool)
plt.show()
hl_rc = plt.subplot(1,2,1, xlabel = "Wall Area")
hl_rc = plt.scatter(X['X3'], heat)
cl_rc = plt.subplot(1,2,2, xlabel = "Wall Area")
cl_rc = plt.scatter(X['X3'], cool)
plt.show()
hl_rc = plt.subplot(1,2,1, xlabel = "Roof Area")
hl_rc = plt.scatter(X['X4'], heat)
cl_rc = plt.subplot(1,2,2, xlabel = "Roof Area")
cl_rc = plt.scatter(X['X4'], cool)
plt.show()
hl_rc = plt.subplot(1,2,1, xlabel = "Overall Height")
hl_rc = plt.scatter(X['X5'], heat)
cl_rc = plt.subplot(1,2,2, xlabel = "Overall Height")
cl_rc = plt.scatter(X['X5'], cool)
plt.show()
hl_rc = plt.subplot(1,2,1, xlabel = "Orientation")
hl_rc = plt.scatter(X['X6'], heat)
cl_rc = plt.subplot(1,2,2, xlabel = "Orientation")
cl_rc = plt.scatter(X['X6'], cool)
plt.show()
hl_rc = plt.subplot(1,2,1, xlabel = "Glazing Area")
hl_rc = plt.scatter(X['X7'], heat)
cl_rc = plt.subplot(1,2,2, xlabel = "Glazing Area")
cl_rc = plt.scatter(X['X7'], cool)
plt.show()
hl_rc = plt.subplot(1,2,1, xlabel = "Glazing Area Distribution")
hl_rc = plt.scatter(X['X8'], heat)
cl_rc = plt.subplot(1,2,2, xlabel = "Glazing Area Distribution")
cl_rc = plt.scatter(X['X8'], cool)
plt.show()
```




Surface Area
Surface Area
Wall Area
Wall Area
Roof Area
Roof Area
Overall Height
Overall Height



I will be keeping heating and cooling seperate models, but I will plot them together at every turn. I will have both have a train:test set ratio of 20:80, because looking at the overal data graphs involving the various columns and their relation to the heat or cooling loads shows low variance and so to avoid bias I lower the training set size and aim for a percentage of 80-90 correct.

In [7]:

```
from sklearn.preprocessing import StandardScaler
scaler1 = StandardScaler()
scaler2 = StandardScaler()
```

In [8]:

```
# first the heat load (hl) train_test_split
hl_xtr, hl_xte, hl_ytr, hl_yte = tts(X.to_numpy(), heat, test_size = .80)
# now the cooling load (cl) train_test_split
cl_xtr, cl_xte, cl_ytr, cl_yte = tts(X.to_numpy(), cool, test_size = .80)
```

In [9]:

```
hl_xtr = scaler1.fit_transform(hl_xtr)
hl_xte = scaler1.transform(hl_xte)

cl_xtr = scaler2.fit_transform(cl_xtr)
cl_xte = scaler2.transform(cl_xte)
```

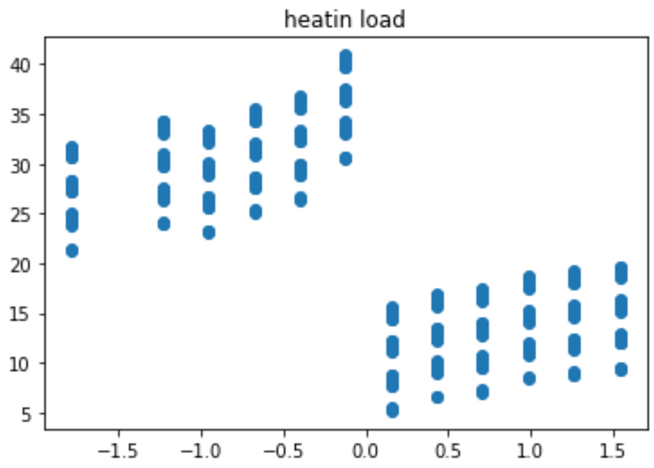
In [10]:

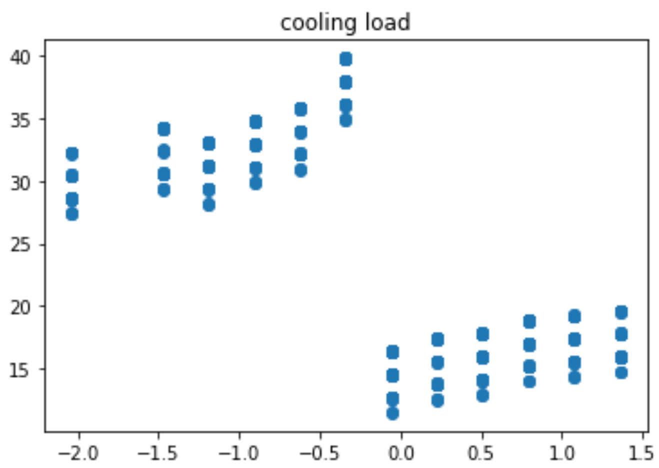
```
# creating and fitting the linear regressions
hl_lr = LR().fit(hl_xtr, hl_ytr)
cl_lr = LR().fit(cl_xtr, cl_ytr)
```

In [11]:

```
hl_pred = hl_lr.predict(hl_xte)
cl_pred = cl_lr.predict(cl_xte)
plt.title("heatin load")
plt.scatter(hl_xte[:,1], hl_pred)
plt.show()
plt.title("cooling load")
plt.scatter(cl_xte[:,1], cl_pred)
plt.show()
print("Heating Load Prediction Scoring: %0.2f" % hl_lr.score(hl_xte, hl_yte))
print("Cooling Load Prediction Scoring: %0.2f" % cl_lr.score(cl_xte, cl_yte))

print("Heating MSE: %0.5f" % MSE(hl_yte, hl_pred, squared = False))
print("Heating MAE: %0.5f" % MAE(hl_yte, hl_pred))
print("Heating R2_score: %0.5f" % r2s(hl_yte, hl_pred,))
print("Cooling MSE: %0.5f" % MSE(cl_yte, cl_pred, squared = False))
print("Cooling MAE: %0.5f" % MAE(cl_yte, cl_pred))
print("Cooling R2_score: %0.5f" % r2s(cl_yte, cl_pred))
```





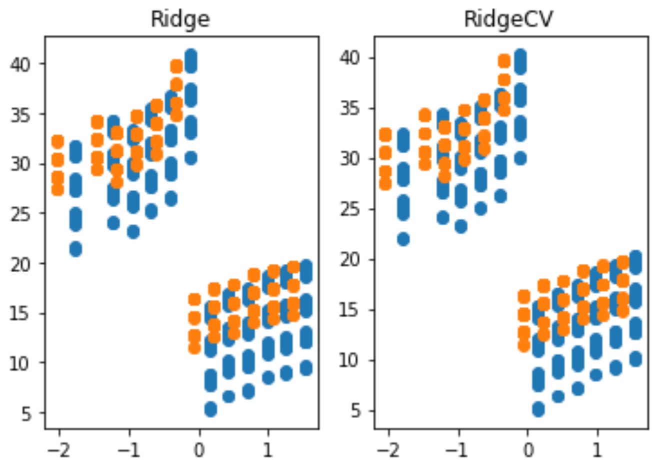

```
Heating Load Prediction Scoring: 0.91
Cooling Load Prediction Scoring: 0.89
Heating MSE: 2.94078
Heating MAE: 2.12236
Heating R2_score: 0.91481
Cooling MSE: 3.24760
Cooling MAE: 2.29925
Cooling R2_score: 0.88524
```

In [12]:

```
# now time to create and fit using Ridge and RidgeCV modeling at default cv and alpha
hl_ridge = Ridge(alpha = .01).fit(hl_xtr, hl_ytr)
hl_rcv = RCV(alphas = [0.001,0.01, 0.1, 1, 10, 100,1000,100000], cv = 3).fit(hl_xtr,
cl_ridge = Ridge(alpha = 0.01).fit(cl_xtr, cl_ytr)
cl_rcv = RCV(alphas = [0.001,0.01, 0.1, 1, 10, 100,1000,100000], cv = 3).fit(cl_xtr,
```

In [13]:

```
# predictions
hl_pred2 = hl_ridge.predict(hl_xte)
cl_pred2 = cl_ridge.predict(cl_xte)
hl_pred3 = hl_rcv.predict(hl_xte)
cl_pred3 = cl_rcv.predict(cl_xte)
#plotting
plt.subplot(1,2,1)
plt.title("Ridge")
plt.scatter(hl_xte[:,1], hl_pred2)
plt.scatter(cl_xte[:,1], cl_pred2)
plt.subplot(1,2,2)
plt.title("RidgeCV")
plt.scatter(hl_xte[:,1], hl_pred3)
plt.scatter(cl_xte[:,1], cl_pred3)
plt.show()
#scoring for normal Ridge scoring
hl_rid_scores = cvs(hl_ridge, hl_xtr, hl_ytr, scoring='neg_mean_absolute_error', n_jo
print('Heating Load Mean and Standard Deviation(normal): %.3f (%.3f)' % (np.mean(hl_r
print("Heating MAE(normal): %0.5f" % MAE(hl_yte, hl_pred2))
print("Heating MSE(normal): %0.5f" % MSE(hl_yte, hl_pred2))
cl_rid_scores = cvs(cl_ridge, cl_xtr, cl_ytr, scoring='neg_mean_absolute_error', n_jo
print('Cooling Load Mean and Standard Deviation(normal): %.3f (%.3f)' % (np.mean(cl_r
print("Cooling MAE(normal): %0.5f" % MAE(cl_yte, cl_pred2))
print("Cooling MSE(normal): %0.5f" % MSE(cl_yte, cl_pred2))
#scoring for RidgeCV scoring
hl_rcv_scores = cvs(hl_rcv, hl_xtr, hl_ytr, scoring='neg_mean_absolute_error', n_jobs
print('Heating Load Mean and Standard Deviation(3-fold): %.3f (%.3f)' % (np.mean(hl_r
print("Heating MAE(3-fold): %0.5f" % MAE(hl_yte, hl_pred3))
print("Heating MSE(3-fold): %0.5f" % MSE(hl_yte, hl_pred3))
cl_rcv_scores = cvs(cl_rcv, cl_xtr, cl_ytr, scoring='neg_mean_absolute_error', n_jobs
print('Cooling Load Mean and Standard Deviation(3-fold): %.3f (%.3f)' % (np.mean(cl_r
print("Cooling MAE(3-fold): %0.5f" % MAE(cl_yte, cl_pred3))
print("Cooling MSE(3-fold): %0.5f" % MSE(cl_yte, cl_pred3))
#score function scores
print("Heating Load Prediction Scoring(normal): %0.2f" % hl_ridge.score(hl_xte, hl_yt
print("Cooling Load Prediction Scoring(normal): %0.2f" % cl_ridge.score(cl_xte, cl_yt
print("Heating Load Prediction Scoring(3-fold): %0.2f" % hl_rcv.score(hl_xte, hl_yte)
print("Cooling Load Prediction Scoring(3-fold): %0.2f" % cl_rcv.score(cl_xte, cl_yte)
# r2 scoring
print("Heating R2_score: %0.5f" % r2s(hl_yte, hl_pred2))
print("Cooling R2_score: %0.5f" % r2s(cl_yte, cl_pred2))
print("Heating R2_score(3-fold): %0.5f" % r2s(hl_yte, hl_pred3))
print("Cooling R2_score(3-fold): %0.5f" % r2s(cl_yte, cl_pred3))
```




```
Heating Load Mean and Standard Deviation(normal): -2.398 (0.390)
Heating MAE(normal): 2.12120
Heating MSE(normal): 8.64593
Cooling Load Mean and Standard Deviation(normal): -2.316 (0.347)
Cooling MAE(normal): 2.29927
Cooling MSE(normal): 10.54834
Heating Load Mean and Standard Deviation(3-fold): -2.383 (0.371)
Heating MAE(3-fold): 2.10247
```

```
Heating MSE(3-fold): 8.67634
Cooling Load Mean and Standard Deviation(3-fold): -2.320 (0.328)
Cooling MAE(3-fold): 2.30031
Cooling MSE(3-fold): 10.56559
Heating Load Prediction Scoring(normal): 0.91
Cooling Load Prediction Scoring(normal): 0.89
Heating Load Prediction Scoring(3-fold): 0.91
Cooling Load Prediction Scoring(3-fold): 0.89
Heating R2_score: 0.91484
Cooling R2_score: 0.88522
Heating R2_score(3-fold): 0.91454
Cooling R2_score(3-fold): 0.88503
```

In [14]:

```
# Now the polynomial Regression
heat_poly = pf(degree = 2, include_bias= False)
hx_poly = heat_poly.fit_transform(hl_xtr)
cool_poly = pf(degree = 2, include_bias= False)
cx_poly = cool_poly.fit_transform(cl_xtr)
heat_poly2 = pf(degree = 3, include_bias= False)
hx_poly2 = heat_poly2.fit_transform(hl_xtr)
cool_poly2 = pf(degree = 3, include_bias= False)
cx_poly2 = cool_poly2.fit_transform(cl_xtr)
```

In [15]:

```
hl_lr2 = LR().fit(hx_poly, hl_ytr)
hl_lr3 = LR().fit(hx_poly2, hl_ytr)
cl_lr2 = LR().fit(cx_poly, cl_ytr)
cl_lr3 = LR().fit(cx_poly2, cl_ytr)
```

In [16]:

```
hl_new_poly = heat_poly.transform(hl_xte)
hl_new_poly2 = heat_poly2.transform(hl_xte)
cl_new_poly = cool_poly.transform(cl_xte)
cl_new_poly2 = cool_poly2.transform(cl_xte)
```

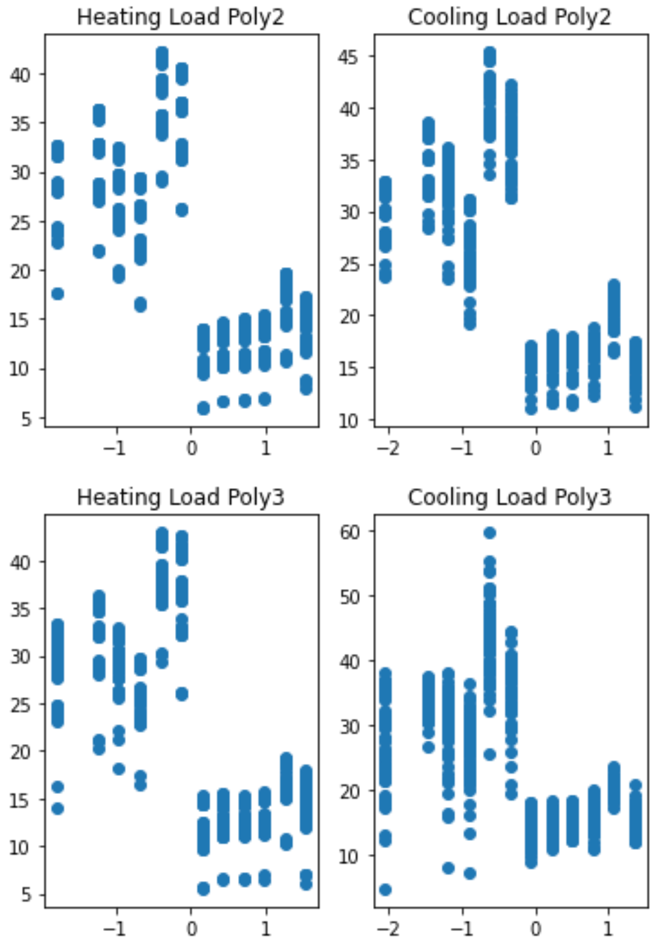
In [17]:

```
hl_poly_pred = hl_lr2.predict(hl_new_poly)
hl_poly_pred2 = hl_lr3.predict(hl_new_poly2)
cl_poly_pred = cl_lr2.predict(cl_new_poly)
cl_poly_pred2 = cl_lr3.predict(cl_new_poly2)
```

In [18]:

```
plt.subplot(1,2,1)
plt.title("Heating Load Poly2")
plt.scatter(hl_xte[:,1], hl_poly_pred)
plt.subplot(1,2,2)
plt.title("Cooling Load Poly2")
plt.scatter(cl_xte[:,1], cl_poly_pred)
plt.show()

plt.subplot(1,2,1)
plt.title("Heating Load Poly3")
plt.scatter(hl_xte[:,1], hl_poly_pred2)
plt.subplot(1,2,2)
plt.title("Cooling Load Poly3")
plt.scatter(cl_xte[:,1], cl_poly_pred2)
plt.show()
```



In [19]:

```
hl_poly_scores = cvs(hl_lr2, hl_xtr, hl_ytr, scoring='neg_mean_absolute_error', n_jok
print('Heating Load Mean and Standard Deviation(2-degree): %.3f (%.3f)' % (np.mean(hl
print("Heating Load MSE(2-degree): %0.2f" % MSE(hl_yte, hl_poly_pred))
print("Heating Load MAE(2-degree): %0.2f" % MAE(hl_yte, hl_poly_pred))
hl_poly2_scores = cvs(hl_lr3, hl_xtr, hl_ytr, scoring='neg_mean_absolute_error', n_jc
print('Heating Load Mean and Standard Deviation(3-degree): %.3f (%.3f)' % (np.mean(hl
print("Heating Load MSE(3-degree): %0.2f" % MSE(hl_yte, hl_poly_pred2))
print("Heating Load MAE(3-degree): %0.2f" % MAE(hl_yte, hl_poly_pred2))
cl_poly_scores = cvs(cl_lr2, cl_xtr, cl_ytr, scoring='neg_mean_absolute_error', n_jok
print('Cooling Load Mean and Standard Deviation(2-degree): %.3f (%.3f)' % (np.mean(cl
print("Cooling Load MSE(2-degree): %0.2f" % MSE(cl_yte, cl_poly_pred))
print("Cooling Load MAE(2-degree): %0.2f" % MAE(cl_yte, cl_poly_pred))
cl_poly2_scores = cvs(cl_lr3, cl_xtr, cl_ytr, scoring='neg_mean_absolute_error', n_jc
print('Cooling Load Mean and Standard Deviation(3-degree): %.3f (%.3f)' % (np.mean(cl
print("Cooling Load MSE(3-degree): %0.2f" % MSE(cl_yte, cl_poly_pred2))
print("Cooling Load MAE(3-degree): %0.2f" % MAE(cl_yte, cl_poly_pred2))
# score functions
print("Heating Load Prediction Scoring(2-degree): %0.2f" % hl_lr2.score(hl_new_poly,
print("Cooling Load Prediction Scoring(2-degree): %0.2f" % cl_lr2.score(cl_new_poly,
print("Heating Load Prediction Scoring(3-degree): %0.2f" % hl_lr3.score(hl_new_poly2,
print("Cooling Load Prediction Scoring(3-degree): %0.2f" % cl_lr3.score(cl_new_poly2,
# r2 score
print("Heating R2_score(2-degree): %0.5f" % r2s(hl_yte, hl_poly_pred))
print("Cooling R2_score(2-degree): %0.5f" % r2s(cl_yte, cl_poly_pred))
print("Heating R2_score(3-degree): %0.5f" % r2s(hl_yte, hl_poly_pred2))
print("Cooling R2_score(3-degree): %0.5f" % r2s(cl_yte, cl_poly_pred2))
```

```
Heating Load Mean and Standard Deviation(2-degree): -2.399 (0.390)
Heating Load MSE(2-degree): 0.67
Heating Load MAE(2-degree): 0.63
Heating Load Mean and Standard Deviation(3-degree): -2.399 (0.390)
Heating Load MSE(3-degree): 0.61
Heating Load MAE(3-degree): 0.56
Cooling Load Mean and Standard Deviation(2-degree): -2.316 (0.348)
Cooling Load MSE(2-degree): 3.92
```

```
Cooling Load MAE(2-degree): 1.31
Cooling Load Mean and Standard Deviation(3-degree): -2.316 (0.348)
Cooling Load MSE(3-degree): 14.75
Cooling Load MAE(3-degree): 2.36
Heating Load Prediction Scoring(2-degree): 0.99
Cooling Load Prediction Scoring(2-degree): 0.96
Heating Load Prediction Scoring(3-degree): 0.99
Cooling Load Prediction Scoring(3-degree): 0.84
Heating R2_score(2-degree): 0.99338
Cooling R2_score(2-degree): 0.95738
Heating R2_score(3-degree): 0.99399
```

Despite the split of 20:80 for the train_test_split function, the low variance still made it difficult as we see with polynomial regression where the prediction score reaches 99% for 3-degree polynomial prediction. Regardless, however the best model of the three i've tried is Ridge Regression with the fact that its the lowest score wise, but still the better choice as the my goal was to stay in the 80-90% range which is best vs having 90-100% which may sound good but I would imagine shows a clear sign the model is over fitted, which truth be told is difficult with this low variance.

In [20]:

```
print("range of heating load: %0.4f - %0.4f" % (np.min(heat), np.max(heat)) )
print("range of cooling load: %0.4f - %0.4f" % (np.min(cool), np.max(cool)) )
```

```
range of heating load: 6.0100 - 43.1000
range of cooling load: 10.9000 - 48.0300
```