CS5014 - Practical 1:
Predicting the energy performance of residential buildings

Student ID: 180029539

Problem:

Given, the data set of room properties like surface area, wall area, etc.., we are asked to predict the respective heating and cooling loads for that room.

1) Data Preparation:

We split these parts into multiple tasks. First, I opened the data and saw that all the data columns are numerical and no data encoding is required. So, the next step is to split up the training and testing set. I also extract a validation set just in case some algorithms require it. I used as 70-20-10 split.

Next, after splitting the data, we look at the inputs and output column statistics. Given below are the statistics of individual columns:

	X1	X2	X3	X4
count	537.000000	537.000000	537.000000	537.000000
mean	0.761993	673.544693	319.458101	177.043296
std	0.105510	87.930746	44.271567	45.291630
min	0.620000	514.500000	245.000000	110.250000
25%	0.660000	612.500000	294.000000	122.500000
50%	0.740000	686.000000	318.500000	220.500000
75%	0.820000	759.500000	343.000000	220.500000
max	0.980000	808.500000	416.500000	220.500000

	X5	X6	X7	X8
count	537.000000	537.000000	537.000000	537.000000
mean	5.305400	3.513966	0.239292	2.795158
std	1.750754	1.108099	0.132646	1.548632
min	3.500000	2.000000	0.000000	0.000000
25%	3.500000	3.000000	0.100000	1.000000
50%	7.000000	4.000000	0.250000	3.000000
75%	7.000000	4.000000	0.400000	4.000000
max	7.000000	5.000000	0.400000	5.000000

	Y1	Y2
count	537.000000	537.000000
mean	22.544190	24.832272
std	10.119578	9.607369
min	6.010000	10.900000
25%	13.010000	15.730000
50%	19.360000	24.610000
75%	32.060000	33.230000
max	42.960000	48.030000

From these numbers, I understood that there are no missing values and the data looks to be normally distributed. To confirm this I draw scatter plots of all inputs with each of the

outputs in the next section. But, as some of the columns have a range of 514 to 800 and some have a range of 0.6 to 0.9. It is a good idea to normalise the inputs. Statistics after scaling are given below.

	X1	X2	X3	X4
count	537.000000	537.000000	537.000000	537.000000
mean	0.001033	0.874988	0.420484	0.227252
std	0.000275	0.019524	0.061820	0.040602
min	0.000678	0.835963	0.321894	0.154808
25%	0.000840	0.862761	0.373556	0.199099
50%	0.000972	0.872815	0.436392	0.218204
75%	0.001162	0.890806	0.464583	0.258621
max	0.001626	0.901338	0.526366	0.289716

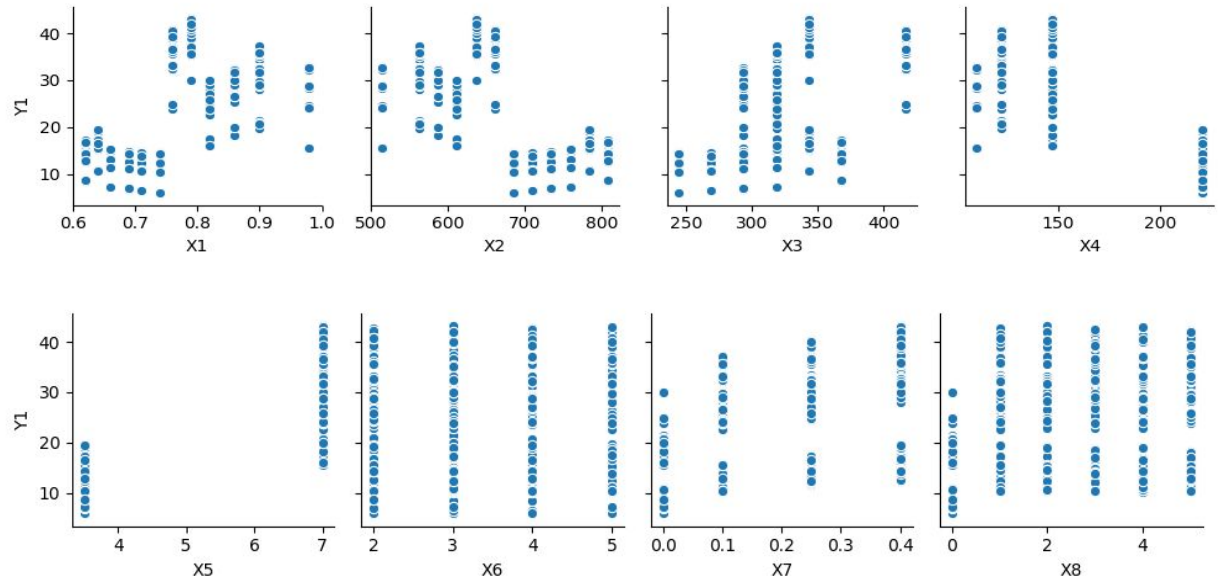
	X5	X6	X7	X8
count	537.000000	537.000000	537.000000	537.000000
mean	0.007111	0.004641	0.000309	0.003703
std	0.003059	0.001627	0.000184	0.002102
min	0.003825	0.002186	0.000000	0.000000
25%	0.004259	0.003278	0.000131	0.002186
50%	0.004599	0.004554	0.000316	0.003651
75%	0.010390	0.005937	0.000469	0.005418
max	0.011613	0.008295	0.000664	0.008294

From the above statistics, we can see that all the values are in a standard range. Next, we visualise the data using scatter plots to better understand the input and output properties.

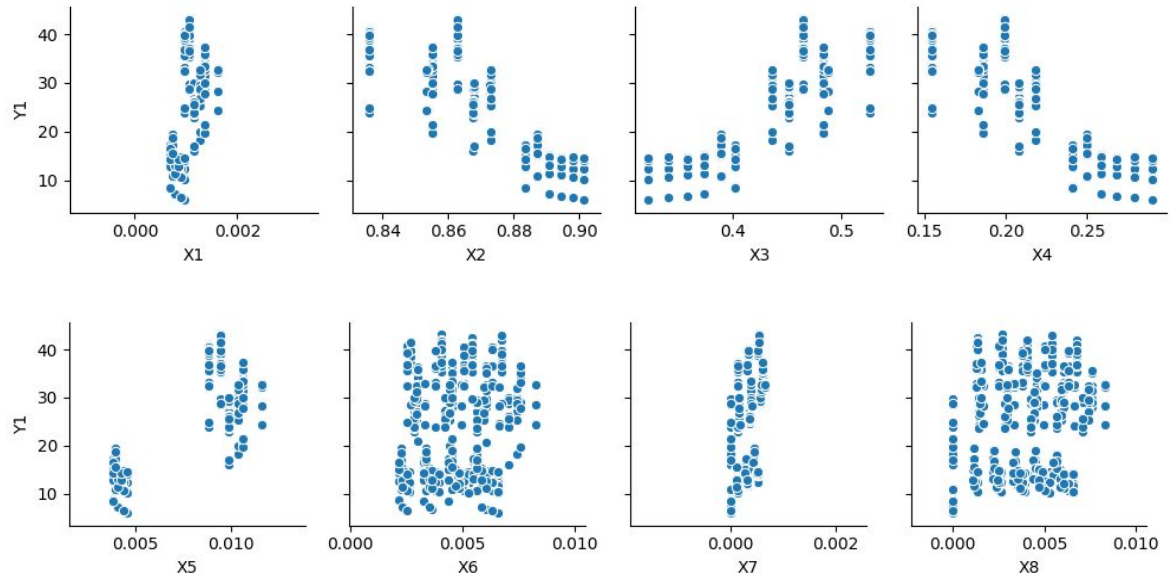
(Continued on next page)

2) Data Visualization and Analysis:

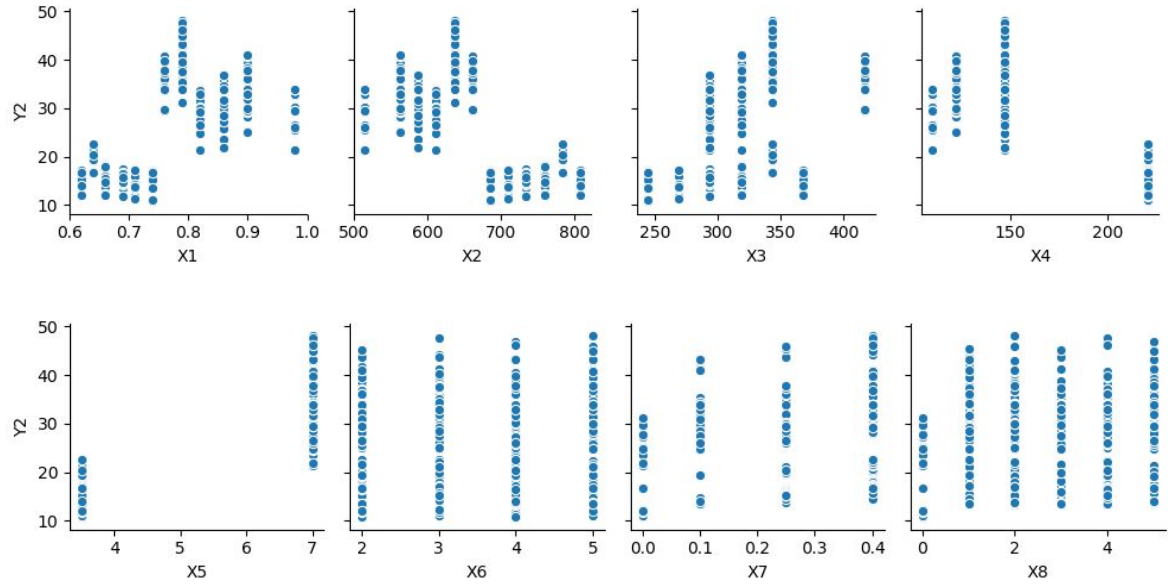
Scatter Plots of inputs vs heating load output (before normalizing):



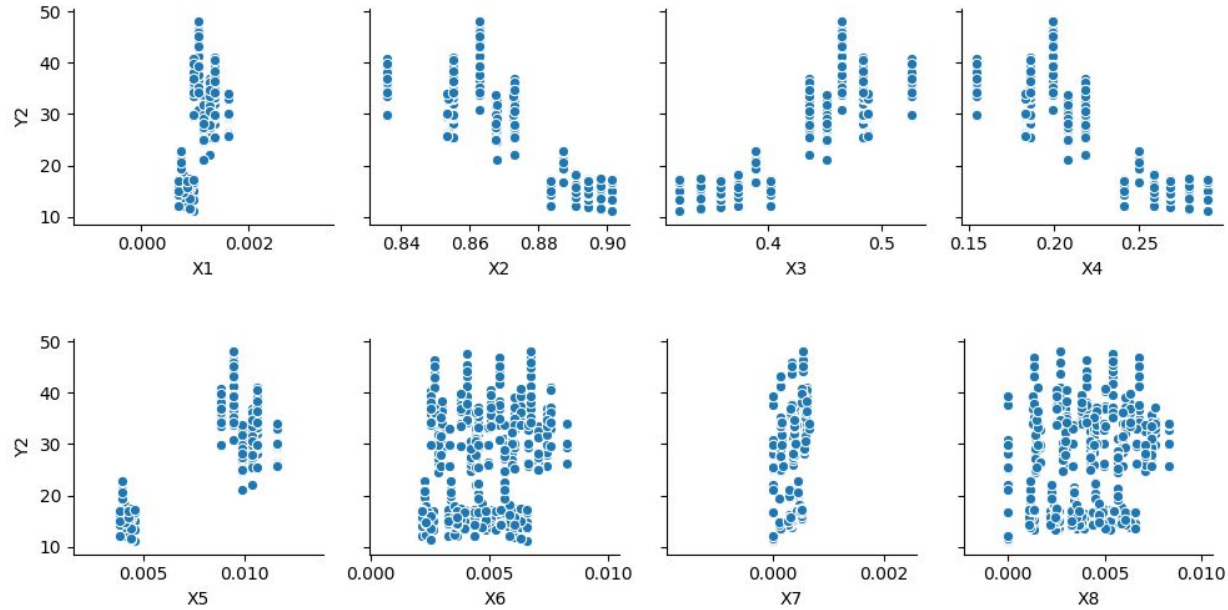
Scatter Plots of inputs vs heating load output (after normalizing):



Scatter Plots of inputs vs cooling load output (before normalizing):



Scatter Plots of inputs vs cooling load output (after normalizing):



We can see that the data is not normalised, and we can also see the use of scaling from the scatterplots. The x-axis of the normalised scatter plots are more similar.

(Continued on the next page)

3) Feature Selection:

To understand the importance of features, we can look at the correlation matrix.

Correlation Matrix:

From the correlation matrix, we understand that the inputs ‘X1’ to ‘X5’ are highly correlated with each other. But from the scatterplots, we can see that our data has a multicollinearity problem. Also, the high collinearity of greater than 0.9 between features also confirms the multicollinearity. So, I will retain all the input features.

4) Model Selection:

Based on our previous observations, as data is not normally distributed, a linear regression might not be an ideal choice. But, let’s implement a linear regression model to test the accuracy and later compare it with other models.

Linear Regression:

I have implemented a linear regression model, using sklearn libraries. First, I have trained the regression model using my scaled data inputs and normal outputs. Then, I test the model with the outputs that I separated at the beginning of the code. Then, I print out the R-squared value, Mean Squared Error(MSE), and Root Mean Squared Error(RMSE).

Polynomial Regression:

Next, I have used polynomial regression as the error was higher in linear regression. As we have 8 features, I thought it would be helpful to have a curve rather than a line to predict the heating and cooling loads. Then, I print out the R-squared, Mean Squared Error(MSE), and Root Mean Squared Error(RMSE).

Lasso Regression:

Final regression model I tried was Lasso Regression. The reason I selected Lasso was because of the multicollinearity of our data. I used the Lasso Regression from SKLearn libraries for implementation. I have used the alpha value of 0.002 after few trials.

5) Evaluating the performance of our Models:

Let us now see the performance across different models. We will see why in the next section.

Regression Model	R-Squared Value	MSE	RMSE
Linear Regression model	90%	9.5	3.0
Polynomial Regression of degree 2	98%	1.5	1.2
Lasso Regression of alpha 0.00005	90%	10	3.2
Ridge Regression Of alpha 0.0000001	90%	9.5	3.1

6) Critical Analysis of Results and Approach:

Now, let us look at why different algorithms gave different accuracies. First, the linear regression gave a very low r-squared score as 90% and rmse of 3.0. It is due to the fact that our data has multi-collinearity. The difference between MSE and RMSE is that RMSE is given in y units. It makes it easier to visualise the difference between the actual and predicted value. Next, the reason polynomial regression works so well is due to the fact that due to curves that the polynomial regression uses to fit the model. Using Lasso Regression helps to overcome the multicollinearity problem. If you alter the alpha value, you can also regularise and understand the importance of features. When you start increasing the alpha parameter, the model starts penalizing the coefficients of features that are not impacting the output as much as the others. It makes the coefficients zero as they become less and less important. The only difference in Ridge from Lasso is it makes the coefficients very small but not zero. So, if you observe the coefficients in Ridge, even the largest alpha values, don't make the coefficients zero. But, in lasso you can see that coefficient of X4 becomes zero at alpha = 0.00005.

Mistakes I made and corrected that helped better understand the process:

- Using of standardization instead of normalization affects the model by a huge extent as standardization only works better for normal distributions. Hence, updated the use of scale function with normalise function.
- Removing features based on correlation matrix values. As, the correlation values are highly impacted by the multicollinearity.

Improvements to be made:

- I have experimented with stepwise regression using forward selection. It keeps adding features that improve the r-square value of the model. But, it was taking a long time to run in some cases. If could implement, it will help solve multicollinearity problem by reducing the correlated features
- Implementation of K-fold cross validation or Random forest regressor would help randomize the training set and also randomize the process to reduce overfitting.