Estimation of cooling load of buildings

Group 8

Abstract

We develop a model based on linear regression to study the effect of eight input variables (relative compactness, surface area, wall area, roof area, overall height, orientation, glazing area, glazing area distribution) on cooling load, of residential buildings. We systematically investigate the association strength of each input variable with each of the output variables using a variety of linear and non-parametric regression methods.

We propose a polynomial regression model and a regression tree to reliably predict the cooling load for a given setting of predictor variables.

1. Introduction

One of the factors that affects energy efficiency of a building is its cooling load. Many countries require to give an estimate of the Cooling load of a building so as to accommodate the energy needs in the planning process. Building dimensions mainly affect the cooling load, they include Relative compactness,surface area, roof area, Height,Orientation,Glazing area and Glazing variations. In order to build energy efficient buildings, this study analyses the effects of various properties of buildings to predict the cooling load incurred by those factors. The results can be used to come up with building specifications that yield less cooling load and in turn less energy consumption. We try to explain how important each dimension is in explaining the cooling load of the building. We also try to estimate the mean response so that it can be used as a benchmark.

2. Data

The dataset contains 768 observations with different settings of dependent variable cooling load and independent variables described below. We use 80% of the data for training, 10% of the data for validation and 10% of the data for testing. The training data is used to determine the coefficients for the model. Using the validation test we estimate the parameters to select the final model and eventually we test the predictive performance of the model using the testing set.
Summary of Independent Variables:
X1 -(Relative Compactness- No units),
X2 - Surface Area – m²,
X3 - Wall Area – m²,
X4 - Roof Area – m²,
X5 - Height – m,
X6 -,Orientation - 2:North, 3:East, 4:South, 5:West - No units,
X7 - Glazing Area - 0%, 10%, 25%, 40% (of floor area) - No units,
X8 - Glazing Variations - 1:Uniform, 2:North, 3:East, 4:South, 5:West,Y - Cooling Load – kWh/m².

X6, X8 are categorical variables whereas, other variables are continuous variables. The continuous variables in the dataset have very short ranges.

From the correlation matrix below, we can see that there is very high correlation between some of the variables. This shows that non-additive effects exist in the dataset. Moreover, the scatter matrix below also shows that the data is not normally distributed.

Table 2.1

	X1	X2	X3	X4	X5	X6	X7	X8	Y
X1	1.0000	-0.9919	-0.2038	-0.8688	0.8277	0.0000	0.0000	0.0000	0.6343
X2	-0.9919	1.0000	0.1955	0.8807	-0.8581	0.0000	0.0000	0.0000	-0.6730
X3	-0.2038	0.1955	1.0000	-0.2923	0.2810	0.0000	0.0000	0.0000	0.4271
X4	-0.8688	0.8807	-0.2923	1.0000	-0.9725	0.0000	0.0000	0.0000	-0.8625
X5	0.8277	-0.8581	0.2810	-0.9725	1.0000	0.0000	0.0000	0.0000	0.8958
X6	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0143
X7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.2130	0.2075
X8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2130	1.0000	0.0505
Y	0.6343	-0.6730	0.4271	-0.8625	0.8958	0.0143	0.2075	0.0505	1.0000

Plots also indicate high multi collinearity between X1,X2 with almost linear dependence,X4 is peculiar for it has a variance from some range of values,while it has perfect dependence for others.This is going to be problem because,R might find that there is a full dependence and drop it but since there is variance for some values, they might be unaccounted and thus our residuals.This can be solved by inducing a logical vlaue for X4,for example asking R to consider only those values where X4<122.5.In literature this is known as “Piecewise regression”.

X6 is orientation of building and X8 is Glazing variations,both are categorical in nature.But for any given X6, X1-X8 have almost similar range.X8 has to be seen from a microscope to understand it better,0 in X8 may be indication that there is no glazing variation,for X8==0, X7 is 0,confirming our understanding.Also the sequencing of X8 is north,east,south,west, the order of sunlight entering the building is also same,so we might well use X8 as continous rather than categorical.

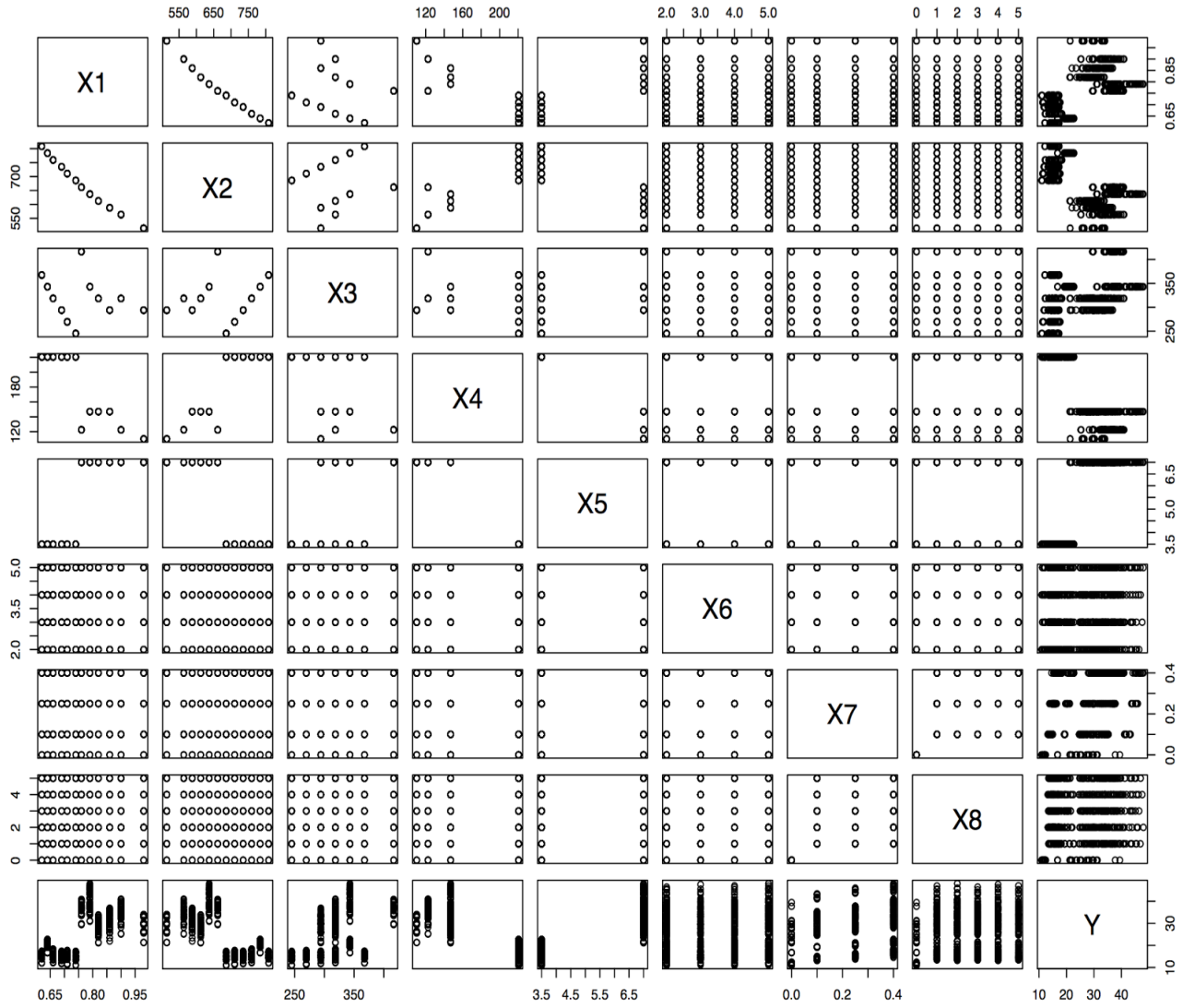
X7 has zero in it,and when X7 is zero it’s range with other predictor variable is different compared to when it is not.With Trial and Error we found out that X7 is massively inflating the error terms at its tails in a normal plots,various transformation did not work,nor using X7==0 in the model,while dropping would work with little reduction in prediction ability.

3. Methods

3.1 Plots and Figures

We construct different figures and tables to perform the preliminary exploratory data analysis. Preliminary data analyses include, scatter plot matrix which hints at high multi col-linearity among some variables. For instance, X1 and X2 are highly linearly correlated and this is backed up by the Pearson correlation matrix. It is also evident that the values are clustered around nearby values which stems from the short ranges in the predictor variables. For example, X5 takes only two values, therefore all the corresponding values of other variables are only available for those two values. The correlation matrix and the scatter matrix also reveal some of the less influential variables like X6 and X8 which are qualitative in nature. The scatter matrix also indicates that there is no variability introduced when X6 and X8 are considered.

Figure 3.1.1



We also use different residual plots to evaluate the regression model and determine different patterns in the data. For example, Residual vs fitted plots helps us in identifying if the constancy of error terms is maintained. The normal plot of the residuals helps us examine the normality of the error terms. Using these plots we can also decide what remedial procedures to follow. For example, witnessing an increasing trend in the residuals vs fitted values plot indicates using weighted regression.

Figure 3.1.2

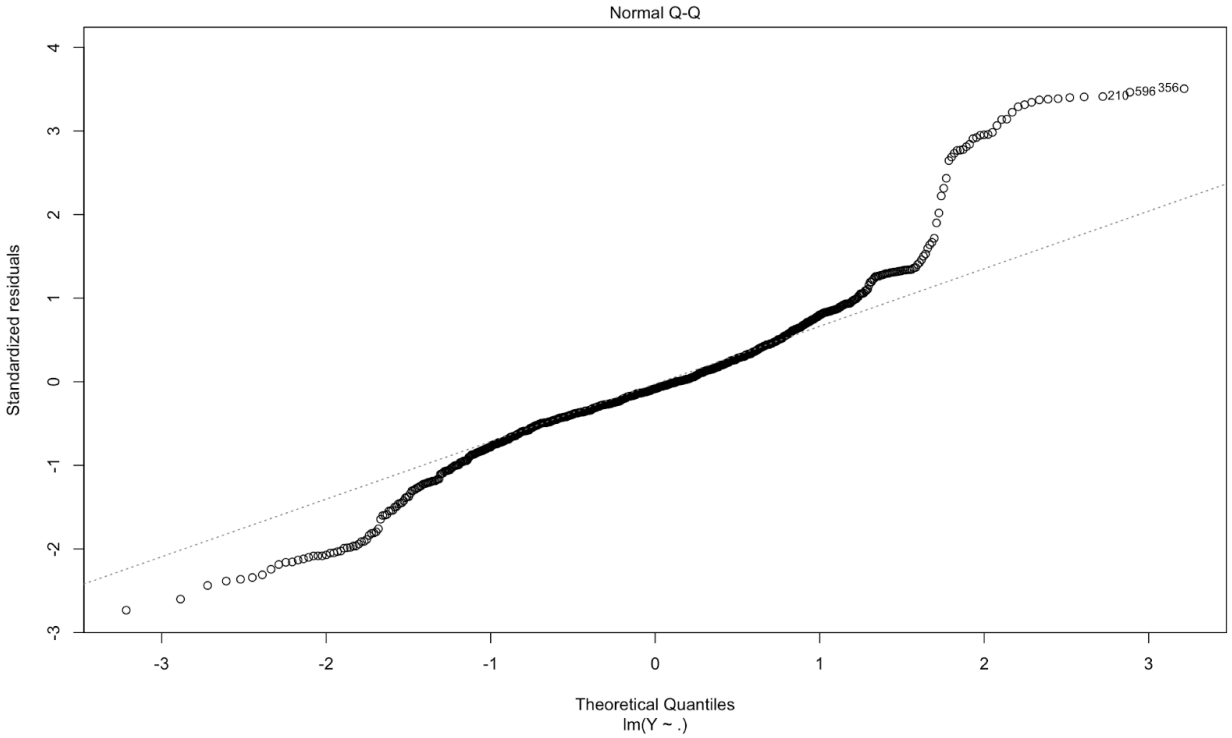
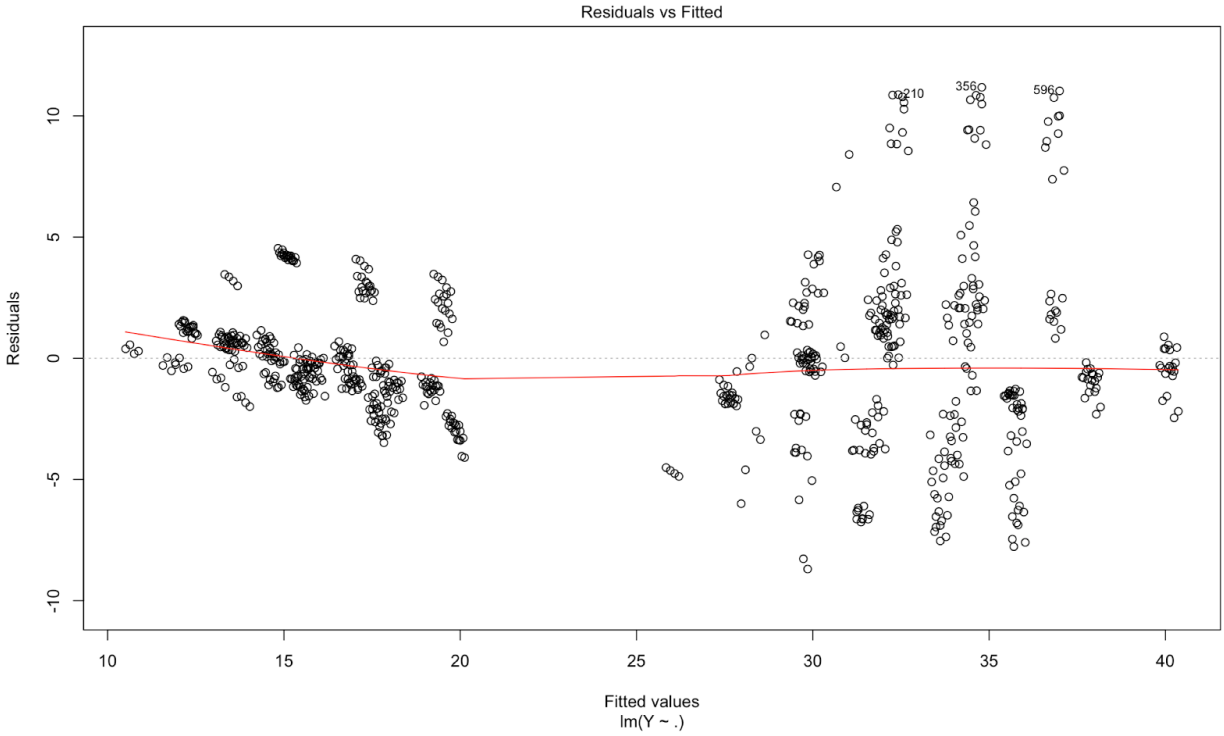


Figure 3.1.3



For example, from the residuals plots of the linear model containing all the predictors we can easily interpret that the error terms are neither constant nor normal.

3.2 Multiple Linear Regression

As specified in the data section, we split the data into three sets. We start by building a linear regression model containing all the variable and keep on improving using various model selection methods like stepwise regression and best subsets method. As described in the plots and figures section, we also analyze the residual plots to determine the remedial procedure. Transformation to the response variable is applied and we try different higher order terms in the model.

Due to high interaction effects we propose the following regression model:

$$Y = \beta_0 + \beta_1 X1^1 + \beta_2 X1^2 + \beta_3 X1^3 + \beta_4 X1^4 + \beta_5 X2^1 + \beta_6 X2^2 + \beta_7 X3^1 + \ldots + \beta_k X3^\lambda + \beta_{k+1} X5 + \beta_{k+2} X8$$

We estimate the lambda value by validating on the validation set. The results for the validation are shown below. Here, lambda = 2 seems to be explaining the model better hence we choose it as our final model. The final test is performed on this model using the test set.

Lambda	F-statistic	Adjusted R-Squared	RMSE	Prdicted R-Squared
2	598.9	0.907	2.91	0.9049
1	520	0.8944	2.99	0.8922

Table 3.2.1

3.3 Tree Regression

As seen in the previous section, we obtain a polynomial regression model which gives us a formula to predict the values. These kinds of models are not appropriate when there is high interaction effect between predictors. It is evident from the scatter plot matrix in figure 3.1.1 that there is a high interaction effect among predictors. Moreover the linear relationship between the predictors and the response variable is not trivial. To alleviate these issues, using a non-parametric model seems to be an obvious choice. We choose regression tree as the non-parametric model.

Regression tree is built by growing a binary tree by introducing predictor variables as nodes. At each node, the complexity parameter is calculated which represents the importance of introducing that variable in the model. In this method, the data space is divided into small chunks and each chunk is represented by a simple model which is unlike the regression models where just one formula describes the data space globally. The recursive division of the space is called recursive partitioning.

Figure 3.3.1

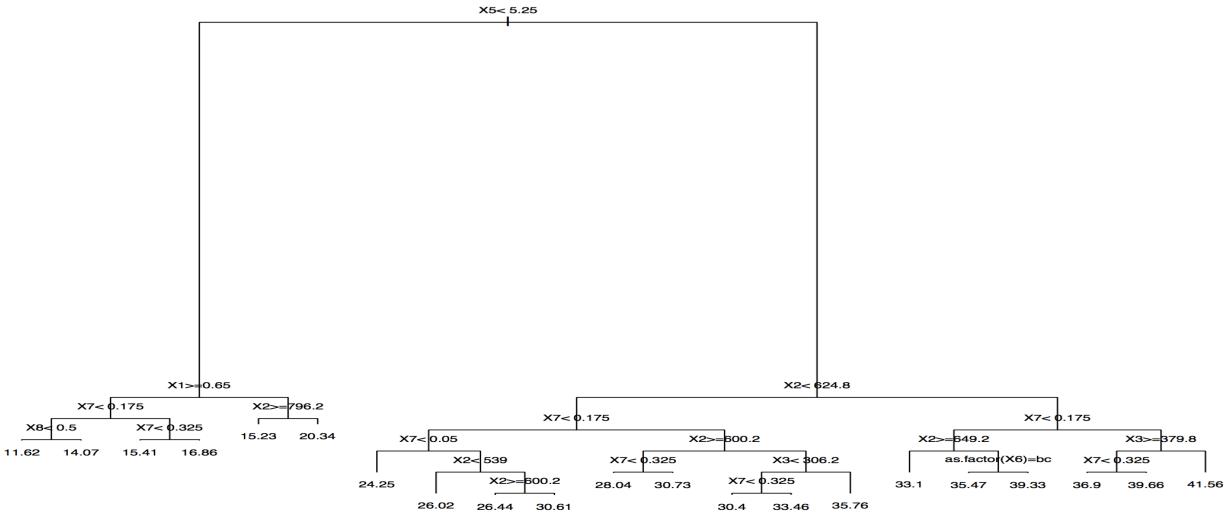


Figure 3.3.1 shows the regression tree we built for the building data. We have 22 terminals in the data. We use the complexity parameter 0.001 as the cutoff value, which means the partition stops when the value of complexity parameter drops lower than 0.001 at any node. Generally, the cutoff value is 0.01 and decreasing this value results in more branches in the tree, which may lead to overfitting. But for this data we use test errors to choose appropriate cutoff value. The rules specified at each node are easily interpretable but they can get complicated. For this purpose, we attempt to prune the tree by using the minimum deviance, but for this particular data using the cutoff 0.001 yields the optimum number of terminals and therefore pruning the tree yields the same regression tree as before.

3.3.1 Model Selection:

We use the same 3 splits to train, validate and test the regression trees. To select the best model, we try 3 different parameter values for complexity parameter as shown in the figure below. We choose the complexity parameter 0.001 as the final parameter, since it gives the best performance.

Table 3.3.1.1

CP	RMSE
0.01	3.21
0.001	1.918

3.4 Model Validation and Evaluation

After training the model using the training set and selecting appropriate models using the validation set, the models are evaluated to check their predictive accuracies on unseen data having similar characteristics.

To evaluate the performance of each model we calculate the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). The Root Mean Squared error is given as:

$$RMSE=\sqrt{\frac{1}{T}\left(\sum_{i=1}^{T}\left(Y-\hat{Y}\right)^{2}\right.}$$

The Mean Absolute Error can be given as:

$$MAE = \frac{1}{T}\sum_{i=1}^{T} |Y - \hat{Y}|$$

Where T is the number of observations in the Test set. $\hat{Y}$ is the mean response predicted by the given model. Y is the actual value of the response variable in the test set. We predict the cooling load for the test data and report the error values for only the test data.

4. Results

After selecting the model parameter for the polynomial regression using the validation set and building the regression tree on the training set, we now proceed by making predictions on the out of sample data and compare the performances of both the models.

4.1 Prediction Interval

We use the mean values for all the variables in the model and find its prediction interval for the regression model. The estimated prediction interval is: [50.01764, 63.31356]. The standard error is 6.64796.

4.2 RMSE and MAE for Tree and Regression Models

	RMSE	MAE
Tree	2.03	1.40
Regression	2.64	2.16

As discussed in the methods section, we calculate the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for both models. The test errors for the regressions indicates that we can rely on the polynomial regression model to make the predictions of cooling load. As expected, regression trees perform better on the test data due to prevailing multicollinearity in the data.

The importance of variables obtained after building the regression trees also bolster the multicollinearity in the data. The importance for X1 and X2 and X4 and X5 seems to be equal. We suspect that this stems from the very high correlation between the two variables in each pair. Incidentally, this collinearity is used by the regression models to make better predictions, whereas it makes the regression model more complex.

Table 4.2.1

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-9.8061473	6.04447744	-1.622332	1.052550e-01
poly(X1, 4)1	4346.9244626	243.74027188	17.834248	1.879074e-57
poly(X1, 4)2	2043.2044598	120.68696668	16.929786	6.869156e-53
poly(X1, 4)3	-445.5687175	27.09550026	-16.444381	1.785894e-50
poly(X1, 4)4	162.4412203	10.74242878	15.121461	4.924203e-44
poly(X2, 2)1	4127.5740780	246.25994929	16.761045	4.778305e-52
poly(X2, 2)2	-2675.2067577	145.59685291	-18.374070	3.280950e-60
poly(X3, 2)1	67.6308968	20.49252871	3.300271	1.022897e-03

poly(X3, 2)2	-89.0981478	8.22984048	-10.826230	4.410314e-25
X5	6.4129938	1.16265203	5.515832	5.155936e-08
X8	0.2883825	0.07583559	3.802733	1.576734e-04

The t-value and P-value suggest that there is a linear association between the dependent and predictor variables

4.3 Variable Importance in Regression Tree

Figure 4.3.1

X1	X2	X4	X5	X3	X7	X8
23	23	21	21	9	1	1

5. DISCUSSION

The preliminary statistical analysis of the data reveals that the linear relationship between the response variable and the predictors is very unusual. Moreover, the data suffers from multicollinearity making it even more challenging to satisfy the assumptions of a linear model. For example, the full linear model containing all the predictors has a S-curved normality plot for error terms. Attempting to model a linear regression formula for such data helped us gain deep insights about various remedial procedures. For example, we tried weighted regression when the residual vs fitted plot resembled a megaphone shape for one of the intermediate models, but the resulting model with the residuals from the linear model of absolute residuals and predicted Y values as weights did little to make the residuals constant. We speculate that this was because of the non-additive effects within the model.

We also used various model selection methods like stepwise regression and best subsets method, but were able to get the final model by analyzing the residual plot which indicated the presence of higher order terms in the model. While we understand that higher order terms can complicate our model and can easily lead to extrapolation, it has become necessary in our case.

Although the resulting polynomial model can be used reliably to make predictions, there are some restrictions to the model. The model can not be used to make predictions for wide range of values since the data used for model construction has very short range of values. Moreover, in the process of moulding the data to comply to the linear regression assumptions, we lose a lot of interpretability and generality of the model. This may be because of regression model's inability to model complex relationships.Also the high multi collinearity problem, although literature says that it does not affect the predictability of a model,it becomes hard to estimate the the coefficients, reason why we did not go with subset or forward regression methods, standardization did not help, dropping some variables will help but that makes the model very localized, hence we need to look at other areas like non-parametric regression.There is no point in torturing the data to get to the result,eventually it will fail,so its better to change the method of problem solving.

Therefore, we propose regression trees which seems like a much more flexible and interpretable method for the data we are dealing with. We do try the pruning methods to get the optimal number of terminals to attain a less complex tree model.

We can potentially improve the tree model by trying various other non-parametric methods.

Reference

1. **http://mlr.cs.umass.edu/ml/datasets/Energy+efficiency**
2. http://www.stat.cmu.edu/~cshalizi/350/lectures/22/lecture-22.pdf

Appendix

30 November 2016

```
# Load Data
require("XLConnect")
```

```
## Loading required package: XLConnect
```

```
## Loading required package: XLConnectJars
```

```
## XLConnect 0.2-12 by Mirai Solutions GmbH [aut],
##   Martin Studer [cre],
##   The Apache Software Foundation [ctb, cph] (Apache POI, Apache Commons
##     Codec),
##   Stephen Colebourne [ctb, cph] (Joda-Time Java library),
##   Graph Builder [ctb, cph] (Curvesapi Java library)
```

```
## http://www.mirai-solutions.com ,
## http://miraisolutions.wordpress.com
```

```
require("nortest")
```

```
## Loading required package: nortest
```

```
library(car)
library(tree)
library(rpart)
setwd("~/Documents/CS7280/Project")
data = loadWorkbook("ENB2012_data.xlsx", create = FALSE)
energy_data = readWorksheet(data, sheet = 1)
heating_data = energy_data[, 1:9]
drop = c("Y1")
cooling_data = energy_data[, !(names(energy_data) %in% drop)]

# Rename columns
names(heating_data)[names(heating_data)=="Y1"] = "Y"
names(cooling_data)[names(cooling_data)=="Y2"] = "Y"
cooling = cooling_data
cooling$Y = log(cooling$Y)
cooling$X6 = as.factor(cooling$X6)
```

```
# Fit initial models
heating_model = lm(Y ~ ., data=heating_data)
summary(heating_model)
```

```
##
## Call:
## lm(formula = Y ~ ., data = heating_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8965 -1.3196 -0.0252  1.3532  7.7052
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  84.013418  19.033613   4.414 1.16e-05 ***
## X1          -64.773432  10.289448  -6.295 5.19e-10 ***
## X2           -0.087289   0.017075  -5.112 4.04e-07 ***
## X3            0.060813   0.006648   9.148  < 2e-16 ***
## X4                  NA         NA      NA       NA
## X5            4.169954   0.337990  12.338  < 2e-16 ***
## X6           -0.023330   0.094705  -0.246  0.80548
## X7           19.932736   0.813986  24.488  < 2e-16 ***
## X8            0.203777   0.069918   2.915  0.00367 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.934 on 760 degrees of freedom
## Multiple R-squared:  0.9162, Adjusted R-squared:  0.9154
## F-statistic:  1187 on 7 and 760 DF,  p-value: < 2.2e-16
```

The model has good R-squared value, but it doesn't have a good fit.

Looking at the model summary, (after converting X4 into a factor) it is clear that when X4 is 220.5, X5 invariably remains 3.5.

```
# To illustrate the point below
heating_data[which(heating_data$X4 == 220.5), ]$X5
```

```
##   [1] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
##  [18] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
##  [35] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
##  [52] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
##  [69] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
##  [86] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [103] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [120] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [137] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [154] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [171] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [188] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [205] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [222] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [239] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [256] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [273] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [290] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [307] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [324] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [341] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [358] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
## [375] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
```

So the model automatically drops X4 level 220.5 and X5.

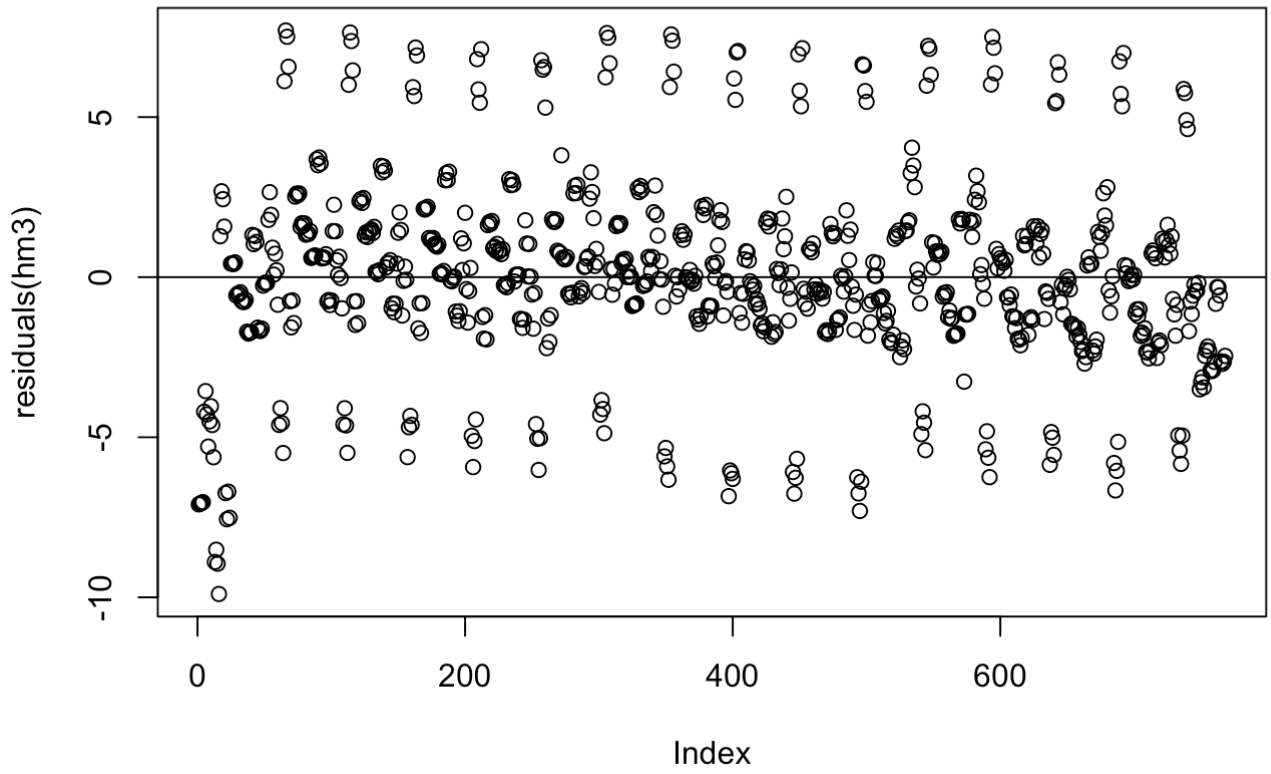
```
# Trying another model
hm2 = lm(Y ~ X1 + X2 + X3 + X4 + X6 + X7 + X8, data=heating_data)
summary(hm2)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X3 + X4 + X6 + X7 + X8, data = heating_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8468 -2.0124 -0.2825  1.4104  9.1552
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  261.37773   13.65753  19.138  < 2e-16 ***
## X1          -145.94342    8.66166 -16.849  < 2e-16 ***
## X2            -0.26248    0.01038 -25.279  < 2e-16 ***
## X3             0.13689    0.00272  50.330  < 2e-16 ***
## X4                  NA         NA      NA       NA
## X6            -0.02333    0.10369  -0.225  0.82204
## X7            19.93274    0.89120  22.366  < 2e-16 ***
## X8             0.20378    0.07655   2.662  0.00793 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.213 on 761 degrees of freedom
## Multiple R-squared:  0.8994, Adjusted R-squared:  0.8986
## F-statistic:  1134 on 6 and 761 DF,  p-value: < 2.2e-16
```

```
hm3 = lm(Y ~ X1 + X2 + X3 + X5 + X6 + X7 + X8, data=heating_data)
summary(hm3)
```

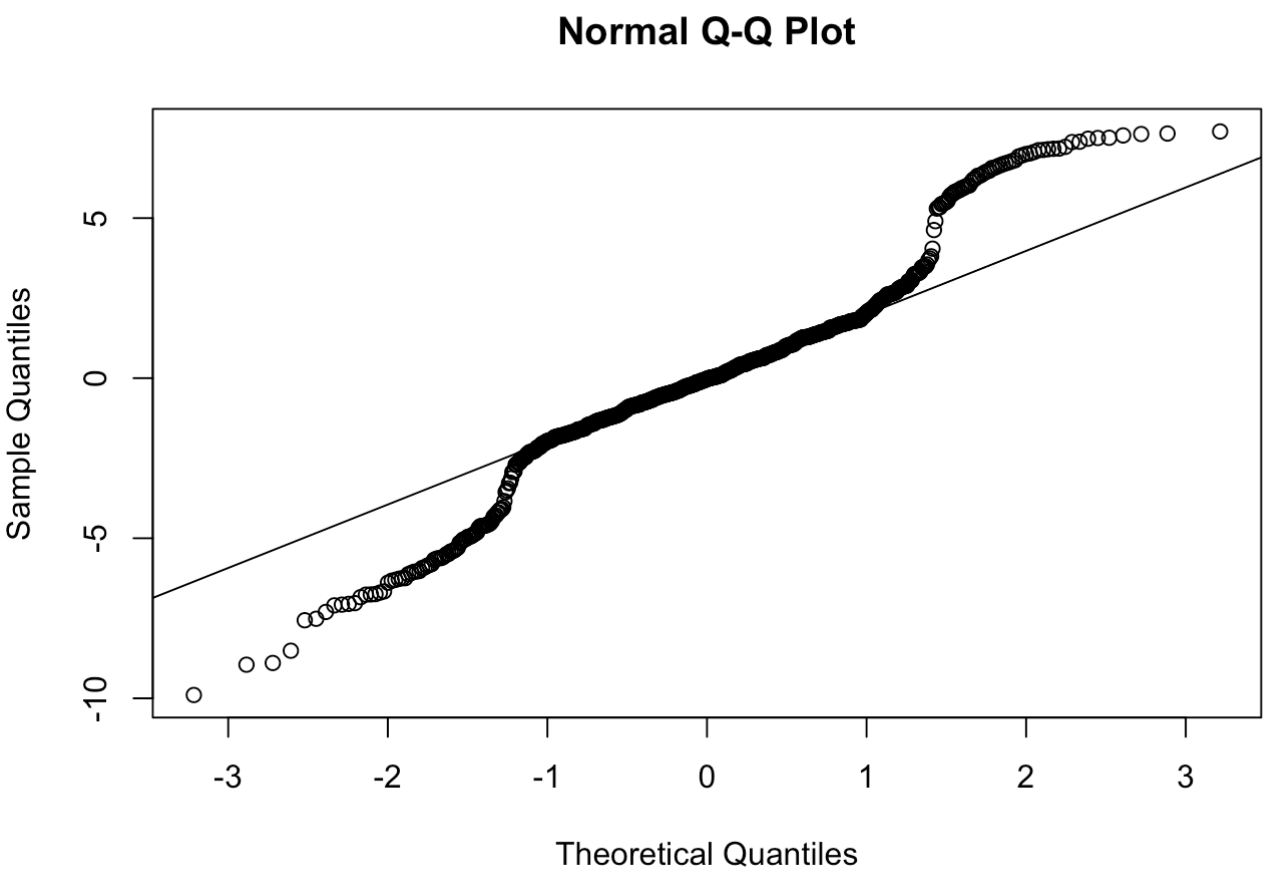
```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X3 + X5 + X6 + X7 + X8, data = heating_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8965 -1.3196 -0.0252  1.3532  7.7052
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  84.013418  19.033613   4.414 1.16e-05 ***
## X1          -64.773432  10.289448  -6.295 5.19e-10 ***
## X2           -0.087289   0.017075  -5.112 4.04e-07 ***
## X3            0.060813   0.006648   9.148  < 2e-16 ***
## X5            4.169954   0.337990  12.338  < 2e-16 ***
## X6           -0.023330   0.094705  -0.246  0.80548
## X7           19.932736   0.813986  24.488  < 2e-16 ***
## X8            0.203777   0.069918   2.915  0.00367 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.934 on 760 degrees of freedom
## Multiple R-squared:  0.9162, Adjusted R-squared:  0.9154
## F-statistic:  1187 on 7 and 760 DF,  p-value: < 2.2e-16
```

```
# Plotting the residuals vs fitted values
plot(residuals(hm3))
abline(0, 0)
```




```
# The plot is not random, it has some patterns. Also has some residual outliers.

# Normal plot of residuals
qqnorm(residuals(hm3))
qqline(residuals(hm3))
```




```
# Error term distribution is not normal
```

Let's try boxcox transformation and find a good lambda value, since the error terms aren't normal.

```
calculate_weights = function(values, model) {
  print(summary(model))
  residual_model = lm(abs(residuals(model)) ~ values)
  plot(values, abs(residuals(model)))
  abline(residual_model)
  return(1 / (residual_model$fitted.values)^2)
}
```

```
#train = cooling_data[sample(nrow(cooling_data), 461), ]

# Partitioning
## 75% of the sample size
smp_size <- floor(0.8 * nrow(cooling_data))

set.seed(123)
train_ind <- sample(seq_len(nrow(cooling_data)), size = smp_size)

train <- cooling_data[train_ind, ]
test <- cooling_data[-train_ind, ]
```

```
df <- cooling_data

fractionTraining   <- 0.80
fractionValidation <- 0.10
fractionTest       <- 0.10

sampleSizeTraining   <- floor(fractionTraining   * nrow(df))
sampleSizeValidation <- floor(fractionValidation * nrow(df))
sampleSizeTest       <- floor(fractionTest       * nrow(df))

indicesTraining    <- sort(sample(seq_len(nrow(df)), size=sampleSizeTraining))
indicesNotTraining <- setdiff(seq_len(nrow(df)), indicesTraining)
indicesValidation  <- sort(sample(indicesNotTraining, size=sampleSizeValidation))
indicesTest        <- setdiff(indicesNotTraining, indicesValidation)

train  <- df[indicesTraining, ]
validation <- df[indicesValidation, ]
test       <- df[indicesTest, ]
```

Calculating Predicted R-Squared

```
pred_r_squared = function(model) {
  pr <- residuals(model)/(1 - lm.influence(model)$hat)
  PRESS <- sum(pr^2)
  tss <- sum(anova(model)$"Sum Sq")
  return(1 - PRESS / (tss))
}

formula = Y ~ X5 + X7 + X3 + X1
log_formula = log(Y) ~ X5 + X7 + X3 + X1
imp_formula = log(Y) ~ X5 + X3 + X8
```

```
# Create model with given sample size
makeModel = function(size) {
  thres = cooling_data[sample(nrow(cooling_data), size), ]
  thres.model = lm(log(Y) ~ X5 + X7 + X3 + X1, data=thres)
  print(summary(thres.model))
  plot(thres.model)
  print(paste("Predicted R-squared", pred_r_squared(thres.model)))
}
```

Model Selection

```
require(leaps)
```

```
## Loading required package: leaps
```

```
# Subsetting method for model selection
summary(regsubsets(log(Y) ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8, data=train, nvmax=2,
 method="backward"))
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
## Subset selection object
## Call: regsubsets.formula(log(Y) ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 +
##     X8, data = train, nvmax = 2, method = "backward")
## 8 Variables  (and intercept)
##    Forced in Forced out
## X1     FALSE      FALSE
## X2     FALSE      FALSE
## X3     FALSE      FALSE
## X5     FALSE      FALSE
## X6     FALSE      FALSE
## X7     FALSE      FALSE
## X8     FALSE      FALSE
## X4     FALSE      FALSE
## 1 subsets of each size up to 3
## Selection Algorithm: backward
##          X1  X2  X3  X4  X5  X6  X7  X8
## 1  ( 1 ) " " " " " " " " "*" " " " " " "
## 2  ( 1 ) " " " " " " " " "*" " " "*" " "
## 3  ( 1 ) " " " " "*" " " "*" " " "*" " "
```

```
summary(regsubsets(Y ~ X1 + X2 + X3 + X4 + X5 + as.factor(X6) + X7 + X8, data=train,
nvmax=2, method="forward"))
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
## Subset selection object
## Call: regsubsets.formula(Y ~ X1 + X2 + X3 + X4 + X5 + as.factor(X6) +
##     X7 + X8, data = train, nvmax = 2, method = "forward")
## 10 Variables  (and intercept)
##                Forced in Forced out
## X1                 FALSE      FALSE
## X2                 FALSE      FALSE
## X3                 FALSE      FALSE
## X5                 FALSE      FALSE
## as.factor(X6)3     FALSE      FALSE
## as.factor(X6)4     FALSE      FALSE
## as.factor(X6)5     FALSE      FALSE
## X7                 FALSE      FALSE
## X8                 FALSE      FALSE
## X4                 FALSE      FALSE
## 1 subsets of each size up to 3
## Selection Algorithm: forward
##          X1  X2  X3  X4  X5  as.factor(X6)3 as.factor(X6)4 as.factor(X6)5
## 1  ( 1 ) " " " " " " " " "*" " "            " "             " "
## 2  ( 1 ) " " " " " " " " "*" " "            " "             " "
## 3  ( 1 ) "*" " " " " " " "*" " "            " "             " "
##          X7  X8
## 1  ( 1 ) " " " "
## 2  ( 1 ) "*" " "
## 3  ( 1 ) "*" " "
```

```
summary(regsubsets(log(Y) ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8, data=train, nvmax=2,
 method="exhaustive"))
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
## Subset selection object
## Call: regsubsets.formula(log(Y) ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 +
##     X8, data = train, nvmax = 2, method = "exhaustive")
## 8 Variables  (and intercept)
##    Forced in Forced out
## X1     FALSE      FALSE
## X2     FALSE      FALSE
## X3     FALSE      FALSE
## X5     FALSE      FALSE
## X6     FALSE      FALSE
## X7     FALSE      FALSE
## X8     FALSE      FALSE
## X4     FALSE      FALSE
## 1 subsets of each size up to 3
## Selection Algorithm: exhaustive
##          X1  X2  X3  X4  X5  X6  X7  X8
## 1  ( 1 ) " " " " " " " " "*" " " " " " "
## 2  ( 1 ) " " " " " " " " "*" " " "*" " "
## 3  ( 1 ) " " " " "*" " " "*" " " "*" " "
```

Cross validation

```
require(caret)
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
train_control <- trainControl(method="cv", number=10, repeats=3)
model = train(Y~X5+(X7==0)+(X7>=0.25)+X3+poly(X1,4)+poly(X2,2), data=cooling, trContr
ol=train_control, method="lm")
print(model)
```

```
## Linear Regression
##
## 768 samples
##   5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 692, 691, 691, 690, 691, 692, ...
## Resampling results:
##
##   RMSE        Rsquared
##   0.09412946  0.9428342
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
##
```

```
rmse <- function(model, data) {
  predictions = predict(model, data)
  return(sqrt(mean((predictions - data$Y) ^2)))
}

mae <- function(model, data) {
  prediction = predict(model, data)
  return(mean(abs(prediction - data$Y)))
}
```

```
# Find bestcp of a tree for pruning it
bestcp = function(tree) {
  return(tree$cptable[which.min(tree$cptable[, "xerror"]), "CP"])
}
```

```
require(rpart)
require(tree)
tree = tree(Y ~ X1 + X2 + X3 + X4 + X5 + as.factor(X6) + X7 + X8, data=train, control=t
ee.control(nrow(train), mindev = 0.001))
summary(tree)
```

```
##
## Regression tree:
## tree(formula = Y ~ X1 + X2 + X3 + X4 + X5 + as.factor(X6) + X7 +
##     X8, data = train, control = tree.control(nrow(train), mindev = 0.001))
## Variables actually used in tree construction:
## [1] "X5"            "X2"            "X7"            "X1"
## [5] "as.factor(X6)" "X3"
## Number of terminal nodes:  22
## Residual mean deviance:  3.177 = 1881 / 592
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -6.90000 -0.67310 -0.03838  0.00000  0.65520  5.69700
```

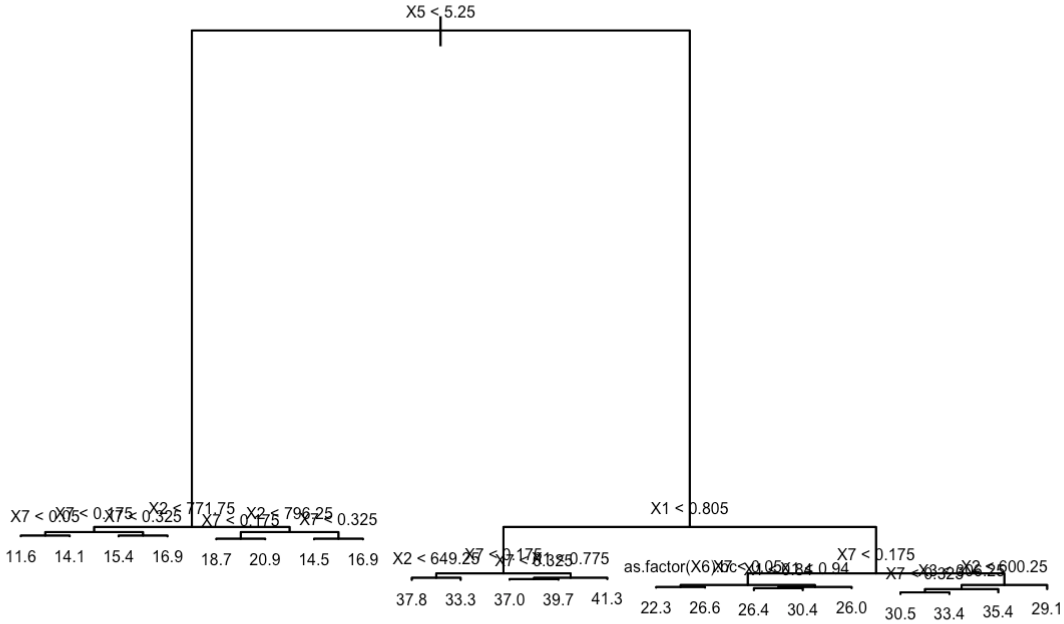
```
print(sqrt(summary(tree)$dev / summary(tree)$df)) # Print RMSE on train
```

```
## [1] 1.782548
```

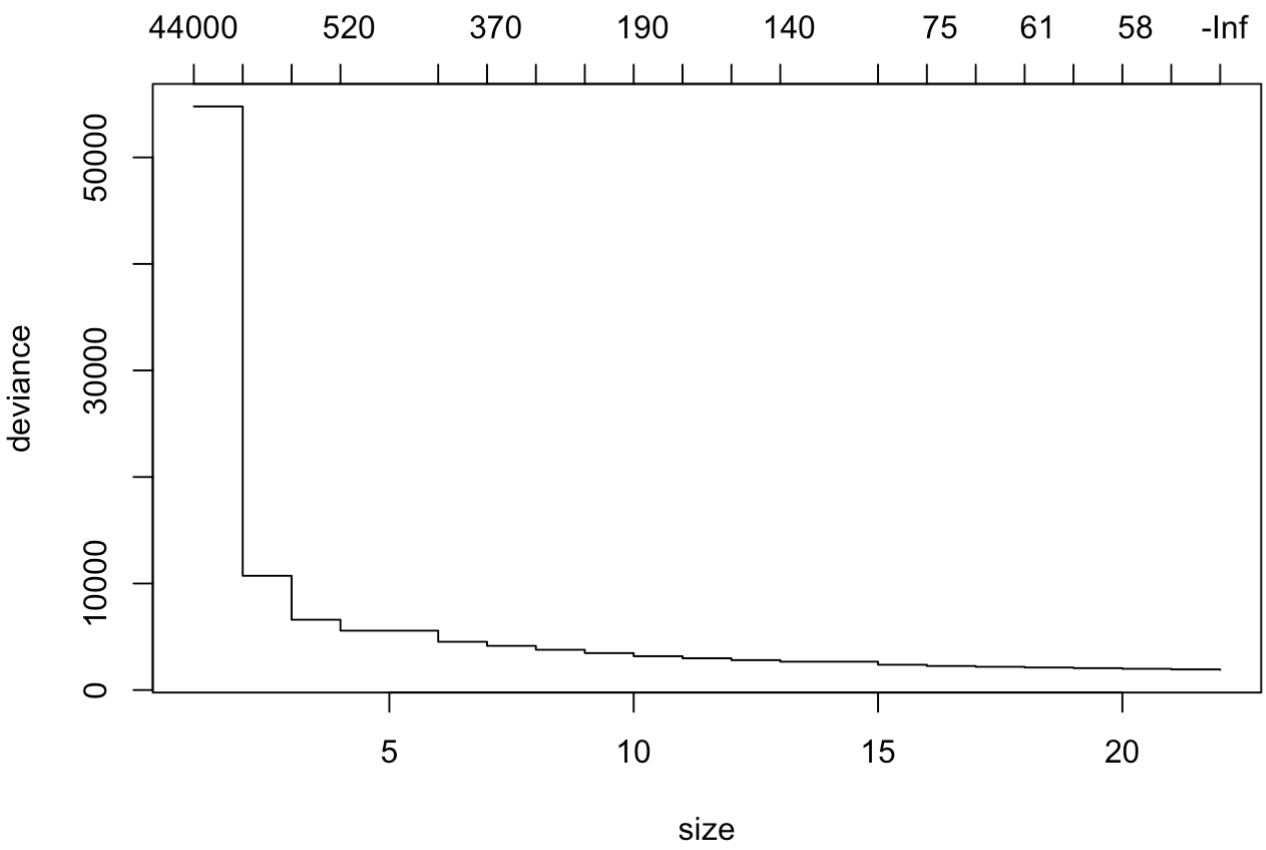
```
print(rmse(tree, test)) # RMSE on test
```

```
## [1] 2.048588
```

```
# Plotting the tree
plot(tree);text(tree, cex=0.5, digits=3)
```



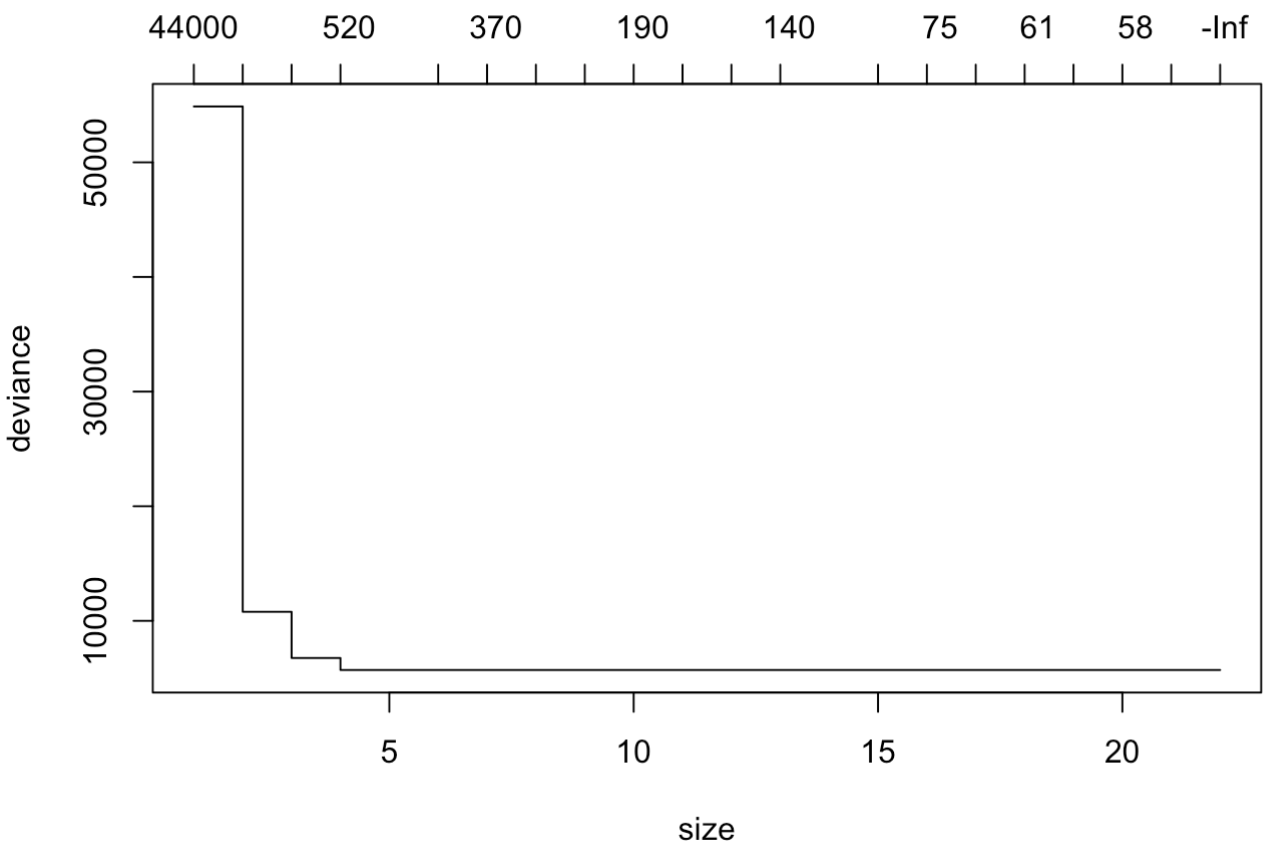

```
# prune tree
tree.pruned = prune.tree(tree)
plot(tree.pruned)
```




```
# Optimal min tree
opt.trees = which(tree.pruned$dev == min(tree.pruned$dev))
min(tree.pruned$size[opt.trees])
```

```
## [1] 22
```

```
plot(cv.tree(tree))
```



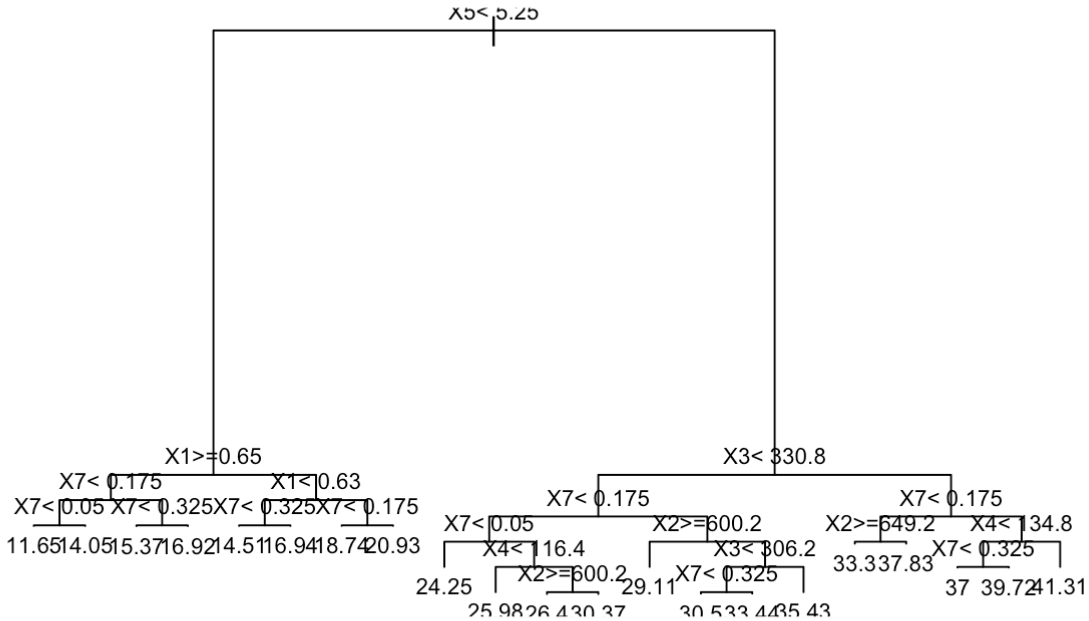

```
# Same as above, using rpart
# Model selection using different parameter value for complexity parameter.
tree1 = rpart(Y ~ X1 + X2 + X3 + X4 + X5 + as.factor(X6) + X7 + X8, data=train, cp=0.
01)
#summary(tree1)
rmse(tree1, validation)
```

```
## [1] 2.860527
```

```
tree = rpart(Y ~ X1 + X2 + X3 + X4 + X5 + as.factor(X6) + X7 + X8, data=train, cp=0.0
01)
#summary(tree)
rmse(tree, validation)
```

```
## [1] 1.723889
```

```
plot(tree);text(tree, cex=0.7)
```




```
train.model1 = m1=lm(Y ~poly(X1,4)+poly(X2,2)+poly(X3,2)+X5+X8,data=train)
summary(train.model1)
```

```
##
## Call:
## lm(formula = Y ~ poly(X1, 4) + poly(X2, 2) + poly(X3, 2) + X5 +
##     X8, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.907 -2.108  0.121  1.878  8.040
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.057e+01  5.885e+00  -1.797 0.072859 .
## poly(X1, 4)1  4.509e+03  2.532e+02  17.811  < 2e-16 ***
## poly(X1, 4)2  2.025e+03  1.191e+02  17.007  < 2e-16 ***
## poly(X1, 4)3 -4.386e+02  2.638e+01 -16.628  < 2e-16 ***
## poly(X1, 4)4  1.612e+02  1.079e+01  14.932  < 2e-16 ***
## poly(X2, 2)1  4.299e+03  2.552e+02  16.844  < 2e-16 ***
## poly(X2, 2)2 -2.664e+03  1.451e+02 -18.362  < 2e-16 ***
## poly(X3, 2)1  6.456e+01  1.924e+01   3.355 0.000844 ***
## poly(X3, 2)2 -8.073e+01  7.991e+00 -10.102  < 2e-16 ***
## X5            6.496e+00  1.116e+00   5.822 9.45e-09 ***
## X8            3.325e-01  7.421e-02   4.481 8.91e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.847 on 603 degrees of freedom
## Multiple R-squared:  0.9108, Adjusted R-squared:  0.9093
## F-statistic: 615.3 on 10 and 603 DF,  p-value: < 2.2e-16
```

```
train.model2 = m1=lm(Y ~poly(X1,4)+poly(X2,2)+poly(X3,1)+X5+X8,data=train)
summary(train.model2)
```

```
##
## Call:
## lm(formula = Y ~ poly(X1, 4) + poly(X2, 2) + poly(X3, 1) + X5 +
##     X8, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4518  -2.0816   0.1373   1.7258  10.3942
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.247e+01  4.511e+00 -11.630  < 2e-16 ***
## poly(X1, 4)1  3.605e+03  2.559e+02  14.090  < 2e-16 ***
## poly(X1, 4)2  1.626e+03  1.214e+02  13.396  < 2e-16 ***
## poly(X1, 4)3 -3.188e+02  2.546e+01 -12.524  < 2e-16 ***
## poly(X1, 4)4  7.972e+01  7.753e+00  10.282  < 2e-16 ***
## poly(X2, 2)1  3.771e+03  2.699e+02  13.972  < 2e-16 ***
## poly(X2, 2)2 -2.061e+03  1.429e+02 -14.427  < 2e-16 ***
## poly(X3, 1)  -8.361e+01  1.346e+01  -6.213 9.71e-10 ***
## X5            1.445e+01  8.547e-01  16.902  < 2e-16 ***
## X8            3.395e-01  8.018e-02   4.235 2.65e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.077 on 604 degrees of freedom
## Multiple R-squared:  0.8956, Adjusted R-squared:  0.8941
## F-statistic:   576 on 9 and 604 DF,  p-value: < 2.2e-16
```

We worked from the beginning of the project closely.Made all the decisions together.So any credit or dis-credit should be shared equally.However we contributed more or less in the following areas.

Manthan Thakkar

- Weighted Regression
- Tree Regression
- Model selection
- Model Validation

Krishna Vikas

- Plots and Figures
- Multiple regression
- Parameter estimation