CKME136 Capstone Project / Fall 2017

Student: Walter Cios; Student ID 500791945

Table of Contents

[Introduction 2](#_Toc498773985)

[Research Questions 2](#_Toc498773986)

[Dataset 2](#_Toc498773987)

[Block diagram with steps analysis points 4](#_Toc498773988)

[Step 1: Data Cleaning and Preparation 6](#_Toc498773989)

[Loading the data, review structure, and missing data 6](#_Toc498773990)

[Change column names to real variable names 7](#_Toc498773991)

[Converting categorical variables to factors 8](#_Toc498773992)

[Review summary for "data" 9](#_Toc498773993)

[Counting the unique values per variable 9](#_Toc498773994)

[Step 2: Exploratory Data Analysis 10](#_Toc498773995)

[Checking for correlated variables using paired scatterplots for "data\_numeric" 10](#_Toc498773996)

[Outlier detection 12](#_Toc498773997)

[Creating a Correlation Matrix Heatmap 15](#_Toc498773998)

[Investigating the Categorical Variables 16](#_Toc498773999)

[Step 3 Define Model Evaluation Procedure 18](#_Toc498774000)

[Methodology 18](#_Toc498774001)

[Configuration 20](#_Toc498774002)

[Step 4 Models 20](#_Toc498774003)

[Step 4.3 Random Forest 20](#_Toc498774004)

[Random Forest for Regression 20](#_Toc498774005)

[Training with cross-validation 21](#_Toc498774006)

[Step 4.2 CART 24](#_Toc498774007)

[Step 4.1 Linear Model 26](#_Toc498774008)

[Steps 5 and 6 - Model Selection and Evaluation for the Linear Model 27](#_Toc498774009)

[Training the model using backward elimination 28](#_Toc498774010)

[Step 4.4 Gradient Boosted Trees 34](#_Toc498774011)

[Step 6: Model Evaluation / Model Comparison 35](#_Toc498774012)

[A more in-depth look 35](#_Toc498774013)

[Results 43](#_Toc498774014)

[Model Comparison 43](#_Toc498774015)

[The best performing model 44](#_Toc498774016)

[Step 7: Answering the Research Questions 44](#_Toc498774017)

[Conclusions 49](#_Toc498774018)

# Introduction

Building enclosures have become more energy efficient as energy codes have become more stringent in recent years, highlighting the importance of improvements such as better windows, orientation, or insulation. The heating and cooling load calculation is the first step of the HVAC design procedure. Heating and cooling load are used to right-size the HVAC equipment of a building.

The scope of this project is to build a statistical model to predict the cooling load and to select the best performing model that provides the highest accuracy. The quantitative research will also investigate which variables are influencing the most the cooling efficiency and will explore the trade-offs between accuracy and interpretability of the model. Last, the energy efficiency impact when changing the building parameters(the window area or house orientation for example) will also be investigated as part of this study.

This model and analysis will help HVAC equipment selection and prevent purchasing of oversized equipment.

Although the heating load is also present in the dataset, for the purpose of this project,time constrains, and similarities of calculation, was decided to investigate the cooling only and disregard the heating.

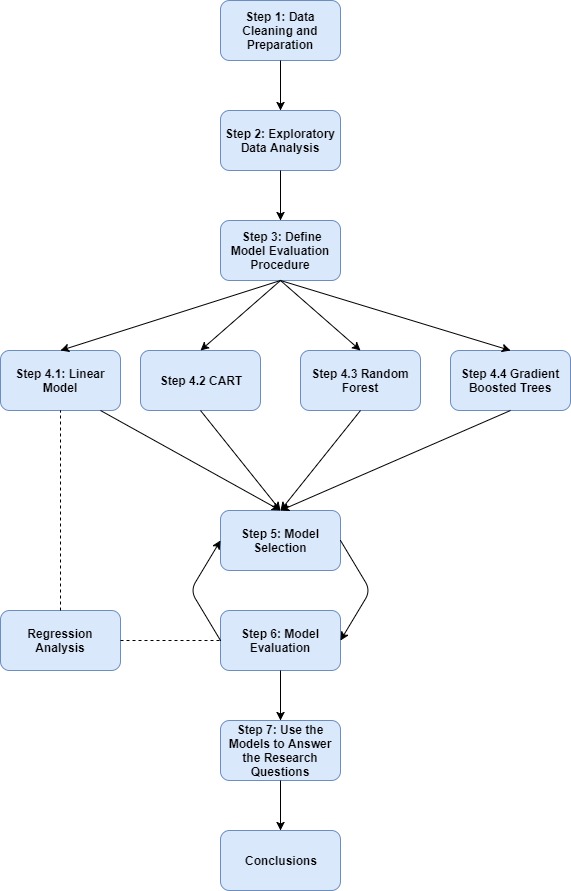
# Research Questions

* Can the cooling load be predicted accurately with a statistical model?
* Which variable are influencing the most the cooling efficiency?
* Explore trade-offs between accuracy and interpretability of the model.
* Investigate energy impact when changing some building parameters

# Dataset

The dataset analyzed is a public dataset, created in 2011 by Angeliki Xifara & Athanasios Tsanas, two UK professors. The data is a simulation of 768 residential seven person buildings, assumed to be in Athens, Greece and has eight input variables (surface area, wall area, roof area, orientation, glazing area.) and two output variables (heating and cooling load). This dataset was part of a study published in 2012. For additional details please see "Literature Review". Going forward, this study will be refered as the "original study" when comparing with the current analysis results.

# Block diagram with steps analysis points



Block Diagram

This block diagram includes the following steps that will be followed during the analysis:

* Step 1: Data Cleaning and Preparation
* Step 2: Exploratory Data Analysis
* Step 3: Define Model Evaluation Procedure
* Step 4: Models Step 4.1: Linear Model (Regression Analysis) Step 4.2: CART Step 4.3:Random Forest Step 4.4: Gradient Boosted Trees
* Step 5: Model Selection
* Step 6: Model Evaluation
* Step 7: Use the Model to Answer the Research Questions
* Step 8: Conclusions

# Step 1: Data Cleaning and Preparation

* Load package readxl to read the raw data from Excel;
* Load package ggplot2 for graphs;
* Load package dplyr for applying transformations on the dataframes;
* Load package GGally to be used for plotting especially with function ggpairs
* Load package gridExtra to be usedfor side-by-side plots of predictions vs.observed
* Load package reshape2 to be used for transformation between long and wide formats
* Load package rpart to be used for partitioning and regression trees
* Load package rpart.plot to be used to plot rpart trees
* Load package randomForestSRC that is the newer Random Forest package used for regression and classification
* Load package caret to be used for evaluation metrics

library(readxl)  
library(ggplot2)  
library(dplyr)  
library(GGally)  
library(gridExtra)  
library(reshape2)  
library(rpart)  
library(rpart.plot)  
library(randomForestSRC)  
library(caret)

### Loading the data, review structure, and missing data

After examining the original Excel file with the dataset, it can be noticed that only the first sheet has data and only this sheet will be read from the file.The other two empty sheets (with greek names) will be ignored.

raw\_data <- read\_excel("./Data/ENB2012\_data.xlsx", sheet = 1)

The structure of "raw\_data" is reviewed to make sure that all 768 observations are loaded.

str(raw\_data)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 768 obs. of 10 variables:  
## $ X1: num 0.98 0.98 0.98 0.98 0.9 0.9 0.9 0.9 0.86 0.86 ...  
## $ X2: num 514 514 514 514 564 ...  
## $ X3: num 294 294 294 294 318 ...  
## $ X4: num 110 110 110 110 122 ...  
## $ X5: num 7 7 7 7 7 7 7 7 7 7 ...  
## $ X6: num 2 3 4 5 2 3 4 5 2 3 ...  
## $ X7: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ X8: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Y1: num 15.6 15.6 15.6 15.6 20.8 ...  
## $ Y2: num 21.3 21.3 21.3 21.3 28.3 ...

Because readxl loaded all variables as numeric there cannot be blank spaces.

Review for Missing Data: the check for NA's is being done for the whole dataset

sum(sapply(raw\_data, is.na))

## [1] 0

and also as an alternative,an NA check per columns is being reviewed. As expected there was found none NA.

sapply(raw\_data, function(x) { sum(is.na(x)) })

## X1 X2 X3 X4 X5 X6 X7 X8 Y1 Y2   
## 0 0 0 0 0 0 0 0 0 0

### Change column names to real variable names

The column names have been changed to real variable names. The structure of "data" is being reviewed and the "raw\_data" will be kept in the workspace in case it's needed later in the analysis

data <- raw\_data  
names(data) <- c("Relative Compactness", "Surface Area", "Wall Area", "Roof Area", "Overall Height",  
 "Orientation", "Glazing Area", "Glazing Area Distribution", "Heating Load", "Cooling Load")  
  
str(data)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 768 obs. of 10 variables:  
## $ Relative Compactness : num 0.98 0.98 0.98 0.98 0.9 0.9 0.9 0.9 0.86 0.86 ...  
## $ Surface Area : num 514 514 514 514 564 ...  
## $ Wall Area : num 294 294 294 294 318 ...  
## $ Roof Area : num 110 110 110 110 122 ...  
## $ Overall Height : num 7 7 7 7 7 7 7 7 7 7 ...  
## $ Orientation : num 2 3 4 5 2 3 4 5 2 3 ...  
## $ Glazing Area : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Glazing Area Distribution: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Heating Load : num 15.6 15.6 15.6 15.6 20.8 ...  
## $ Cooling Load : num 21.3 21.3 21.3 21.3 28.3 ...

Exclude from analysis 48 tuples where "Glazing Area Distribution" = "0"

unique(data$`Glazing Area Distribution`)

## [1] 0 1 2 3 4 5

The original dataset has initially for "Glazing Area Distribution" six different values: "0" = No Glazing Area (no windows); 1 = Uniform (25 % windows on each side), 2 = North, 3 = East, 4 = South, 5 = West

Values like "0" (No Glazing Area) might never apply to residential buildings and could hinder in answering the research questions by influencing the model fit, since there is not possibility that a house has no windows. This is why it was decided to remove those "0" tuples from the Glazing Area Distribution.

By removing the 48 tuples with "0" Glazing Area Distribution, also all the "0" in "Glazing Area are being removed the same time.

data <- filter(data, `Glazing Area Distribution` != 0)  
unique(data$`Glazing Area Distribution`)

## [1] 1 2 3 4 5

Now after removal, the Glazing Area Distribution has only 5 values and "0" has been removed.

### Converting categorical variables to factors

As per the dataset description, it was established that there are two categorical variables:"Orientation" and "Glazing Area Distribution". The variable "Orientation" has only four values, the cardinal directions: N,W,S,E. The variable "Glazing Area Distribution" has all the above four and in addition "1" for "Uniform".

Using the dataset dictionary, the integers from the raw data have been mapped to categorical values.

"Relevel" was used in order to bring "Uniform" (that has value "1") to the first position

toCategorical <- Vectorize(function(column) {  
 switch(as.character(column),  
 "1" = "Uniform",  
 "2" = "North",  
 "3" = "East",  
 "4" = "South",  
 "5" = "West",  
 NA)  
})  
  
data$`Glazing Area Distribution` <- as.factor(toCategorical(data$`Glazing Area Distribution`))  
data$Orientation <- as.factor(toCategorical(data$Orientation))  
  
data$`Glazing Area Distribution` <- relevel(data$`Glazing Area Distribution`, "Uniform")

### Review summary for "data"

summary(data)

## Relative Compactness Surface Area Wall Area Roof Area   
## Min. :0.6200 Min. :514.5 Min. :245.0 Min. :110.2   
## 1st Qu.:0.6825 1st Qu.:606.4 1st Qu.:294.0 1st Qu.:140.9   
## Median :0.7500 Median :673.8 Median :318.5 Median :183.8   
## Mean :0.7642 Mean :671.7 Mean :318.5 Mean :176.6   
## 3rd Qu.:0.8300 3rd Qu.:741.1 3rd Qu.:343.0 3rd Qu.:220.5   
## Max. :0.9800 Max. :808.5 Max. :416.5 Max. :220.5   
## Overall Height Orientation Glazing Area Glazing Area Distribution  
## Min. :3.50 East :180 Min. :0.10 Uniform:144   
## 1st Qu.:3.50 North:180 1st Qu.:0.10 East :144   
## Median :5.25 South:180 Median :0.25 North :144   
## Mean :5.25 West :180 Mean :0.25 South :144   
## 3rd Qu.:7.00 3rd Qu.:0.40 West :144   
## Max. :7.00 Max. :0.40   
## Heating Load Cooling Load   
## Min. :10.07 Min. :13.20   
## 1st Qu.:13.17 1st Qu.:15.78   
## Median :21.05 Median :23.67   
## Mean :22.84 Mean :24.91   
## 3rd Qu.:32.13 3rd Qu.:33.48   
## Max. :43.10 Max. :48.03

From the summary, can be concluded that Orientation and Glazing Area Distribution, are qualitative-categorical variables, the other variables are quantitative-continuous. Those conclusions and the Min and Max will be used in an Excel table that explains the dataset in more detail.

### Counting the unique values per variable

sapply(data, function(col) length(unique(col)))

## Relative Compactness Surface Area   
## 12 12   
## Wall Area Roof Area   
## 7 4   
## Overall Height Orientation   
## 2 4   
## Glazing Area Glazing Area Distribution   
## 3 5   
## Heating Load Cooling Load   
## 548 596

Review str after the change to display the 720 remaining values

str(data)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 720 obs. of 10 variables:  
## $ Relative Compactness : num 0.98 0.98 0.98 0.98 0.9 0.9 0.9 0.9 0.86 0.86 ...  
## $ Surface Area : num 514 514 514 514 564 ...  
## $ Wall Area : num 294 294 294 294 318 ...  
## $ Roof Area : num 110 110 110 110 122 ...  
## $ Overall Height : num 7 7 7 7 7 7 7 7 7 7 ...  
## $ Orientation : Factor w/ 4 levels "East","North",..: 2 1 3 4 2 1 3 4 2 1 ...  
## $ Glazing Area : num 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 ...  
## $ Glazing Area Distribution: Factor w/ 5 levels "Uniform","East",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ Heating Load : num 24.6 24.6 24.6 24.6 29 ...  
## $ Cooling Load : num 26.5 26.4 26.4 26.3 32.9 ...

# Step 2: Exploratory Data Analysis

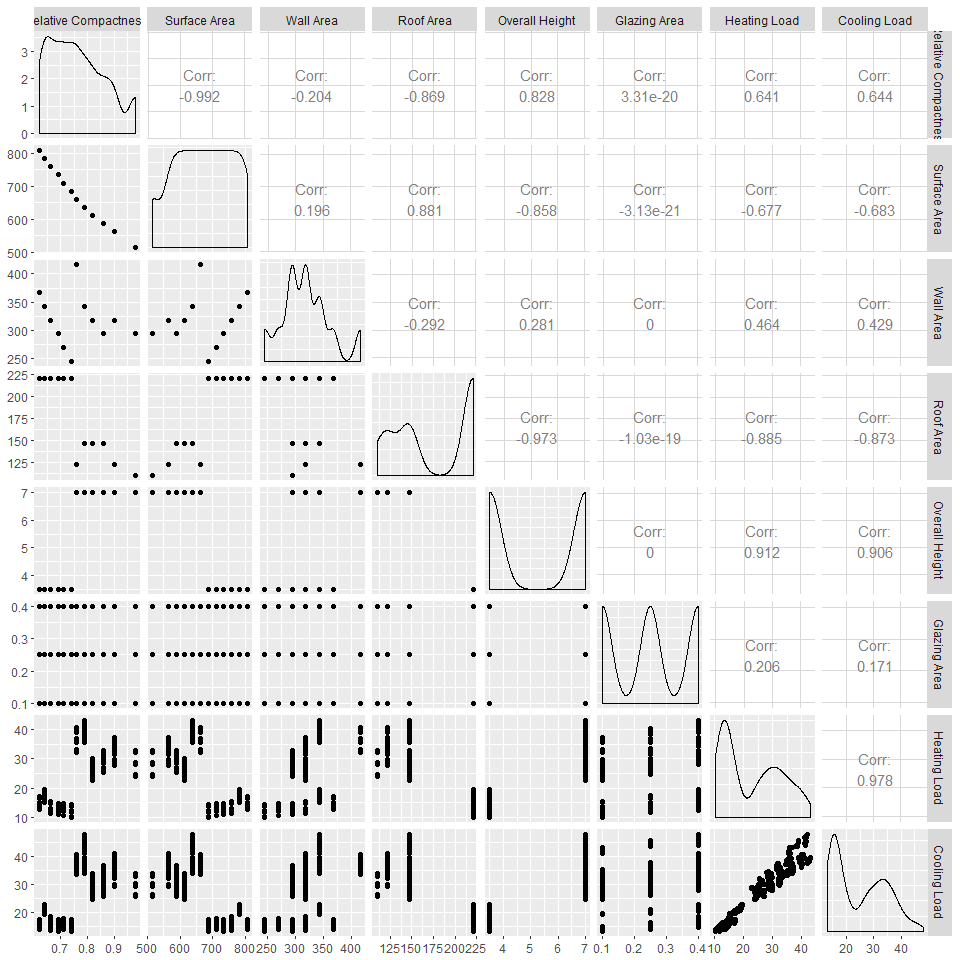
The categorical variables are split from the continuous variables, into a new dataframe called "data\_categorical" and we keep the continuous variables in "data\_numeric". This is done to make it clear which methods apply to the continuous variables(correlations) and which apply to the categorical variables(counting with the table function).

data\_numeric <- select(data, -`Orientation`, -`Glazing Area Distribution`)  
data\_categorical <- select(data, `Orientation`, `Glazing Area Distribution`)

### Checking for correlated variables using paired scatterplots for "data\_numeric"

Next the analysis visually checked for correlated variables in the numerical dataset using an optimal display of plot and selected the package "GGally" where the function ggpairs is being used for creating scatterplots between all pairs of continuous variables from "data\_numeric".

ggpairs(data\_numeric)



The following are the interpretations of the major correlations:

Relative Compacteness & Surface Area / strong almost perfect negative correlation at -0.992

Relative Compacteness & Wall Area / low negative correlation

Relative Compacteness & Roof Area / strong negative correlation

Surface Area & Overall Height / strong negative correlation

Surface Area & Roof Area / strong positive correlation

Relative Compactness & Glazing Area / "0" correlation

Relative Compactness & Heating Load /medium strong positive correlation

Wall Area & Heating Load / low positive correlation

Relative Compacteness & Cooling Load / poitive high correlation

Surface Area & Cooling Load / negative high correlation

Wall Area & Cooling Load / low positive correlation

Roof Area & Cooling Load / strong negative correlation

Overall Height & Cooling Load / strong positive correlation

Glazing Area & Cooling Load / low positive correlation

Heating Load & Cooling Load / strong almost perfect positive correlation

### Outlier detection

Review if there are outliers by using boxplots. Going forward this analysis will try to use only ggplot for all graphs to create a consistent view.

Looking at the Excel table "Variable Review Summary" that has the Min and Max, it was decided to split the boxplots of data-numeric into three categories by cluster of data: small (values 0-1); medium (values >1-50); large (>50)

small <- select(data\_numeric,`Relative Compactness`,`Glazing Area`)  
  
medium <- select(data\_numeric,`Overall Height`,`Heating Load`,`Cooling Load`)  
  
large <- select(data\_numeric,`Surface Area`,`Roof Area`,`Wall Area`)

#### Creating boxplots for the grouped variables into small, medium and large

Boxplot are created for the variables that are grouped by value (see Excel Table Fig 2)

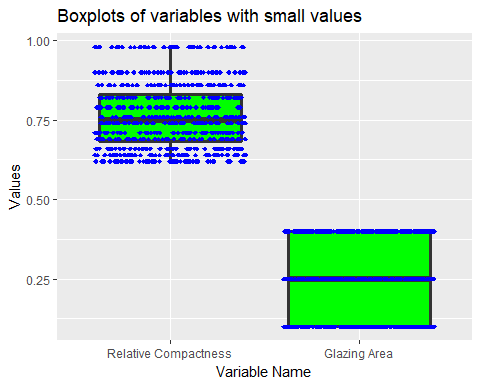
A couple of features have been used to make the boxplots better visible on paper:

* The boxplots were plotted using ggplot2;
* In the geom\_boxplot, the inside was filled "green";
* For the median, it was used "lwd" = 1.2, slightly more than normal, ("lwd" stands for line width) to display a thicker line and this way to be better visible;
* Used geom\_jitter to add a small amount of variation and make the dots more visible by changing: the shape to be "diamond" = shape # 18, colour "blue" to show better on green, and increase slightly the width of the blue diamond dots

The numeric data will be split in three data frames because the variables have different scales.

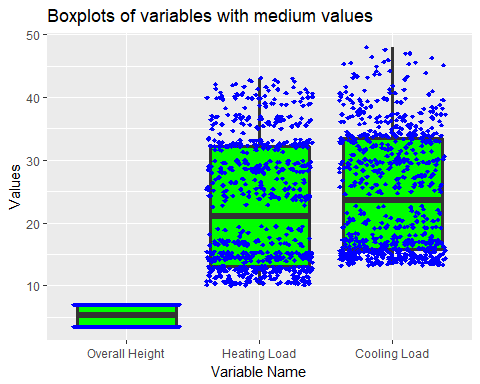
"Melt"" from the reshape2 package was used because the boxplots have the variable name on the x-axis and the long data format (one column with the variable name and another one with the value). "Melt" can be easily used by ggplot2 to create the side-by-side boxplots.

small\_long <- melt(small)  
ggplot(data=small\_long, aes(x = variable, y = value)) +  
 geom\_boxplot(fill = "green", lwd = 1.2) +  
 geom\_jitter(height = 0, width = 0.4, shape = 18, color = "blue") +  
 ggtitle("Boxplots of variables with small values") +  
 xlab("Variable Name") +  
 ylab("Values")



The boxplots of variables with small values (Relative Compactness, Glazing Area) are showing that we have no outliers and that there are several data points that have the same values what is the result of the simulated values in the dataset. "Relative Compacteness" takes 12 values only and "Glazing Area" only four values.

medium\_long <- melt(medium)  
  
ggplot(data=medium\_long, aes(x = variable, y = value)) +  
 geom\_boxplot(fill = "green",lwd = 1.2) +  
 geom\_jitter(height = 0, width = 0.4, shape = 18, color = "blue") +  
 ggtitle("Boxplots of variables with medium values") +  
 xlab("Variable Name") +  
 ylab("Values")



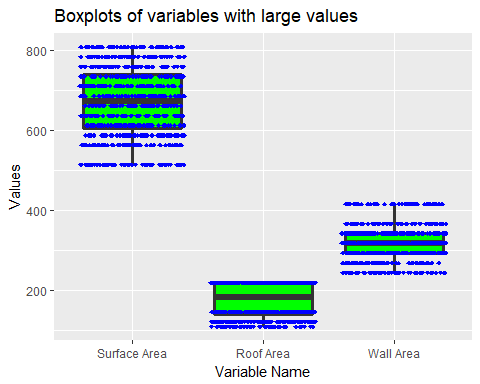
The boxplots of variables with medium values (Overall Height, Heating Load, and Cooling Load) are showing that we have no outliers and that there are several data points clustered in the bottom part of the boxplot of the Heating Load and Cooling Load (between 10 and 15).

For the Overall Height, this has only two values 3,5 and 7.

For Heating Load we see that the data is mostly clustered in the first quartile of the boxplot.

For Cooling Load we see more variability in the data but most data is grouped together in the first quartile.

large\_long <- melt(large)  
  
ggplot(data=large\_long, aes(x = variable, y = value)) +  
 geom\_boxplot(fill = "green", lwd = 1.2) +  
 geom\_jitter(height = 0, width = 0.4, shape = 18, color = "blue") +  
 ggtitle("Boxplots of variables with large values") +  
 xlab("Variable Name") +  
 ylab("Values")



The boxplots of variables with high values (Surface Area, Roof Area, and Wall Area) are showing that there are no outliers and that Surface Area has only 12 possible values, Wall Area has only 7 values, and Roof Area has only 4 values.

### Creating a Correlation Matrix Heatmap

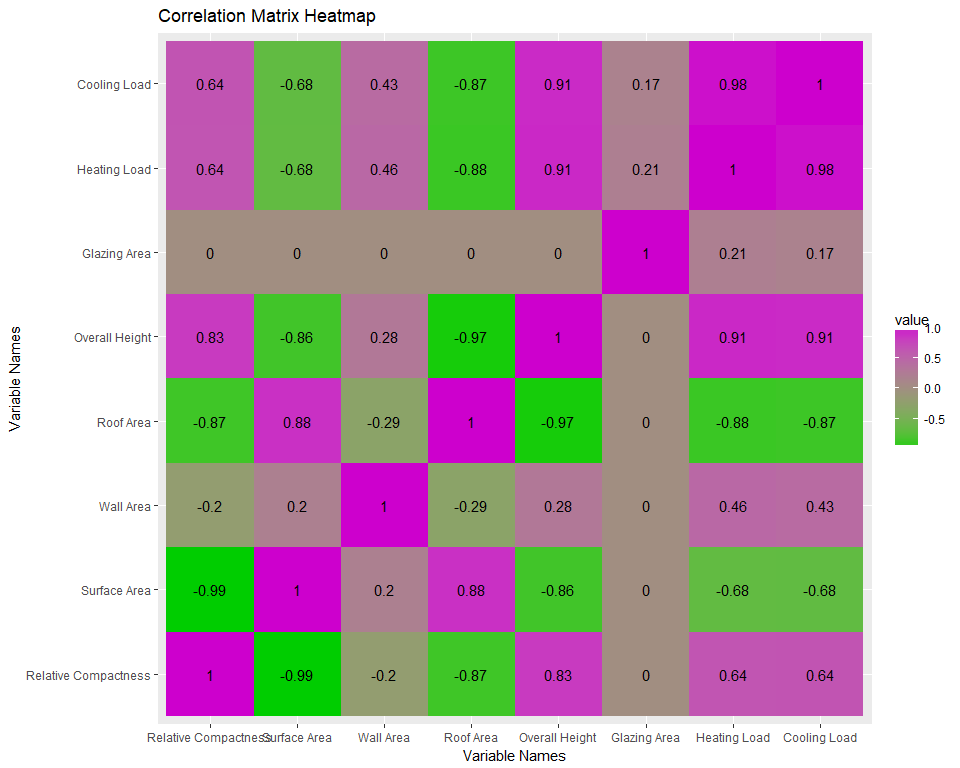
A Correlation Matrix Heatmap was created (using "ggplot") because colours are better way to quickly visualize any large or small correlations.

A couple of features have been used to make the Correlation Matrix Heatmap a better visualization:

* The cor method "Pearson"
* geom\_text to write the cor values as text in the square
* Selected 2 colours "green3" and "magenta 3" From: <http://sape.inf.usi.ch/quick-reference/ggplot2/colour>
* A scale gradient of colours with values between -1 and 1
* The green colour is showing the negative correlation and the magenta colour the positive correlation
* The geom\_tile is displaying the rectangles
* Set the figure height at 8 and width at 10 to better show up when printed.

Having the Correlation Matrix Heatmap it is easy to visualize the lighter and darker colours that stand for low and high correlation.

correlationMatrix <- cor(data\_numeric, method = "pearson")  
correlationsLong <- melt(correlationMatrix)  
  
ggplot(data = correlationsLong, aes(x = Var1, y = Var2)) +  
 geom\_tile(aes(fill = value)) +  
 geom\_text(aes(label = round(value, 2))) +  
 scale\_fill\_gradient(low = "green3", high = "magenta3") +  
 ggtitle("Correlation Matrix Heatmap") +  
 xlab("Variable Names") +  
 ylab("Variable Names")



### Investigating the Categorical Variables

The two categorical variables are: "Orientation" and "Glazing Area Distribution". "Orientation"" has only the four values of the cardinal directions: East, North, South, and West

table(data\_categorical$Orientation)

##   
## East North South West   
## 180 180 180 180

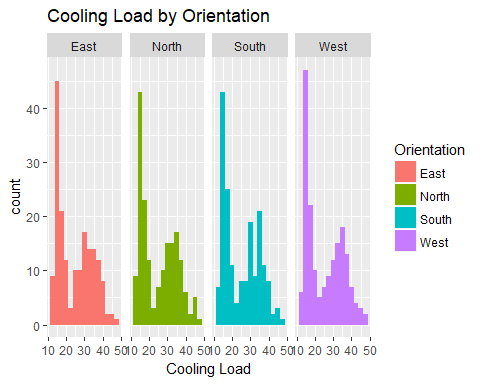
The tables are displayed for both categorical variable "Orientation" and "Glazing Are Distribution" to show that the "Glazing Area Distribution" has the four values of the cardinal directions and includes also "Uniform".

table(data\_categorical$`Glazing Area Distribution`)

##   
## Uniform East North South West   
## 144 144 144 144 144

Next, facet\_grid will be used to plot the histograms of the Cooling Load for each value of the categorical variables.

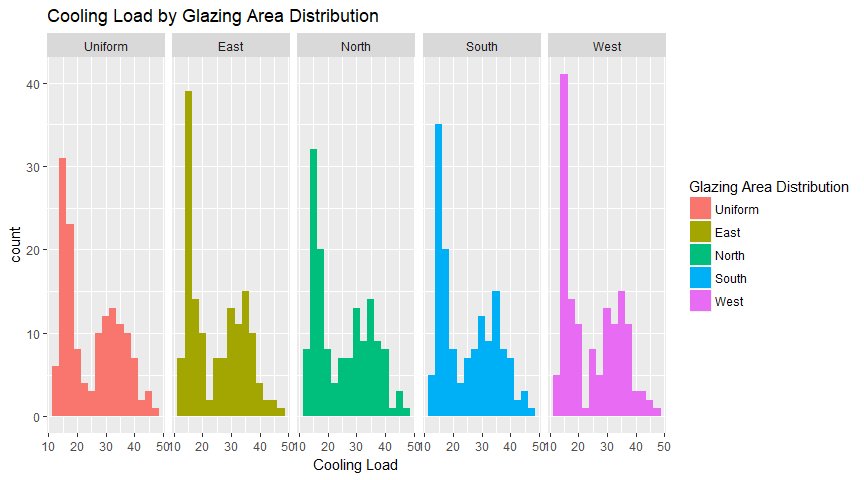
plotData <- select(data, `Cooling Load`, `Orientation`)  
  
ggplot(plotData, aes(x = `Cooling Load`, fill = `Orientation`)) +  
 geom\_histogram(binwidth = 2.5) +  
 facet\_grid(~ Orientation) +  
 ggtitle("Cooling Load by Orientation")



Plot of the categorical variable "Orientation" against the predicted variable "Cooling Load"

For all orientations the Cooling Load distribution is roughly bimodal, with one peak around 15 and the other around 35.

plotData <- select(data, `Cooling Load`, `Glazing Area Distribution`)  
  
ggplot(plotData, aes(x = `Cooling Load`, fill = `Glazing Area Distribution`)) +  
 geom\_histogram(binwidth = 2.5) +  
 facet\_grid(~ `Glazing Area Distribution`) +  
 ggtitle("Cooling Load by Glazing Area Distribution")



Similar results will be obtained for Glazing Area Distribution

The initial intent was to use One-way ANOVA to check if these categorical variables would have an impact on the predicted variable, but because the distributions within the groups are clearly not normal, this can't be used.

# Step 3 Define Model Evaluation Procedure

### Methodology

Regression will be the main focus of this study because the dataset contains one numeric variable (Cooling Load) that needs to be predicted, and a set of independent variables which can be used to make that prediction.

In general, the objectives of regressions are:

* To review if there is a relationship between two variables and especially if there is a statistically significant relationship between those two
* To predict new observations.

The objectives of this study are to determine which predictors influence the response variable and the nature of the relationships between the predictors and the response

The models that will be used are:

* Random forest with all the variables because it is a very powerful model capable of capturing non-linear relationships between the predictors and the response variable and this was the best performing model in the original study
* CART because it is an interpretable model which could help in answering some of the research questions and because the scatterplots in the previous step had some groups which could be handled by the splits
* Multiple linear regression because it can represent very simple linear relationships between the predictors and the response variable and if it performs well, it could answer more questions than the other two models

The models will use the following variables:

* random forest: all variables
* CART: all variables
* Linear model: multiple models created with backward elimination, starting from the model with all variables and eliminating the variables one by one

For all the models, prediction and residual analyses will be performed, but this is especially important in the case of the linear model, which makes the assumption that the errors are normally distributed.

To avoid overfitting, all the measures will be computed on different datasets than the ones that were used to train the models. After the best performing models are selected, they will be trained on the complete dataset.

The most important criterion for choosing the best model will be R2. This study will not use ANOVA to compare the stepwise models, instead it will use information criteria (AIC and/or BIC) at the end of the procedure to select the model with the fewest variables that still has a good R2. ANOVA will not be used to stop eliminating variables, the backward elimination procedure will be stopped when R2 becomes too small. For non-linear models like CART and Random Forest, k-fold cross-validation (k = 10) will be used to pick the best performing model according to R2.

Going forward MSE and MAE will be used as evaluation measures.

The interpretability of the models and how can they help us to answer the original research questions, will also be reviewed.

From the two output variables the prediction of the "Cooling Load" only will be analyzed (that was marked in the original study with "y2"), and this is why the "Heating Load" will be excluded.

The set.seed will be used to ensure the reproducibility of the results. The data is shuffled because it is generated and the order in which it was generated would create biased folds in cross-validation.

The analysis will start with random forest and 10-fold cross-validation. In other models, a classical train-test split(70-30) will be used.

### Configuration

The following code block contains the configuration of the model evaluation procedures, to facilitate an easy change of k or the training proportion if needed.

modelData <- select(data, -`Heating Load`)  
set.seed(100)  
shuffledModelData <- modelData[sample(1:nrow(modelData), nrow(modelData), replace = F),]  
k <- 10  
trainProportion <- 0.7

# Step 4 Models

The regression variant of Random Forest and CART models will be used to predict the cooling load because this is a numeric output variable.

The analysis will start with Random Forest with 10-fold cross-validation.

Step 5 "Model Selection" and step 6 "Model Evaluation" will be done individually for each model.

# Step 4.3 Random Forest

Random Forest is defined as per Wikipedia as "a way of averaging multiple decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally boost the performance of the final model".

### Random Forest for Regression

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set with responses bagging repeatedly (B times) selects a "random sample with replacement" of the training set and fits trees to these samples:

For b = 1,...,B there are samples with replacements taken and "n" training examples from X, Y, and call these , .

In addition to bagging, to further decorrelate the trees there is an additional technique used by random forest, specifically only using a limited subset of variables when trying to make a split. Usually these subsets are of a fixed size and that size is chosen before training and used as a hyperparameter called "mtry".

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x'.

Here "X" = the entire dataset of numerical and categorical variables.

X = Relative Compactness, Surface Area, Wall Area, Roof Area, Overall Height, Orientation, Glazing Area, Glazing Area Distribution

Y = Cooling Load

Prediction formulas on an unseen data point x':

It can be clearly seen from the formula that the prediction is a simple average on the predictions from all the trees.

### Training with cross-validation

The cross-validation results will be used to pick the best performing model. A simple grid search will be used for the hyperparameter selection. In the grid values have been selected such that the best performing model is not at the extremes. The intervals of the hyperparameters (mtry, nodesize, ntree, and nodedepth) were picked after several trial runs.

The following will explain the grid search method. The optimization of the model hyperparmeters can be done using either a "randomized search" or a "grid search". In this study the method of full grid search over all parameters was selected in order to improve the quality of the results. Other factors that could have been investigated for improvements of results could have been: better feature engineering, using different regularization methods for regression or different regression methods.

Grid search is done by picking a set of values for each hyperparameter, generating all possible combinations, then training and evaluating all the models and picking the one which scored best according to the evaluation criteria.

#### Defining the evaluation function and the hyperparameter grid

For random forest the hyperparameter grid will be generated based on the following values:

* mtry(number of possible choices at each split): 2, 3, 4
* nodesize(average number of data points in a leaf): 5, 10, 15
* ntree(number of trees): 50, 100, 150
* nodedepth(maximum depth): 2, 3, 4, 5, 6, 7

validateRF <- function(model, data) {  
 observed <- data$`Cooling Load`  
 predicted <- predict(model, newdata = data)$predicted  
 list(  
 R2 = R2(predicted, observed),  
 MSE = RMSE(predicted, observed)^2,  
 MAE = MAE(predicted, observed))  
}  
  
RFparams <- expand.grid(  
 mtry = 2:4,  
 nodesize = seq(5, 15, 5),  
 ntree = seq(50, 150, 50),  
 nodedepth = 2:7)

For evaluation, several measures will be computed (R2, MSE, MAE) and extract the mean over all folds. A total of 162 different random forest models will be fit, but only display the best 10 and the worst 5, sorted by R2 in descending order(after training). There are 162 models because it is the cardinality of the cartesian product between all the hyperparameters(3 from mtry x 3 from nodesize x 3 from ntree x 6 from nodedepth = 162 models).

str(RFparams)

## 'data.frame': 162 obs. of 4 variables:  
## $ mtry : int 2 3 4 2 3 4 2 3 4 2 ...  
## $ nodesize : num 5 5 5 10 10 10 15 15 15 5 ...  
## $ ntree : num 50 50 50 50 50 50 50 50 50 100 ...  
## $ nodedepth: int 2 2 2 2 2 2 2 2 2 2 ...  
## - attr(\*, "out.attrs")=List of 2  
## ..$ dim : Named int 3 3 3 6  
## .. ..- attr(\*, "names")= chr "mtry" "nodesize" "ntree" "nodedepth"  
## ..$ dimnames:List of 4  
## .. ..$ mtry : chr "mtry=2" "mtry=3" "mtry=4"  
## .. ..$ nodesize : chr "nodesize= 5" "nodesize=10" "nodesize=15"  
## .. ..$ ntree : chr "ntree= 50" "ntree=100" "ntree=150"  
## .. ..$ nodedepth: chr "nodedepth=2" "nodedepth=3" "nodedepth=4" "nodedepth=5" ...

#### Model Training

For each model, the cross-validation results are computed as the mean over all folds, and then arranged in descending order by R2.

RFevalCV <- data.frame()  
for (paramIndex in 1:nrow(RFparams)) {  
 params <- RFparams[paramIndex,]  
   
 folds <- cut(1:nrow(shuffledModelData), breaks=k, label=F)  
   
 cvR2 <- c()  
 cvMSE <- c()  
 cvMAE <- c()  
 for (foldIndex in 1:k) {  
 testData <- modelData[folds == foldIndex,]  
 trainData <- modelData[folds != foldIndex,]  
   
 model <- rfsrc(`Cooling Load` ~ ., data = as.data.frame(trainData),  
 mtry = params$mtry,  
 nodesize = params$nodesize,  
 ntree = params$ntree,  
 nodedepth = params$nodedepth)  
 modelEval <- validateRF(model, as.data.frame(testData))  
 cvR2 <- c(cvR2, modelEval$R2)  
 cvMSE <- c(cvMSE, modelEval$MSE)  
 cvMAE <- c(cvMAE, modelEval$MAE)  
 }  
   
 evalRow <- mutate(params, R2 = mean(cvR2), MSE = mean(cvMSE), MAE = mean(cvMAE))  
 RFevalCV <- rbind(RFevalCV, evalRow)  
}  
RFevalCV <- arrange(RFevalCV, desc(R2))  
rbind(head(RFevalCV, 10), tail(RFevalCV, 5))

## mtry nodesize ntree nodedepth R2 MSE MAE  
## 1 4 5 150 6 0.9689602 2.791938 1.161716  
## 2 4 5 100 7 0.9689443 2.715298 1.134985  
## 3 4 5 100 6 0.9686782 2.799561 1.162938  
## 4 4 5 150 7 0.9685585 2.728734 1.128049  
## 5 4 10 50 6 0.9685152 2.854634 1.187069  
## 6 3 5 100 7 0.9684285 3.011059 1.245999  
## 7 4 5 50 6 0.9682746 2.805293 1.174041  
## 8 4 5 50 7 0.9682439 2.775195 1.146437  
## 9 4 10 50 7 0.9681948 2.818231 1.187488  
## 10 4 10 150 6 0.9681410 2.915003 1.202348  
## 158 2 15 100 2 0.9081906 10.559527 2.509921  
## 159 2 5 100 2 0.9080179 10.396570 2.490647  
## 160 2 10 150 2 0.9067724 10.643412 2.491688  
## 161 2 5 50 2 0.9062137 10.608288 2.509234  
## 162 2 15 50 2 0.9060386 10.416410 2.458308

Next, will fit the model that performed best in cross-validation on the whole dataset because the models were trained on the cross-validation folds.Next, the model will also be displayed.

RFmodel <- rfsrc(`Cooling Load` ~ ., data = as.data.frame(modelData), importance = T,  
 mtry = RFevalCV$mtry[1],  
 nodesize = RFevalCV$nodesize[1],  
 ntree = RFevalCV$ntree[1],  
 nodedepth = RFevalCV$nodedepth[1])  
  
RFmodel

## Sample size: 720  
## Number of trees: 150  
## Forest terminal node size: 5  
## Average no. of terminal nodes: 56.32667  
## No. of variables tried at each split: 4  
## Total no. of variables: 8  
## Analysis: RF-R  
## Family: regr  
## Splitting rule: mse  
## % variance explained: 96.7  
## Error rate: 2.99

The model comparison will be done after all the models are trained.

# Step 4.2 CART

CART (that stands for Classification and Regression Trees) is a recursive partitioning method to build regression trees to predict the value of a continuous variable from one or more predictor variables that can be continuous and/or categorical.

The CART model will be used since it is a more interpretable model compared to random forest.

When creating the CART process of computing the regression trees, these four steps should be followed:

* Specify the criteria for predictive accuracy
* Select the splits
* Determining when to stop splitting
* Select the right-size tree

For CART a train-test split will be used, but would still use the grid search method for hyperparameter selection. In the grid values have been picked such that the best performing model is not at the extremes. The intervals were picked after several trial runs.

One of the reasons why the CART model was used here, originated in the fact that it was also used in the following study included in the "Literature Review" section: "Advancing monthly streamflow prediction accuracy of CART models using ensemble learning paradigms" - by Halil Ibrahim Erdal, Onur Karakurt.

#### Defining the evaluation function and the hyperparameter grid for CART

The hyperparameter grid will be generated based on the following values:

* minsplit(minimum number of points in node before split): 10, 20, 30, 40
* cp(cost complexity factor): 0.00001, 0.0005, 0.0001, 0.005, 0.001
* maxdepth(maximum tree depth): 4, 5, 6, 7, 8

validateCART <- function(model, data) {  
 observed <- data$`Cooling Load`  
 predicted <- predict(model, newdata = data)  
 list(  
 R2 = R2(predicted, observed),  
 MSE = RMSE(predicted, observed)^2,  
 MAE = MAE(predicted, observed))  
}  
  
CARTparams <- expand.grid(  
 minsplit = seq(10, 40, 10),  
 cp = c(1e-5, 0.0005, 0.0001, 0.005, 0.001),  
 maxdepth = 4:8)

100 CART models will be evaluated on the training set:

str(CARTparams)

## 'data.frame': 100 obs. of 3 variables:  
## $ minsplit: num 10 20 30 40 10 20 30 40 10 20 ...  
## $ cp : num 1e-05 1e-05 1e-05 1e-05 5e-04 5e-04 5e-04 5e-04 1e-04 1e-04 ...  
## $ maxdepth: int 4 4 4 4 4 4 4 4 4 4 ...  
## - attr(\*, "out.attrs")=List of 2  
## ..$ dim : Named int 4 5 5  
## .. ..- attr(\*, "names")= chr "minsplit" "cp" "maxdepth"  
## ..$ dimnames:List of 3  
## .. ..$ minsplit: chr "minsplit=10" "minsplit=20" "minsplit=30" "minsplit=40"  
## .. ..$ cp : chr "cp=1e-05" "cp=5e-04" "cp=1e-04" "cp=5e-03" ...  
## .. ..$ maxdepth: chr "maxdepth=4" "maxdepth=5" "maxdepth=6" "maxdepth=7" ...

The CART model performance will be displayed similar to how it was presented for Random Forest.

#### CART Model training

CARTeval <- data.frame()  
for (paramIndex in 1:nrow(CARTparams)) {  
 params <- CARTparams[paramIndex,]  
   
 inTrain <- 1:nrow(modelData) < trainProportion \* nrow(modelData)  
   
 testData <- modelData[!inTrain,]  
 trainData <- modelData[inTrain,]  
   
 model <- rpart(`Cooling Load` ~ ., data = trainData, method = "anova",  
 control = rpart.control(  
 minsplit = params$minsplit,  
 maxdepth = params$maxdepth,  
 cp = params$cp))  
 modelEval <- validateCART(model, testData)  
   
 evalRow <- mutate(params, R2 = modelEval$R2, MSE = modelEval$MSE, MAE = modelEval$MAE)  
 CARTeval <- rbind(CARTeval, evalRow)  
}  
CARTeval <- arrange(CARTeval, desc(R2))  
rbind(head(CARTeval, 10), tail(CARTeval, 5))

## minsplit cp maxdepth R2 MSE MAE  
## 1 30 1e-05 5 0.9722589 5.598062 2.025062  
## 2 40 1e-05 5 0.9722589 5.598062 2.025062  
## 3 30 1e-04 5 0.9722589 5.598062 2.025062  
## 4 40 1e-04 5 0.9722589 5.598062 2.025062  
## 5 10 1e-03 6 0.9718888 4.921170 1.893572  
## 6 10 1e-03 7 0.9718888 4.921170 1.893572  
## 7 10 1e-03 8 0.9718888 4.921170 1.893572  
## 8 40 1e-05 6 0.9713548 5.681350 2.062381  
## 9 30 1e-04 6 0.9713548 5.681350 2.062381  
## 10 40 1e-04 6 0.9713548 5.681350 2.062381  
## 96 40 5e-03 4 0.9656338 8.453481 2.494280  
## 97 20 5e-04 6 0.9654545 5.795976 2.025057  
## 98 20 5e-04 7 0.9654545 5.795976 2.025057  
## 99 20 5e-04 8 0.9654545 5.795976 2.025057  
## 100 10 5e-04 5 0.9647697 6.135821 2.073247

Now the CART model will be fit on the whole dataset because the previous models were fit on the training set.

CARTmodel <- rpart(`Cooling Load` ~ ., data = modelData, method = "anova",  
 control = rpart.control(  
 minsplit = CARTeval$minsplit[1],  
 maxdepth = CARTeval$maxdepth[1],  
 cp = CARTeval$cp[1]))

The prediction analysis and model comparison will be done after training the linear model.

# Step 4.1 Linear Model

The linear model assumes that the dependent variable can be accurately predicted using a linear combination of the dependent variables by using the coefficients vector . is the error term, which represents what cannot be learned by the model.

To make a prediction for an instance of the variable Y, if we have the outcomes for the variables and we know the coefficients , we only need to compute the linear combination:

The R's lm function will be used to estimate the coefficient vector .

### Steps 5 and 6 - Model Selection and Evaluation for the Linear Model

The most important measure used to evaluate the model is the R2, but this study will also use Akaike Information Criterion (AIC), a measure that penalizes the objective function for linear regression by the number of variables included in the model. Penalizing complex models is used to avoid overfitting.

AIC is an estimator of the relative quality of a statistical model for a given data set. When given several models, AIC estimates the quality of each model, relative to each of the other models. AIC is used to make "apples to apples" comparison of different models over the same dataset.

In AIC, the model with the lowest AIC score will have the smallest divergence and will be the better, preferred model.

First the evaluation method will be defined, which includes the model AIC, along with the same train-test (70/30) split that was used to evaluate the CART model.

**Explaining the usage of backward elimination**

The model selection methods which will be used is backward elimination, starting from the complete model with all variables and eliminating variables one by one. Some variables might have multiple coefficients in the model and they will be removed together.

"Backward" is one four most used variable selection methods: forward, backwards, stepwise, and all subsets. Selecting the "backward" method in this was a result of a study analyzed in the "Literature Review" section, called "Benchmarking the energy efficiency of commercial buildings" by William Chung, Y.V. Hui, and Y. Miu Lam and where the "backward" elimination method is referred to as: "applied to select the regression models where insignificant explanatory variables are eliminated.

**Defining the evaluation function and the training set**

validateLM <- function(model, data) {  
 observed <- data$`Cooling Load`  
 predicted <- predict(model, newdata = data)  
 list(  
 R2 = R2(predicted, observed),  
 MSE = RMSE(predicted, observed)^2,  
 MAE = MAE(predicted, observed),  
 AIC = extractAIC(model)[2])  
}  
  
inTrain <- 1:nrow(modelData) < trainProportion \* nrow(modelData)  
   
testData <- modelData[!inTrain,]  
trainData <- modelData[inTrain,]

The extractAIC function returns a vector with 2 values, the first value from the result is the degrees of freedom, which will be ignored for this evaluation. Only the second value will be used, which is the AIC value.

### Training the model using backward elimination

**Model "completeLinearModel" with all variables**

Next a linear model will be trained on all the variables in the train dataset.

The results of the evaluation will be loaded into a new blank LMeval dataframe where all the models will be analysed

LMeval <- data.frame()  
completeLinearModel <- lm(`Cooling Load` ~ ., data = trainData)  
summary(completeLinearModel)

##   
## Call:  
## lm(formula = `Cooling Load` ~ ., data = trainData)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.5368 -1.5902 -0.2495 1.3641 10.9095   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 97.031052 26.053983 3.724 0.000219  
## `Relative Compactness` -71.457264 14.061224 -5.082 5.33e-07  
## `Surface Area` -0.085277 0.023325 -3.656 0.000284  
## `Wall Area` 0.041841 0.008903 4.700 3.39e-06  
## `Roof Area` NA NA NA NA  
## `Overall Height` 4.386974 0.455804 9.625 < 2e-16  
## OrientationNorth 0.336587 0.407302 0.826 0.408989  
## OrientationSouth 0.103095 0.407302 0.253 0.800284  
## OrientationWest 0.638658 0.408143 1.565 0.118278  
## `Glazing Area` 14.259574 1.702997 8.373 5.93e-16  
## `Glazing Area Distribution`East -0.495815 0.450078 -1.102 0.271168  
## `Glazing Area Distribution`North -0.347273 0.450078 -0.772 0.440733  
## `Glazing Area Distribution`South -0.313002 0.450078 -0.695 0.487109  
## `Glazing Area Distribution`West -0.459356 0.450078 -1.021 0.307941  
##   
## (Intercept) \*\*\*  
## `Relative Compactness` \*\*\*  
## `Surface Area` \*\*\*  
## `Wall Area` \*\*\*  
## `Roof Area`   
## `Overall Height` \*\*\*  
## OrientationNorth   
## OrientationSouth   
## OrientationWest   
## `Glazing Area` \*\*\*  
## `Glazing Area Distribution`East   
## `Glazing Area Distribution`North   
## `Glazing Area Distribution`South   
## `Glazing Area Distribution`West   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.233 on 490 degrees of freedom  
## Multiple R-squared: 0.8849, Adjusted R-squared: 0.8821   
## F-statistic: 313.9 on 12 and 490 DF, p-value: < 2.2e-16

The Roof Area variable will be removed because its coefficient is NA, which means it is linearly dependent to the other variables in the dataset.

**Model "model7Var" with all variables except Roof Area**

model7Var <- lm(`Cooling Load` ~ .,  
 data = select(trainData, -`Roof Area`))  
   
modelEval <- validateLM(model7Var, testData)  
   
evalRow <- data.frame(model = "model7Var", R2 = modelEval$R2,  
 MSE = modelEval$MSE, MAE = modelEval$MAE,  
 AIC = modelEval$AIC)  
LMeval <- rbind(LMeval, evalRow)  
  
summary(model7Var)

##   
## Call:  
## lm(formula = `Cooling Load` ~ ., data = select(trainData, -`Roof Area`))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.5368 -1.5902 -0.2495 1.3641 10.9095   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 97.031052 26.053983 3.724 0.000219  
## `Relative Compactness` -71.457264 14.061224 -5.082 5.33e-07  
## `Surface Area` -0.085277 0.023325 -3.656 0.000284  
## `Wall Area` 0.041841 0.008903 4.700 3.39e-06  
## `Overall Height` 4.386974 0.455804 9.625 < 2e-16  
## OrientationNorth 0.336587 0.407302 0.826 0.408989  
## OrientationSouth 0.103095 0.407302 0.253 0.800284  
## OrientationWest 0.638658 0.408143 1.565 0.118278  
## `Glazing Area` 14.259574 1.702997 8.373 5.93e-16  
## `Glazing Area Distribution`East -0.495815 0.450078 -1.102 0.271168  
## `Glazing Area Distribution`North -0.347273 0.450078 -0.772 0.440733  
## `Glazing Area Distribution`South -0.313002 0.450078 -0.695 0.487109  
## `Glazing Area Distribution`West -0.459356 0.450078 -1.021 0.307941  
##   
## (Intercept) \*\*\*  
## `Relative Compactness` \*\*\*  
## `Surface Area` \*\*\*  
## `Wall Area` \*\*\*  
## `Overall Height` \*\*\*  
## OrientationNorth   
## OrientationSouth   
## OrientationWest   
## `Glazing Area` \*\*\*  
## `Glazing Area Distribution`East   
## `Glazing Area Distribution`North   
## `Glazing Area Distribution`South   
## `Glazing Area Distribution`West   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.233 on 490 degrees of freedom  
## Multiple R-squared: 0.8849, Adjusted R-squared: 0.8821   
## F-statistic: 313.9 on 12 and 490 DF, p-value: < 2.2e-16

Based on the p-values and for the Glazing Area Distribution no coefficients seem relevant, so we will be excluded the first because the p-values for all the coefficients are large.

**Model "model6Var" with all variables except: Roof Area, Glazing Area Distribution**

model6Var <- lm(`Cooling Load` ~ .,  
 data = select(trainData,   
 -`Roof Area`, -`Glazing Area Distribution`))  
  
modelEval <- validateLM(model6Var, testData)  
  
evalRow <- data.frame(model = "model6Var", R2 = modelEval$R2,  
 MSE = modelEval$MSE, MAE = modelEval$MAE,  
 AIC = modelEval$AIC)  
LMeval <- rbind(LMeval, evalRow)  
  
summary(model6Var)

##   
## Call:  
## lm(formula = `Cooling Load` ~ ., data = select(trainData, -`Roof Area`,   
## -`Glazing Area Distribution`))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.7595 -1.6169 -0.2743 1.3334 10.8881   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 96.474022 25.985157 3.713 0.000228 \*\*\*  
## `Relative Compactness` -71.380947 14.026303 -5.089 5.12e-07 \*\*\*  
## `Surface Area` -0.085141 0.023267 -3.659 0.000280 \*\*\*  
## `Wall Area` 0.041778 0.008881 4.704 3.31e-06 \*\*\*  
## `Overall Height` 4.395287 0.454621 9.668 < 2e-16 \*\*\*  
## OrientationNorth 0.336587 0.406295 0.828 0.407827   
## OrientationSouth 0.103095 0.406295 0.254 0.799798   
## OrientationWest 0.636940 0.407131 1.564 0.118350   
## `Glazing Area` 14.664827 1.662746 8.820 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.225 on 494 degrees of freedom  
## Multiple R-squared: 0.8845, Adjusted R-squared: 0.8826   
## F-statistic: 472.9 on 8 and 494 DF, p-value: < 2.2e-16

Now the coefficient p-values are analyzed for Orientation.

Even though OrientationWest seems a little relevant, it will be excluded from the next model and the evaluation table will be used at the end to see if it was the right choice.

**Model "model5Var" with all variables except: Roof Area, Glazing Area Distribution,and Orientation**

model5Var <- lm(`Cooling Load` ~ .,  
 data = select(trainData,   
 -`Roof Area`,  
 -`Glazing Area Distribution`, -`Orientation`))  
  
modelEval <- validateLM(model5Var, testData)  
  
evalRow <- data.frame(model = "model5Var", R2 = modelEval$R2,  
 MSE = modelEval$MSE, MAE = modelEval$MAE,  
 AIC = modelEval$AIC)  
LMeval <- rbind(LMeval, evalRow)  
  
summary(model5Var)

##   
## Call:  
## lm(formula = `Cooling Load` ~ ., data = select(trainData, -`Roof Area`,   
## -`Glazing Area Distribution`, -Orientation))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.7202 -1.6148 -0.2947 1.2610 11.0916   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 96.49668 25.98115 3.714 0.000227 \*\*\*  
## `Relative Compactness` -71.24712 14.02449 -5.080 5.34e-07 \*\*\*  
## `Surface Area` -0.08490 0.02326 -3.649 0.000291 \*\*\*  
## `Wall Area` 0.04167 0.00888 4.692 3.50e-06 \*\*\*  
## `Overall Height` 4.39942 0.45456 9.678 < 2e-16 \*\*\*  
## `Glazing Area` 14.64442 1.66251 8.809 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.225 on 497 degrees of freedom  
## Multiple R-squared: 0.8838, Adjusted R-squared: 0.8827   
## F-statistic: 756.3 on 5 and 497 DF, p-value: < 2.2e-16

Next, Surface Area will be removed from the model, because its coefficient has the largest p-value.

**Model "model4Var" with all variables except: Roof Area, Glazing Area Distribution, Orientation, and Surface Area**

model4Var <- lm(`Cooling Load` ~ .,  
 data = select(trainData,   
 -`Roof Area`, -`Surface Area`,  
 -`Glazing Area Distribution`, -`Orientation`))  
  
modelEval <- validateLM(model4Var, testData)  
  
evalRow <- data.frame(model = "model4Var", R2 = modelEval$R2,  
 MSE = modelEval$MSE, MAE = modelEval$MAE,  
 AIC = modelEval$AIC)  
LMeval <- rbind(LMeval, evalRow)  
  
summary(model4Var)

##   
## Call:  
## lm(formula = `Cooling Load` ~ ., data = select(trainData, -`Roof Area`,   
## -`Surface Area`, -`Glazing Area Distribution`, -Orientation))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.848 -1.567 -0.052 1.426 10.954   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.556987 3.568924 0.716 0.47404   
## `Relative Compactness` -22.237735 4.092065 -5.434 8.62e-08 \*\*\*  
## `Wall Area` 0.017166 0.005883 2.918 0.00368 \*\*   
## `Overall Height` 5.784536 0.253227 22.843 < 2e-16 \*\*\*  
## `Glazing Area` 14.587307 1.682869 8.668 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.264 on 498 degrees of freedom  
## Multiple R-squared: 0.8807, Adjusted R-squared: 0.8798   
## F-statistic: 919.3 on 4 and 498 DF, p-value: < 2.2e-16

The next variable to be removed is Wall Area, because its coefficient estimate has the highest p-value.

**Model "model3Var" with only: Wall Area, Overall Height, and Glazing Area (here Relative Compactness was removed)**

model3Var <- lm(`Cooling Load` ~ .,  
 data = select(trainData,   
 -`Roof Area`, -`Surface Area`, -`Relative Compactness`,  
 -`Glazing Area Distribution`, -`Orientation`))  
  
modelEval <- validateLM(model3Var, testData)  
  
evalRow <- data.frame(model = "model3Var", R2 = modelEval$R2,  
 MSE = modelEval$MSE, MAE = modelEval$MAE,  
 AIC = modelEval$AIC)  
LMeval <- rbind(LMeval, evalRow)  
  
summary(model3Var)

##   
## Call:  
## lm(formula = `Cooling Load` ~ ., data = select(trainData, -`Roof Area`,   
## -`Surface Area`, -`Relative Compactness`, -`Glazing Area Distribution`,   
## -Orientation))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -6.8296 -2.0993 -0.2444 1.6624 12.0092   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -15.853239 1.154319 -13.734 < 2e-16 \*\*\*  
## `Wall Area` 0.042906 0.003588 11.958 < 2e-16 \*\*\*  
## `Overall Height` 4.492601 0.089671 50.101 < 2e-16 \*\*\*  
## `Glazing Area` 14.596516 1.730312 8.436 3.57e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.356 on 499 degrees of freedom  
## Multiple R-squared: 0.8736, Adjusted R-squared: 0.8729   
## F-statistic: 1150 on 3 and 499 DF, p-value: < 2.2e-16

This will be the last linear model included in the evaluation because the R2 is already becoming significantly worse compared to the other models. These are the complete results of the linear model:

LMeval

## model R2 MSE MAE AIC  
## 1 model7Var 0.9061119 9.315848 2.261936 1193.237  
## 2 model6Var 0.9056072 9.411815 2.282419 1186.836  
## 3 model5Var 0.9058620 9.391960 2.275222 1183.772  
## 4 model4Var 0.9013414 9.782951 2.254171 1195.074  
## 5 model3Var 0.9025706 9.708549 2.188891 1222.052

To summarize, the variables included in each linear model are:

* model7Var:Glazing Area, Glazing Area Distribution, Orientation, Overall Height, Relative Compactness, Surface Area, Wall Area
* model6Var:Glazing Area, Orientation, Overall Height, Relative Compactness, Surface Area, Wall Area
* model5Var:Glazing Area, Overall Height, Relative Compactness, Surface Area, Wall Area
* model4Var:Glazing Area, Overall Height, Relative Compactness, Wall Area
* model3Var:Glazing Area, Overall Height, Wall Area

The names of the models are based on how many variables they use.

Based on the above results the best selected model is "model5Var" (without Roof Area, Orientation and Glazing Area Distribution) because it has the second best R2 and MSE, but a lower number of variables. This is also indicated by AIC, which has the lowest value for this model.

Now model5Var will be fit on the whole dataset.

finalLM <- lm(`Cooling Load` ~ .,  
 data = select(modelData, -`Roof Area`, -`Glazing Area Distribution`, -`Orientation`))

### Step 4.4 Gradient Boosted Trees

In general, boosting is used for weak learners and our decision tree was already a very good fit on the data with a R2 of over 0.95 in Random Forest and CART and this is why this model of gradient boosted trees will not be pursued this further.

# Step 6: Model Evaluation / Model Comparison

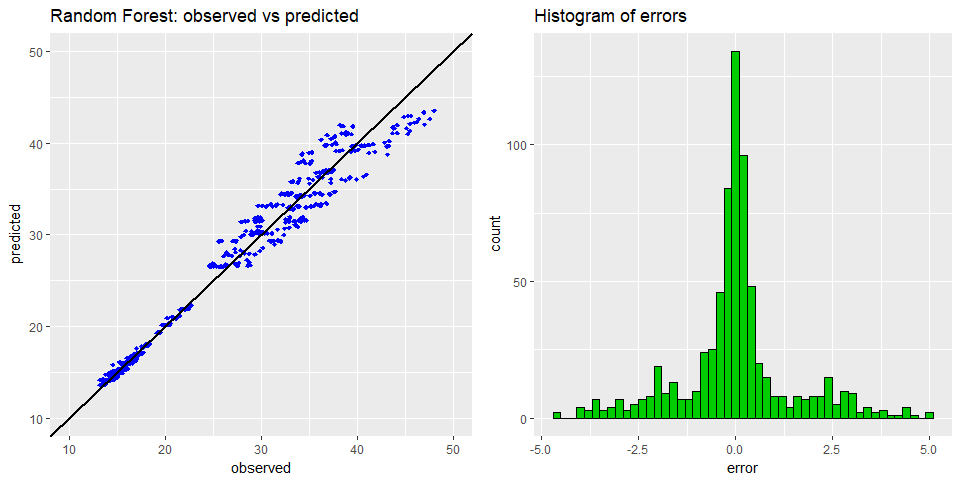
### A more in-depth look

Although the CART MSE is almost twice the random forest MSE, the CART R2 is surprisingly larger, so this is why a more in-depth look at the predictions is needed.

Futher the errors will be ranked and the next step will be to determine the conditions that lead to large errors in the predictions.

#### Random Forest Prediction Analysis

predicted <- predict(RFmodel)$predicted  
observed <- modelData$`Cooling Load`  
  
predictionData <- data.frame(predicted, observed) %>%  
 mutate(error = observed - predicted, abserror = abs(error))  
  
plot1 <- ggplot(predictionData, aes(x = observed, y = predicted)) +  
 geom\_point(shape = 18, color = "blue") +  
 geom\_abline(lwd = 1) +  
 xlim(10, 50) +  
 ylim(10, 50) +  
 ggtitle("Random Forest: observed vs predicted")  
  
plot2 <- ggplot(predictionData, aes(x = error)) +  
 geom\_histogram(binwidth = 0.2, fill = "green3", color = "black") +  
 ggtitle("Histogram of errors")  
  
grid.arrange(plot1, plot2, ncol = 2)



The error distribution is centered at 0 and it seems to be unimodal.

Now the worst predictions will be examined in order to determine if there is a pattern in the data points that the model gets wrong. Only the top 25 errors will be analyzed.

worstPredictions <- modelData %>%  
 bind\_cols(predictionData) %>%  
 arrange(desc(abserror)) %>%  
 head(25)  
  
worstPredictions

## # A tibble: 25 x 13  
## `Relative Compactness` `Surface Area` `Wall Area` `Roof Area`  
## <dbl> <dbl> <dbl> <dbl>  
## 1 0.79 637.0 343.0 147.0  
## 2 0.79 637.0 343.0 147.0  
## 3 0.79 637.0 343.0 147.0  
## 4 0.79 637.0 343.0 147.0  
## 5 0.79 637.0 343.0 147.0  
## 6 0.90 563.5 318.5 122.5  
## 7 0.79 637.0 343.0 147.0  
## 8 0.90 563.5 318.5 122.5  
## 9 0.79 637.0 343.0 147.0  
## 10 0.79 637.0 343.0 147.0  
## # ... with 15 more rows, and 9 more variables: `Overall Height` <dbl>,  
## # Orientation <fctr>, `Glazing Area` <dbl>, `Glazing Area  
## # Distribution` <fctr>, `Cooling Load` <dbl>, predicted <dbl>,  
## # observed <dbl>, error <dbl>, abserror <dbl>

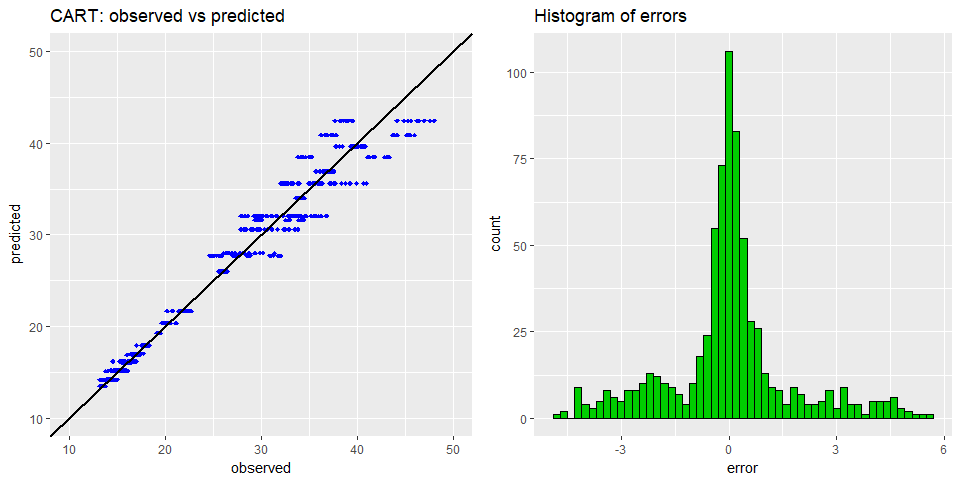
worstPredictions %>%  
 group\_by(`Surface Area`, `Wall Area`, `Roof Area`, `Overall Height`) %>%  
 summarise(count = n()) %>%  
 arrange(desc(count))

## # A tibble: 3 x 5  
## # Groups: Surface Area, Wall Area, Roof Area [3]  
## `Surface Area` `Wall Area` `Roof Area` `Overall Height` count  
## <dbl> <dbl> <dbl> <dbl> <int>  
## 1 637.0 343.0 147.0 7 20  
## 2 563.5 318.5 122.5 7 3  
## 3 588.0 294.0 147.0 7 2

It can be seen that most of the worst predictions are made for one type of building. Because there are very few data points it is difficult to generalize this result and find all the building types for which the model might not work.

#### CART Prediction Analysis

predicted <- predict(CARTmodel)  
observed <- modelData$`Cooling Load`  
  
predictionData <- data.frame(predicted, observed) %>%  
 mutate(error = observed - predicted, abserror = abs(error))  
  
plot1 <- ggplot(predictionData, aes(x = observed, y = predicted)) +  
 geom\_point(shape = 18, color = "blue") +  
 geom\_abline(lwd = 1) +  
 xlim(10, 50) +  
 ylim(10, 50) +  
 ggtitle("CART: observed vs predicted")  
  
plot2 <- ggplot(predictionData, aes(x = error)) +  
 geom\_histogram(binwidth = 0.2, fill = "green3", color = "black") +  
 ggtitle("Histogram of errors")  
  
grid.arrange(plot1, plot2, ncol = 2)



The error distribution is centered at 0 and it seems to be unimodal, although there seem to be more negative errors than positive ones.

Now the worst predictions will be examined in order to determine if there is a pattern in the data points that the model gets wrong. We will only look at the top 25 errors.

worstPredictions <- modelData %>%  
 bind\_cols(predictionData) %>%  
 arrange(desc(abserror)) %>%  
 head(25)  
  
worstPredictions

## # A tibble: 25 x 13  
## `Relative Compactness` `Surface Area` `Wall Area` `Roof Area`  
## <dbl> <dbl> <dbl> <dbl>  
## 1 0.79 637.0 343.0 147.0  
## 2 0.90 563.5 318.5 122.5  
## 3 0.79 637.0 343.0 147.0  
## 4 0.79 637.0 343.0 147.0  
## 5 0.90 563.5 318.5 122.5  
## 6 0.79 637.0 343.0 147.0  
## 7 0.79 637.0 343.0 147.0  
## 8 0.86 588.0 294.0 147.0  
## 9 0.79 637.0 343.0 147.0  
## 10 0.86 588.0 294.0 147.0  
## # ... with 15 more rows, and 9 more variables: `Overall Height` <dbl>,  
## # Orientation <fctr>, `Glazing Area` <dbl>, `Glazing Area  
## # Distribution` <fctr>, `Cooling Load` <dbl>, predicted <dbl>,  
## # observed <dbl>, error <dbl>, abserror <dbl>

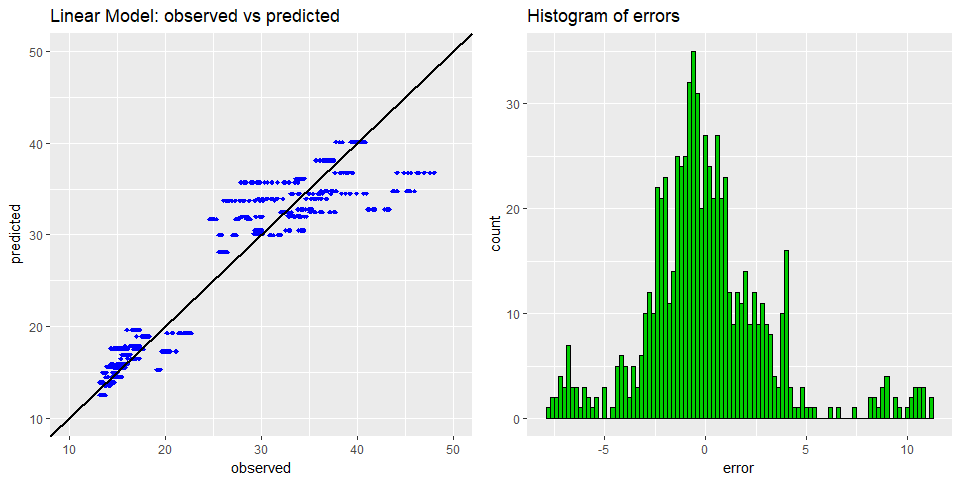
worstPredictions %>%  
 group\_by(`Surface Area`, `Wall Area`, `Roof Area`, `Overall Height`) %>%  
 summarise(count = n()) %>%  
 arrange(desc(count))

## # A tibble: 3 x 5  
## # Groups: Surface Area, Wall Area, Roof Area [3]  
## `Surface Area` `Wall Area` `Roof Area` `Overall Height` count  
## <dbl> <dbl> <dbl> <dbl> <int>  
## 1 637.0 343.0 147.0 7 19  
## 2 563.5 318.5 122.5 7 3  
## 3 588.0 294.0 147.0 7 3

We can see that most of the worst predictions are made for one type of building. Because there are very few data points it is difficult to generalize this result and find all the building types for which the model might not work.

#### Linear Model Prediction Analysis

predicted <- predict(finalLM)  
observed <- modelData$`Cooling Load`  
  
predictionData <- data.frame(predicted, observed) %>%  
 mutate(error = observed - predicted, abserror = abs(error))  
  
plot1 <- ggplot(predictionData, aes(x = observed, y = predicted)) +  
 geom\_point(shape = 18, color = "blue") +  
 geom\_abline(lwd = 1) +  
 xlim(10, 50) +  
 ylim(10, 50) +  
 ggtitle("Linear Model: observed vs predicted")  
  
plot2 <- ggplot(predictionData, aes(x = error)) +  
 geom\_histogram(binwidth = 0.2, fill = "green3", color = "black") +  
 ggtitle("Histogram of errors")  
  
grid.arrange(plot1, plot2, ncol = 2)



The assumption that the linear model makes is that the errors are independent and normally distributed. By doing a histogram of the residuals and a QQ plot the feasibility of this assumption will be investigated.

The residual distribution is not quite centered at 0, more like at -1, there are some data points with large errors that biased the whole model.

Now the worst predictions will be examined in order to determine if there is a pattern in the data points that the model gets wrong. Only the top 25 errors will be analyzed.

worstPredictions <- modelData %>%  
 bind\_cols(predictionData) %>%  
 arrange(desc(abserror)) %>%  
 head(25)  
  
worstPredictions

## # A tibble: 25 x 13  
## `Relative Compactness` `Surface Area` `Wall Area` `Roof Area`  
## <dbl> <dbl> <dbl> <dbl>  
## 1 0.79 637 343 147  
## 2 0.79 637 343 147  
## 3 0.79 637 343 147  
## 4 0.79 637 343 147  
## 5 0.79 637 343 147  
## 6 0.79 637 343 147  
## 7 0.79 637 343 147  
## 8 0.79 637 343 147  
## 9 0.79 637 343 147  
## 10 0.79 637 343 147  
## # ... with 15 more rows, and 9 more variables: `Overall Height` <dbl>,  
## # Orientation <fctr>, `Glazing Area` <dbl>, `Glazing Area  
## # Distribution` <fctr>, `Cooling Load` <dbl>, predicted <dbl>,  
## # observed <dbl>, error <dbl>, abserror <dbl>

worstPredictions %>%  
 group\_by(`Surface Area`, `Wall Area`, `Roof Area`, `Overall Height`) %>%  
 summarise(count = n()) %>%  
 arrange(desc(count))

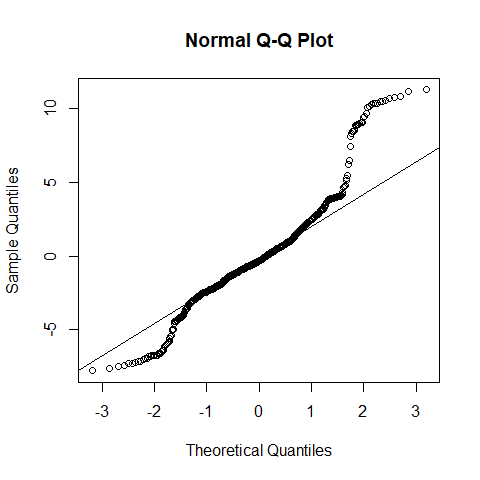
## # A tibble: 1 x 5  
## # Groups: Surface Area, Wall Area, Roof Area [1]  
## `Surface Area` `Wall Area` `Roof Area` `Overall Height` count  
## <dbl> <dbl> <dbl> <dbl> <int>  
## 1 637 343 147 7 25

It is the same building like for the other models, but the errors are even bigger and more domain knowledge is necessary to decide if removing that building from the training set makes sense.

#### Residual Analysis for the Linear Model

The residuals should form a normal distribution centered at 0, but the plot shows that this is not the case.

qqnorm(predictionData$error)  
qqline(predictionData$error)



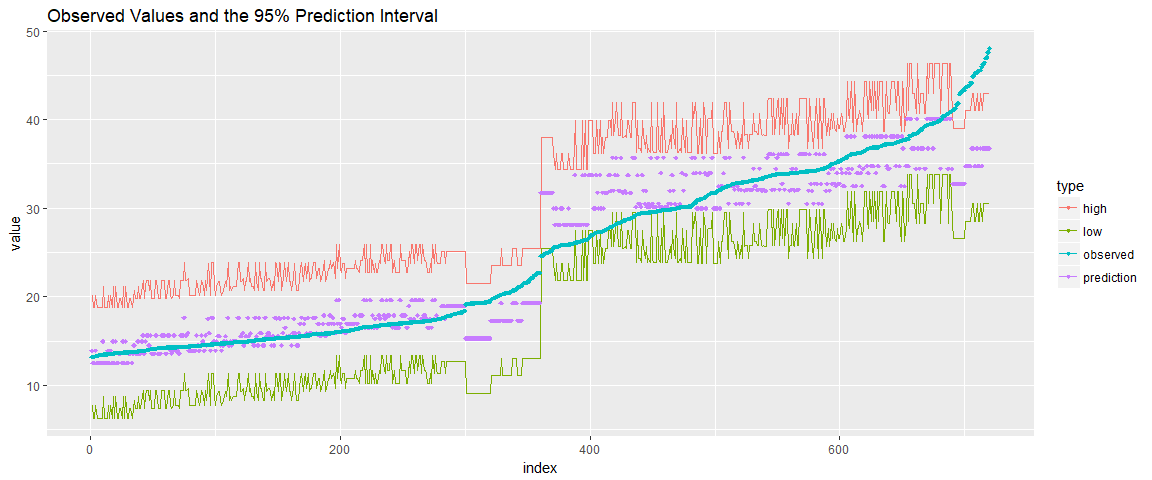
The QQ plot shows that the distribution of the residuals is quite far from normal. Theoretical Quantiles are from the Normal distribution and Sample Quantiles are from the residuals. We can see from the plot that there are more values at the extremes than in a normal distribution with the same parameters.

#### Prediction interval (95%) for the Linear Model

For both models Random Forest and CART this study selected not to investigate confidence intervals because there is no standard theoretically sound way to compute them like there is for linear models.

First the data is sorted by the observed values(Cooling Load), then the 95% prediction intervals are computed, then everything is plotted to see in how many instances the actual values(blue dots) are outside the prediction interval(upper - red, lower - green). The x-axis contains the indices, and the y-axis is used for the values, which are between 13 and 48 for Cooling Load.

set.seed(100)  
sortedData <- arrange(modelData, `Cooling Load`)  
  
predictions <- predict(finalLM, sortedData, interval = "prediction", level = 0.95)  
  
plotData <- data.frame(  
 index = c(rep(1:nrow(sortedData), 3)),  
 value = c(predictions[,1], predictions[,2], predictions[,3]),  
 type = c(rep("prediction",nrow(sortedData)) , rep("low", nrow(sortedData)), rep("high", nrow(sortedData))))  
  
observed <- data.frame(  
 index = 1:nrow(sortedData),  
 value = sortedData$`Cooling Load`,  
 type = "observed")  
  
ggplot(plotData, aes(x = index, y = value, colour = type)) +  
 geom\_line(data = filter(plotData, type != "prediction")) +  
 geom\_point(data = filter(plotData, type == "prediction"), shape = 18) +  
 geom\_point(data = observed, shape = 18) +  
 ggtitle("Observed Values and the 95% Prediction Interval")



The observed values are outside the interval for buildings with high cooling load values, which means that those types of buildings don't fit the pattern learned by the model. A solution would be to gather more data about buildings with high cooling load.

tail(sortedData, 10)

## # A tibble: 10 x 9  
## `Relative Compactness` `Surface Area` `Wall Area` `Roof Area`  
## <dbl> <dbl> <dbl> <dbl>  
## 1 0.79 637 343 147  
## 2 0.79 637 343 147  
## 3 0.79 637 343 147  
## 4 0.79 637 343 147  
## 5 0.79 637 343 147  
## 6 0.79 637 343 147  
## 7 0.79 637 343 147  
## 8 0.79 637 343 147  
## 9 0.79 637 343 147  
## 10 0.79 637 343 147  
## # ... with 5 more variables: `Overall Height` <dbl>, Orientation <fctr>,  
## # `Glazing Area` <dbl>, `Glazing Area Distribution` <fctr>, `Cooling  
## # Load` <dbl>

After a more in detail look, it can be observed that all the observations with a high cooling load belong to that building which all the models had trouble predicting, so this might be a data quality issue. Either that building is an outlier, or more data is needed for buildings with high cooling loads.

# Results

### Model Comparison

At this point we will compare the results of all the models that we have been trained until now.

evaluationResults <- data.frame(  
 model = c("original study", "Random Forest", "CART", "Linear Model"),  
 R2 = c(NA, RFevalCV$R2[1], CARTeval$R2[1], LMeval$R2[1]),  
 MSE = c(6.59, RFevalCV$MSE[1], CARTeval$MSE[1], LMeval$MSE[1]),  
 MAE = c(1.42, RFevalCV$MAE[1], CARTeval$MAE[1], LMeval$MAE[1]))  
  
evaluationResults

## model R2 MSE MAE  
## 1 original study NA 6.590000 1.420000  
## 2 Random Forest 0.9689602 2.791938 1.161716  
## 3 CART 0.9722589 5.598062 2.025062  
## 4 Linear Model 0.9061119 9.315848 2.261936

As it can be seen from the table, the random forest with our chosen parameters seems to outperform the model from the original study, however there are some other facts that contribute to this difference:

* 48 building with "No Glazing Area" have been removed; we experimented with models that included them and the errors on these buildings were the largest and our concern was that they biased the models and we couldn't answer the research questions effectively
* The "original study" repeated the cross-validation many times and the standard error for their MSE was 1.6, which puts us roughly at 2 standard errors from their average MSE
* A newer random forest package was used that didn't exist when the authors of the original study did their study in 2011.

### The best performing model

Both models: Random Forest and CART have very similar R-squared (Random Forest has 0.9689 and CART has 0.9722)

The best performing Model from the Initial Results seems to be Random Forest since it has a lower MSE at 2.79 compared to CART that has 5.59

In case of the best linear model, the R-squared is 0.906 what brings it on the last place.

### Step 7: Answering the Research Questions

**Can the cooling load be predicted accurately with a statistical model?**

As a result of the initial analysis, yes this can be done by using the Random Forest statistical model. The error is small relative to the values of the predicted variable and the R2 looks very good (over 0.97).

**Explore trade-offs between accuracy and interpretability of the model**

It can be seen that while CART is more interpretable, random forest gives a much lower error. The easy way to see how the prediction is calculated has to be given up for a more accurate prediction.

For the linear model, splitting variables into groups produces a more accurate prediction, at the expense of interpretability. Figuring out what happens when those variables change is harder to do when there are two coefficients associated with each variable. Also, the AIC recommended to pick a model with fewer variables instead of the model with all the variables and a slightly bigger R2.

Although from an engineering perspective, the most important variable should be the Glazing Area, in the random forest it is almost insignificant, while in CART and the linear model it is considered important. This means that these models might still be useful, despite the lower accuracy because they capture the real relationships better.

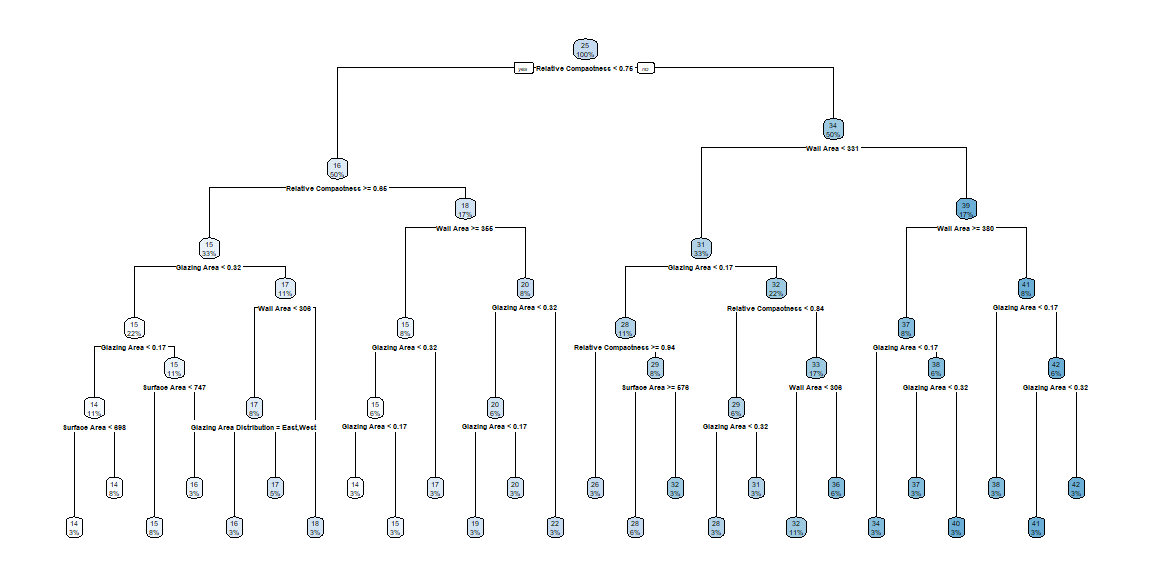
**Which are the most influential variables when predicting the cooling efficiency?**

All the models will be reviewed to answer this question and then compare the results:

*CART interpretation*

As was mentioned in the beginning, CART is a very interpretable model and even a non-technical person can look at the tree and follow the branches to see how a prediction was made and which variables influenced it.

rpart.plot(CARTmodel, uniform = T)



When we look at the tree it seems that Relative Compactness, Wall Area and Glazing Area are the most important variables, as they make up the first three levels. We can see the Surface Area and Glazing Area Distribution show up a few times on the last 2 levels. We don't see at all the Roof Area, Orientation and Overall Height.

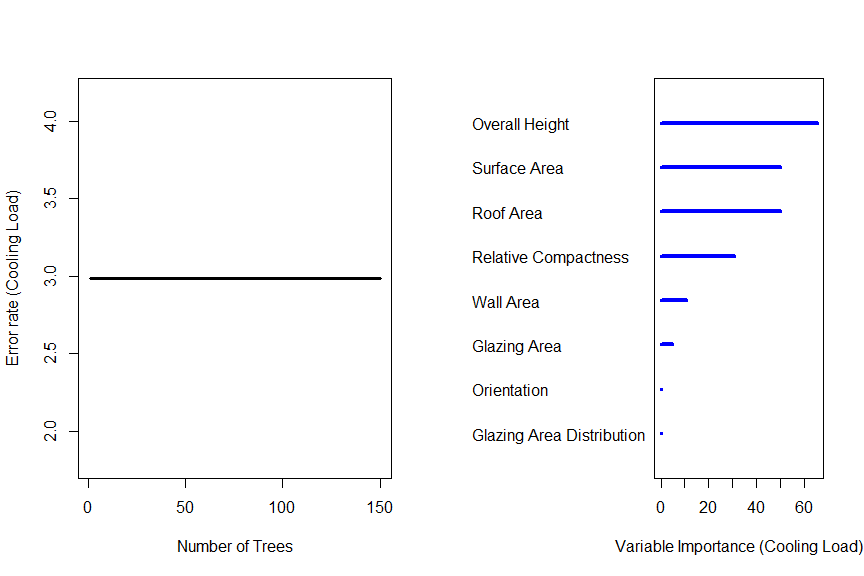
The fact that the Orientation is less important resulted also in another study that was referenced in the Literature Review section: "Modelling energy efficiency performance of residential buildings stocks based on Bayesian statistical model" - by Marta Braulio-Gonzalo, Pablo Juan, Maria D. Bovea, Maria Jose Rua. This referenced study was working with another parameters (like shape factor of the building and year of construction) where it was specified that "The least significant (parameter) is the solar orientation of the main facade of the building"

*Random Forest interpretation*

Because aside from making accurate predictions, this study investigates which predictors have the biggest influence on the response variable, a more simple way will be used for this, to avoid plotting and examining all the trees individually.

vimp is a function from the randomForestSRC package that computes the variable importance, similar to the coefficient p-values in linear models.

plot(vimp(RFmodel))



##   
## Importance Relative Imp  
## Overall Height 65.4935 1.0000  
## Surface Area 49.9843 0.7632  
## Roof Area 49.9532 0.7627  
## Relative Compactness 30.7670 0.4698  
## Wall Area 10.5729 0.1614  
## Glazing Area 4.8273 0.0737  
## Orientation 0.0810 0.0012  
## Glazing Area Distribution 0.0482 0.0007

As we can see from the variable importance plot, the Random Forest model gives different results. The most important variable is the Roof Area, followed closely by the Surface Area, while the Overall Height and Relative Compactness have an average importance. Wall Area and Glazing area have minor influence on the predictions, while Orientation and Glazing Area Distribution don't seem to matter for this model.

*Linear Model interpretation*

The variables used by the model and their units of measure:

* Relative Compactness / no units
* Surface Area / m2
* Wall Area / m2
* Overall Height / m
* Glazing Area / no unit

summary(finalLM)

##   
## Call:  
## lm(formula = `Cooling Load` ~ ., data = select(modelData, -`Roof Area`,   
## -`Glazing Area Distribution`, -Orientation))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.7773 -1.6774 -0.3235 1.2857 11.2717   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 99.37181 21.15591 4.697 3.17e-06 \*\*\*  
## `Relative Compactness` -70.82433 11.43868 -6.192 1.00e-09 \*\*\*  
## `Surface Area` -0.09037 0.01898 -4.761 2.33e-06 \*\*\*  
## `Wall Area` 0.04655 0.00739 6.298 5.26e-10 \*\*\*  
## `Overall Height` 4.23411 0.37574 11.269 < 2e-16 \*\*\*  
## `Glazing Area` 13.25292 0.96109 13.789 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.158 on 714 degrees of freedom  
## Multiple R-squared: 0.8906, Adjusted R-squared: 0.8898   
## F-statistic: 1162 on 5 and 714 DF, p-value: < 2.2e-16

The formula used to make a prediction will be:

The formula was obtained by putting the numeric values of the coefficients in the regression equation.

p-values were considered when doing variable selection but they will not be analysed further. This study recommends a more in depth analysis that takes the effect size into account.

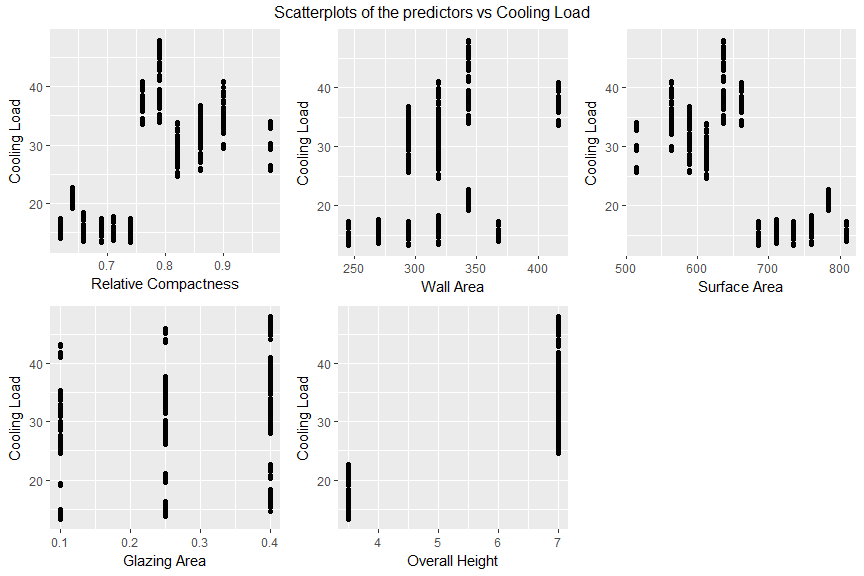
*The most important variables*

The most important variables will be mostly determined by the best performing model, the random forest and these are Surface Area, Overall Height, Relative Compactness and Roof Area. It can be seen that some of these variables are important in the other models as well.

**Investigate energy impact when changing some building parameters**

The following scatterplots will be used to determine for which variables the change impact on the Cooling Load can be analyzed.

p1 <- ggplot(modelData, aes(y = `Cooling Load`)) +  
 geom\_point(aes(x = `Relative Compactness`))  
  
p2 <- ggplot(modelData, aes(y = `Cooling Load`)) +  
 geom\_point(aes(x = `Wall Area`))  
  
p3 <- ggplot(modelData, aes(y = `Cooling Load`)) +  
 geom\_point(aes(x = `Surface Area`))  
  
p4 <- ggplot(modelData, aes(y = `Cooling Load`)) +  
 geom\_point(aes(x = `Glazing Area`))  
  
p5 <- ggplot(modelData, aes(y = `Cooling Load`)) +  
 geom\_point(aes(x = `Overall Height`))  
  
grid.arrange(p1, p2, p3, p4, p5, ncol = 3, nrow = 2,   
 top = "Scatterplots of the predictors vs Cooling Load")



It is clear from the scatterplots that Glazing Area, Overall Height have very few values and Wall Area has such an irregular pattern that extrapolating outside those values is infeasible. It cannot be assumed that only the Surface Area can change, because other variables like Relative Compactness cannot stay constant when it changes because Relative Compactness is inversely proportional to the Surface Area.

The nature of the dataset makes any independent numeric variable change analysis infeasible.

# Conclusions

The cooling load can be predicted accurately using a statistical model, but the best performing model does not capture the real relationships between the variables adequately (Glazing Area is insignificant in random forest for example).

An attempt was made to study the impact of changing predictors on the cooling load, but the nature of the dataset made this infeasible. A future study might be able to do this with more real(non-simulated) data.

This study concludes that the random forest should be used in together with one of the more interpretable models (CART or linear) by using random forest to make predictions and the other model to figure out what variables influenced the prediction.

In conclusion, these results will help building owners, architects, engineers, and other stakeholders to optimize the energy efficiency of the HVAC equipment in their buildings.At the same time those models can help urban planners, developers, and local authorities to identify residential areas that require a higher energy demands and where residential buildings require more urgent energy efficiency refurbishments.