CKME136 Capstone Project / Fall 2017

Student: Walter Cios; Student ID 500791945

# 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, I 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 by Angeliki Xifara & Athanasios 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).

# 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 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(reshape2)  
library(rpart)  
library(rpart.plot)  
library(randomForestSRC)  
library(caret)

### Loading the data

After examining the original Excel file, I 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)

### Display structure of "raw\_data"

The structure 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 can be done for the whole dataset

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

## [1] 0

Or as an alternative, we could also have done an NA check per columns. As expected found none.

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

Review str for "data" and keep the raw\_data in the workspace in case it's needed later

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

We see values like "0" (No Glazing Area) which never apply to residential buildings and might hinder us in answering our research questions by influencing the model fit, since there is not possible that a house has no windows and we will remove those "0" tuples from the Glazing Area Distribution

By removing the 48 tuples with "0" Glazing Area Distribution we are removing also all the "0" in "Glazing Area".

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

## [1] 1 2 3 4 5

Now Glazing Area Distribution has only 5 values and "0" has been removed

### Converting categorical variables to factors

As per the dataset description, I 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 data dictionary from the authors we created mappings from the integers in the raw data 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 summmary we can see 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

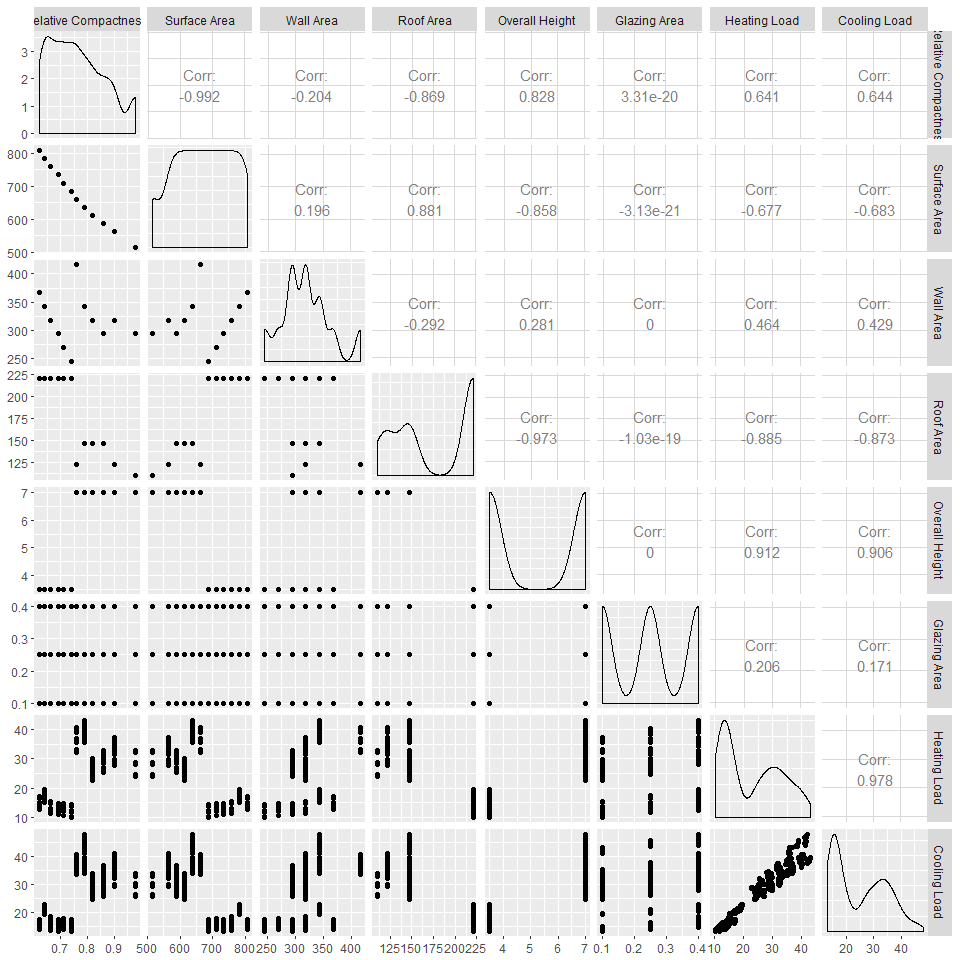
Split up the categorical variables from the continuous ones, into a new dataframe "data\_categorical" and keep the continuos variables in "data\_numeric".

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 I 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 corelation

Wall Area & Heating Load / low postitive 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 I will try to use only ggplot for all graphs to create an consistent view

Looking at the Excel Table "Variable Review for Analysis" that has the Min and Max, I 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

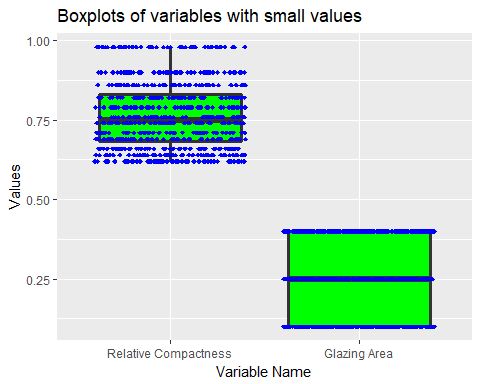
I used a couple of features to make the boxplots better visible on paper:

* The boxplots were plotted using ggplot2;
* In the geom\_boxplot I filled the inside "green";
* For the median, I 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

We are splitting the numeric data in 3 data frames because the variables have different scales.

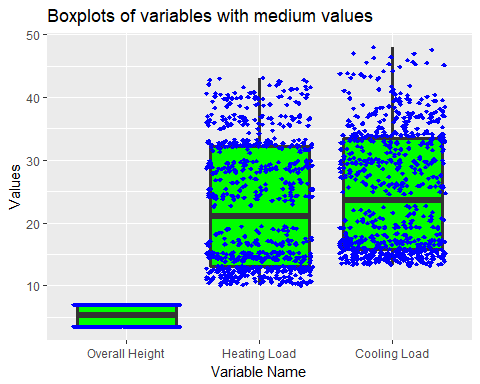
We also used melt from the reshape2 package because the boxplots have the variable name on the x-axis and the long data format(one column with the variable name and another one with the value) can be easily used by ggplot2 to create the side-by-side boxplots.

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 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.

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")



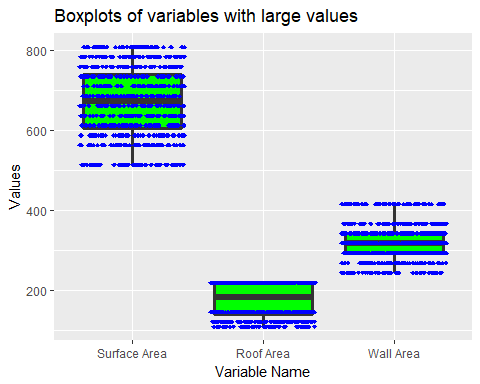
The boxplots of variables with medium values are showing that we have no outliers and that there are several data points clustered in the bottom part of the boxplot of the Heating Load and Cooling Load (between 10 and 15).

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 variabilitry 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 are showing that we have no outliers and that Surface Area has only 12 possible values, Wall Area only 7 values and Roof Area only 4 values.

### Creating a Correlation Matrix

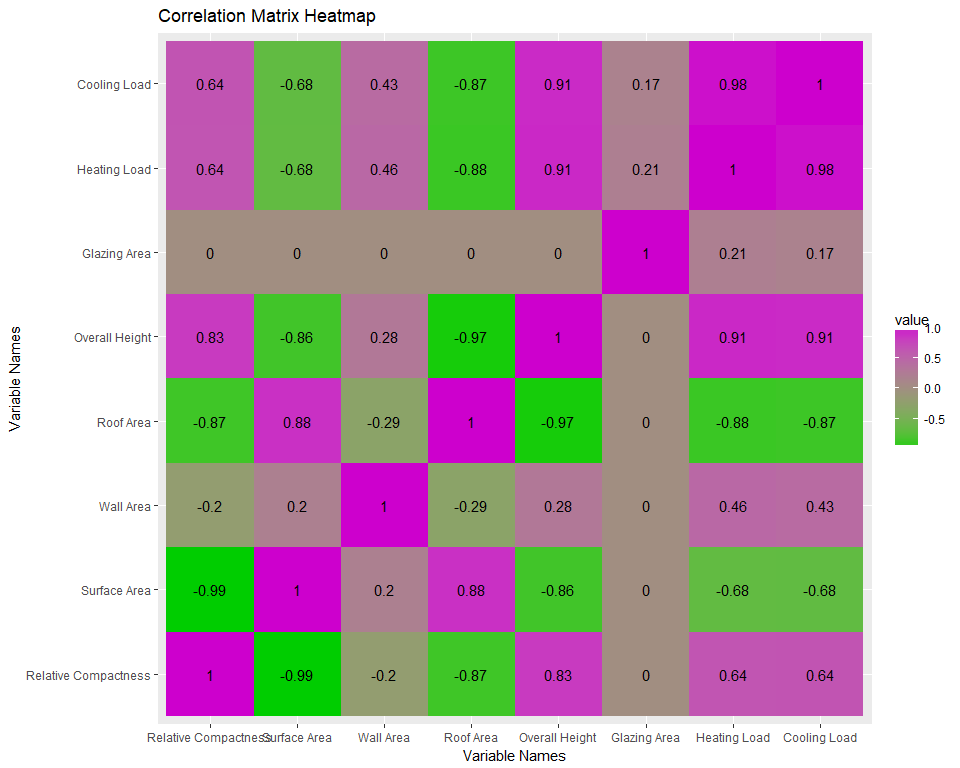
I created a Correlation Matrix Heatmap to further display the correlation of the continuous variables

I used a couple of features 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>
* I obtained 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

"Orientation" that has only the four values of the cardinal directions: East, North, South, West

table(data\_categorical$Orientation)

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

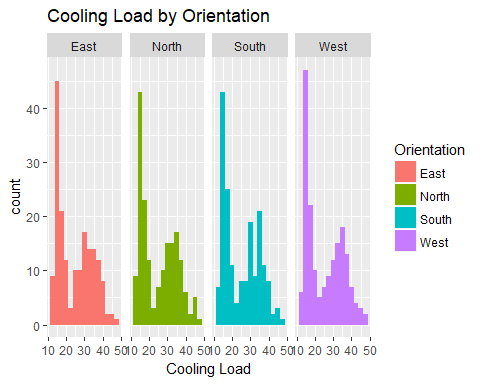
The tables are displayed for both categorical variable "Orientation" and "Glazing Are Distribution"to show that the "Glazing Area Distribution" has the four values of the cardinal directions and includes also"Uniform"

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

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

Next we will use facet\_grid to plot the histograms of the Cooling Load for each value of the categorical variables.

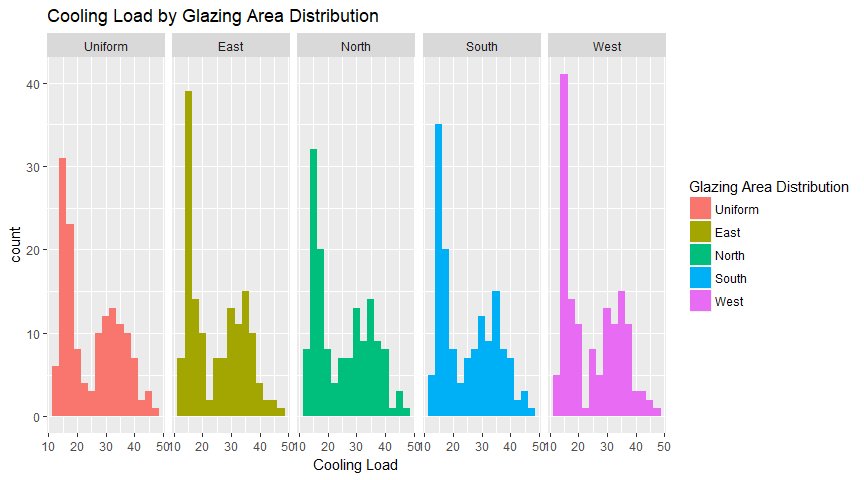
plotData <- select(data, `Cooling Load`, `Orientation`)  
  
ggplot(plotData, aes(x = `Cooling Load`, fill = `Orientation`)) +  
 geom\_histogram(binwidth = 2.5) +  
 facet\_grid(~ Orientation) +  
 ggtitle("Cooling Load by Orientation")



Plot the categorical variable "Orientation" agains 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 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, we cannot do this.

# Step 3 Define Model Evaluation Procedure

The most important criterion for choosing the best model will be R2. For linear models, we will not consider adjusted R2, but we will try to use information criteria (AIC and/or BIC). For non-linear models like CART and Random Forest, we will use k-fold cross-validation (k = 10) to pick the best performing model according to R2. We will also use MSE and MAE as evaluation measures because they were used by the authors of the original study.

We will also consider the interpretability of the models and how can they help us to answer the original research questions.

From the two output variables we are predicting the "Cooling Load" only (that was marked in the original study with "y2"), and this is why we exclude from the data the "Heating Load" from the modelData.

We will use set.seed 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.

Our first priority is to replicate the original results, so we start with random forest and 10-fold cross-validation. We will also use a classical train-test split(70-30) for other models.

The following code block contains the configuration of the model evaluation procedures, so we can easily change 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

### Short Definitions of Evaluation Measures

R-squared is the percentage of the response variable variation that is explained by a model. R-squared is always between 0 and 100%: 0% indicates that the model explains none of the variability of the response data around its mean; and 100% indicates that the model explains all the variability of the response data around its mean. In general we can say that the higher the R-squared, the better the model fits the data.

The Mean Absolute Error (MAE) and Root mean squared error (RMSE) are two of the most common metrics to measure accuracy for continuous variables. The Mean Absolute Error (MAE) measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all differences have equal weight.

The Root Mean Squared Error (RMSE) is a quadratic scoring rule that also measure the average magnitude of error. It's the square root of the average of squared differences between prediction and actual observation.

# Step 4 Models

We will start with random forest with 10-fold cross-validation because this is what the authors of the original study used and we will try to replicate their results first.

Steps 5 and 6 will de done individually for each model.

# Step 4.3 Random Forest

The authors didn't mention which parameters they used for random forest, so we will do a grid search and use the cross-validation results to pick the best performing model. We will use a simple grid search for the hyperparameter selection. In the grid we picked values such that the best performing model is not at the extremes. The intervals were picked after several trial runs.

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 we will compute several measures (R2, MSE, MAE) and we will take the mean over all folds. Because we are fitting 162 models, we will only display the best 10 and the worst 5, sorted by R2 in descending order.

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

We will now fit the model that performed best in cross-validation on the whole dataset.

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])

# Step 4.2 CART

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

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)

We will display the CART model performance similar to what we did for random forest.

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

We now fit the CART model on the whole dataset.

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

### Step 4.1 Linear Model

We will try to fit a linear model on the data, but more investigation is needed on how to handle the nonlinear relationships(Relative Compactness vs Cooling Load for example).

### 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, and this is why we will not pursue this matter further.

### Model Comparison

We will now compare the results of all the models that we have trained until now.

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

As we see from that 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:

* We removed the 48 building with "No Glazing Area"; 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
* They 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
* We used a newer random forest package that didn't exist when they did their study(2011)

### The best performing model

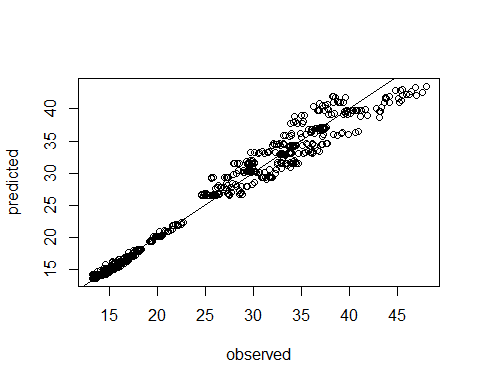
Both models: Random Forest and CART have very similar R-squared (Random Forest has 0.9689 nd CART has 0.9722)

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

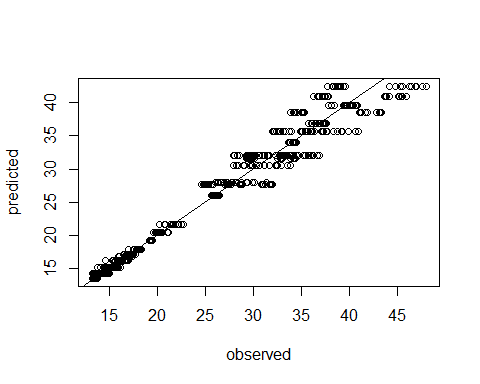
### A more in-depth look

Although the CART MSE is almost twice the random forest MSE, the CART R2 is surprisingly larger, so we want to have a more in-depth look at the predictions.

predicted <- predict(RFmodel)$predicted  
observed <- modelData$`Cooling Load`  
plot(observed, predicted)  
abline(0, 1)



predicted <- predict(CARTmodel)  
observed <- modelData$`Cooling Load`  
plot(observed, predicted)  
abline(0, 1)



After the Inintial Results submission, I will investigate the predictions with the largest errors and try to identify if there is a pattern or some easily identifiable conditions when the model makes mistakes.

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

# Step 7: Answering the Research Questions

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

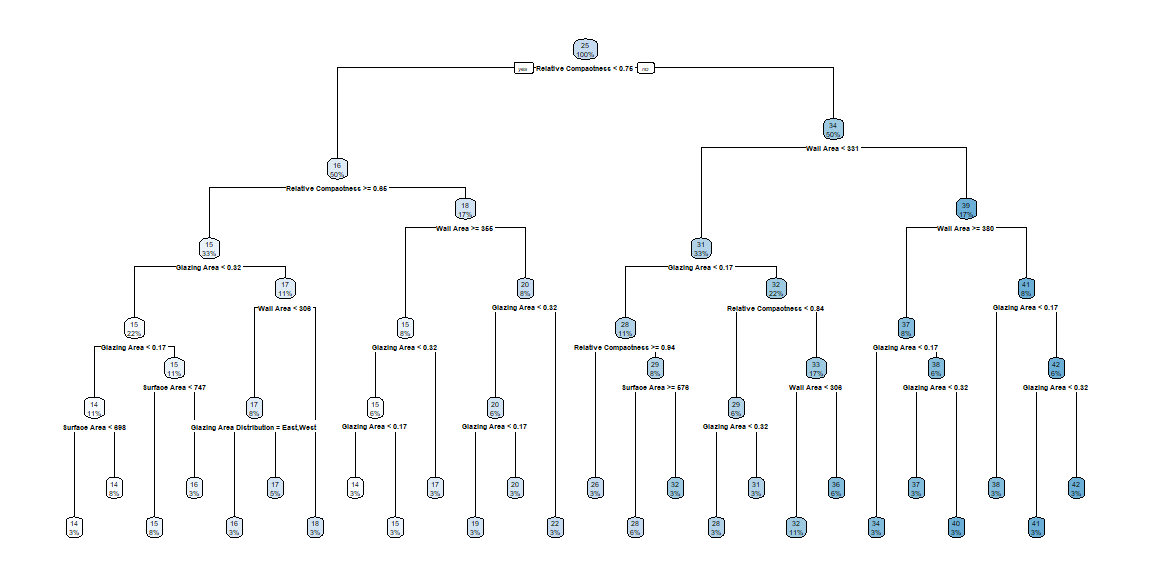
As a result of the initial analysis, yes this can be done byusing the Random Forest statistical model.

### Which are the most infuential variables when predicting the cooling efficiency?

We will try to use all the models to answer this question and then compare the results:

### CART

rpart.plot(CARTmodel, uniform = T)

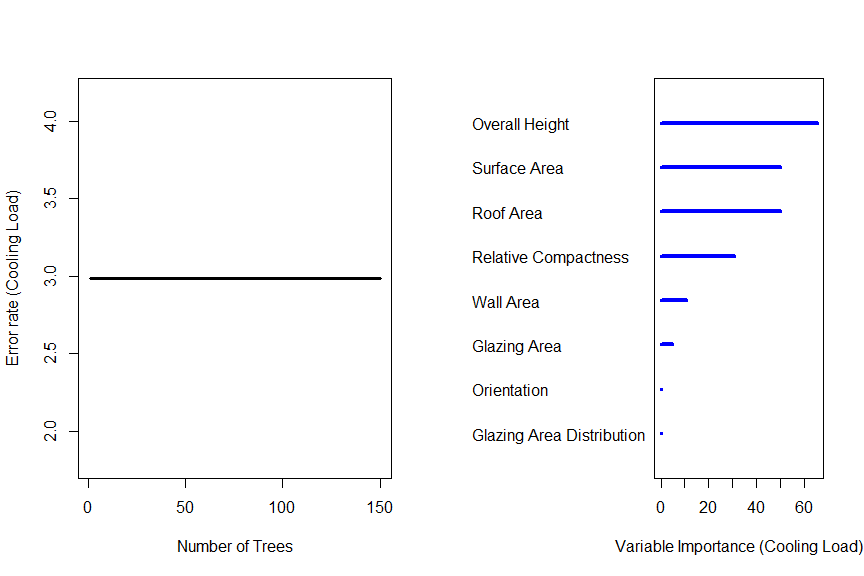


When we look at the tree it seems that Relative Compactness, Wall Area and Glazing Area are the most important variables, as they make up the first three levels. We can see the Surface Area and Glazing Area Distribution show up a few times on the last 2 levels. We don't see at all the Roof Area, Orientation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

### Random Forest

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.