Energy Efficiency

Khalil Zarifsadr (Kyle Zarif)

3/5/2020

https://github.com/kylezarif/Energy_Efficiency

Overview

Energy efficient buildings are those that are designed to reduce energy consumption. In building design heating load (HL) and cooling load (CL) is required to determine the specifications of the heating and cooling equipment. This study presents a machine learning framework that investigates characteristics including surface area, wall area, roof area, overall height, orientation, relative compactness, glazing area, and glazing area distribution to determine the heating load and cooling load in buildings.

In this study three machine learning algorithms are trained using the inputs in one subset (ee set) to predict heating load and cooling load in the validation set. The dataset contains eight features, denoted by X1,..,X8 and two outcomes, denoted by y1 and y2. The aim is to use the eight features to predict each of the two responses. For developing the algorithm, the 'ee' set is split into separate training set and test set. Heating load and cooling load are predicted in the validation set as if they were unknown. Since we have two outcomes, all sets are split in 2 separate sets, one for calculating heating load by dropping cooling load and one for calculating cooling load by dropping heating load. The confusion matrix is used for summarizing the performance and accuracy of each algorithm and evaluate how close the predictions are to the true values in the validation set.

Let's install required libraries and start by creating ee set, and validation set

```
# Energy Efficiency dataset:
# https://archive.ics.uci.edu/ml/machine-learning-databases/00242/
# https://archive.ics.uci.edu/ml/datasets/Energy+efficiency
#
https://raw.githubusercontent.com/kylezarif/Energy_Efficiency/master/ENB2012_data.x
lsx


# Download data
url <-
"https://raw.githubusercontent.com/kylezarif/Energy_Efficiency/master/ENB2012_data.
xlsx"
eefm <- read.delim(url)

# Updating the name of the columns
colnames(eefm) <- c("Relative_Compactness","Surface_Area","Wall_Area",
                    "Roof_Area", "Overall_Height", "Orientation", "Glazing_Area",
                    "Glazing_Area_Distribution", "Heating_Load", "Cooling_Load")
```

```r
# Let's see the class and number of missing values
data.frame(cbind(data.frame(VarType=sapply(eefm,class)),data.frame(Total_Missing=sa
pply(eefm,function(x){sum(is.na(x))}))))
```

```
##                           VarType Total_Missing
## Relative_Compactness      numeric             0
## Surface_Area              numeric             0
## Wall_Area                 numeric             0
## Roof_Area                 numeric             0
## Overall_Height            numeric             0
## Orientation               integer             0
## Glazing_Area              numeric             0
## Glazing_Area_Distribution integer             0
## Heating_Load              numeric             0
## Cooling_Load              numeric             0
```

According to the original project done by Angeliki Xifara and Athanasios Tsanas Let's round outputs Heating Load and Cooling Load to the nearest integer

```r
eefm <- eefm %>%
  mutate(Heating_Load = floor(Heating_Load), Cooling_Load = floor(Cooling_Load))
```

In this step the data from ee set is split into two sets. Training set with 80% of the eefm set and test set.

```r
library(e1071)
# Validation set will be 20% of eefm data
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
# if using R 3.5 or earlier, use `set.seed(1)` instead
n <- nrow(eefm)  # Number of observations
test_index <- round(n*0.80)  # 80% for ee set
set.seed(314)    # Set seed for reproducible results
tindex <- sample(n, test_index)   # Create a random index
ee <- eefm[tindex,]   # Create ee set
validation <- eefm[-tindex,]   # Create validation set

# Validation sets for Heating Load and Cooling Load
validation_hl <- select (validation,-c(Cooling_Load))
validation_cl <- select (validation,-c(Heating_Load))
```

Dividing the ee data into training and test sets

```r
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
# test set will be 20% of ee data
# if using R 3.5 or earlier, use `set.seed(1)` instead
n <- nrow(ee)  # Number of observations
test_index <- round(n*0.80)  # 80% for training set
set.seed(314)    # Set seed for reproducible results
tindex <- sample(n, test_index)   # Create a random index
train_set <- ee[tindex,]   # Create train set
test_set <- ee[-tindex,]   # Create test set

# train and test sets for Heating Load Model by removing Cooling Load
train_set_hl <- select (train_set,-c(Cooling_Load))
test_set_hl <- select (test_set,-c(Cooling_Load))

# train and test sets for Cooling Load Model by removing Heating Load
train_set_Cl <- select (train_set,-c(Heating_Load))
test_set_Cl <- select (test_set,-c(Heating_Load))

set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used

# test set will be 20% of ee data
# if using R 3.5 or earlier, use `set.seed(1)` instead
n <- nrow(ee)  # Number of observations
test_index <- round(n*0.80)  # 80% for training set
set.seed(314)    # Set seed for reproducible results
tindex <- sample(n, test_index)   # Create a random index
train_set <- ee[tindex,]   # Create train set
test_set <- ee[-tindex,]   # Create test set

# train and test sets for Heating Load Model by removing Cooling Load
train_set_hl <- select (train_set,-c(Cooling_Load))
test_set_hl <- select (test_set,-c(Cooling_Load))

# train and test sets for Cooling Load Model by removing Heating Load
train_set_Cl <- select (train_set,-c(Heating_Load))
test_set_Cl <- select (test_set,-c(Heating_Load))
```

Analysis

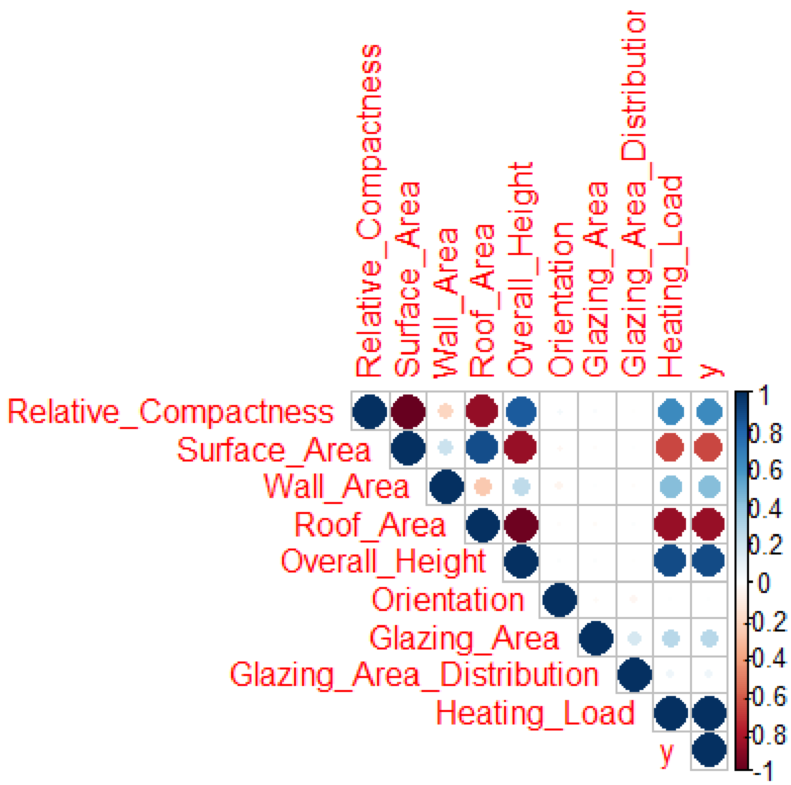
We split the training set into two partitions. The first partition for predicting heating load by droping cooling load , using eight features, and the other for predicting cooling by dropping only heating load. we will investigate the relationships and correlations between features and oucomes by applying several visualization methods.

Data Exploration and Visualization

Considering Heating Load

Let's see if there is any strong positive or negative correlation between variables by constructing a correlation plot

```
y.label <-as.numeric(train_set_hl$Heating_Load)
corrplot(cor(cbind(train_set_hl,y = train_set_hl$Heating_Load)),type="upper")
```




```
cor(train_set_hl,train_set_hl$Heating_Load)

##                                  [,1]
## Relative_Compactness       0.636736857
## Surface_Area              -0.669435924
## Wall_Area                  0.429497945
## Roof_Area                 -0.864617325
## Overall_Height             0.891350437
## Orientation                0.003045007
## Glazing_Area               0.278905956
## Glazing_Area_Distribution  0.064880689
## Heating_Load               1.000000000
```

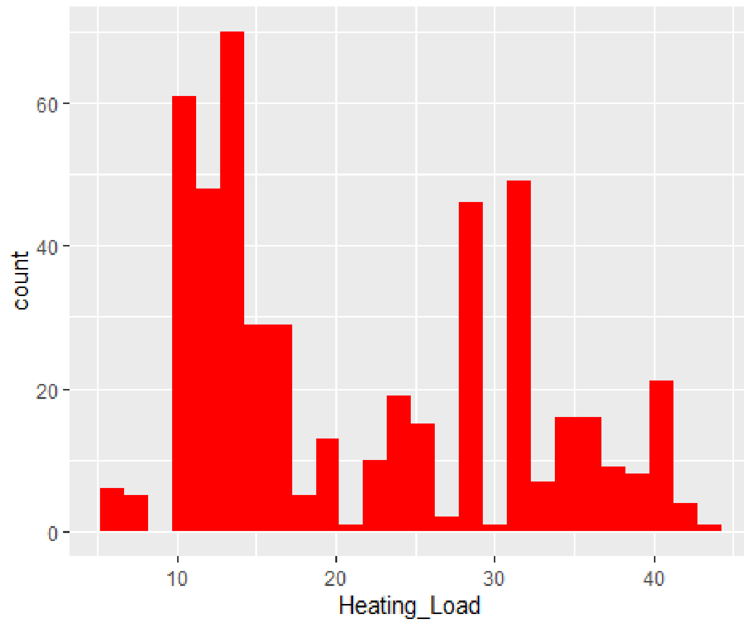
We can see that Overal_Height has the highest correlation with heating load followed by relative compactnes, wall area, and glazing area

```r
# Let's apply a Heating Load model to see P-values
model_hl <- lm(Heating_Load~.,data=train_set_hl)
summary(model_hl)
```

```
##
## Call:
## lm(formula = Heating_Load ~ ., data = train_set_hl)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.603 -1.282  0.004  1.320  8.073
##
## Coefficients: (1 not defined because of singularities)
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               80.311300  24.383861   3.294  0.00106 **
## Relative_Compactness     -62.963673  13.181058  -4.777 2.37e-06 ***
## Surface_Area              -0.084795   0.021979  -3.858  0.00013 ***
## Wall_Area                  0.059994   0.008803   6.816 2.81e-11 ***
## Roof_Area                        NA         NA      NA       NA
## Overall_Height             4.217379   0.441615   9.550  < 2e-16 ***
## Orientation                0.057225   0.122286   0.468  0.64002
## Glazing_Area              19.611062   1.024065  19.150  < 2e-16 ***
## Glazing_Area_Distribution  0.197343   0.089107   2.215  0.02725 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.006 on 483 degrees of freedom
## Multiple R-squared:  0.9128, Adjusted R-squared:  0.9116
## F-statistic: 722.5 on 7 and 483 DF,  p-value: < 2.2e-16
```

P-values for orientation, and roof area are all greater than 0.05. This means that the relationship between the dependent and these independent variables is not significant at the 95% certainty level. We can drop these 2 variables for the model. High p-values for these independent variables do not mean that they should not be used in the model. It could be that some other variables are correlated with these variables and making these variables less useful for prediction. Let’s drop these variables from heating load data set.

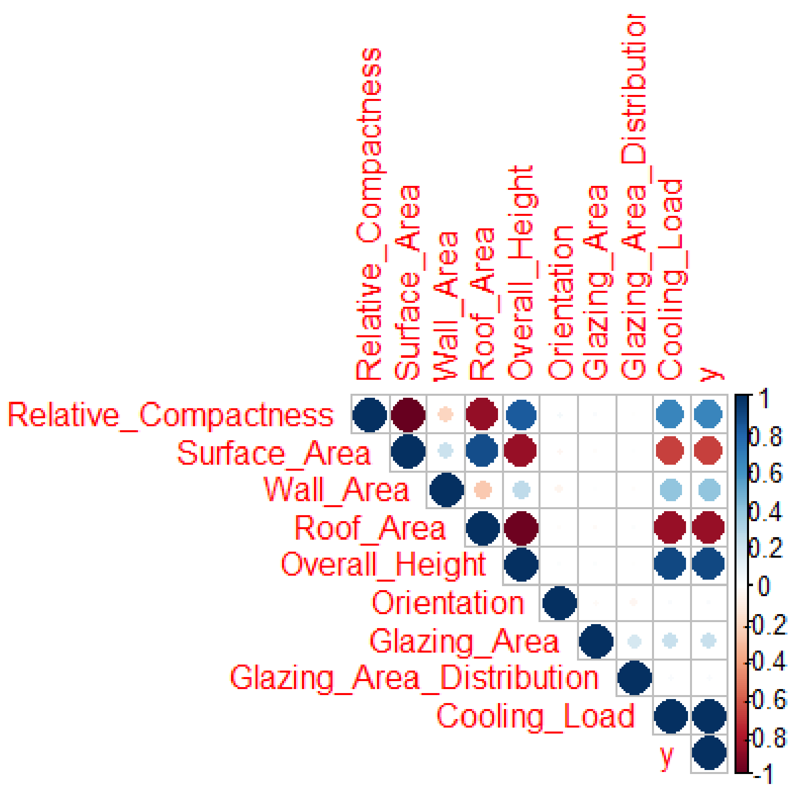
```r
train_set_hl <- select (train_set_hl, -c(Roof_Area), -c(Orientation))

# Histogram of Heating Load
train_set_hl %>%
  ggplot(aes(Heating_Load))+
  geom_histogram(binwidth = 1.5, fill = 'red')
```



Considering Cooling Load

Let's see if there is any strong positive or negative correlation between variables by constructing a correlation plot

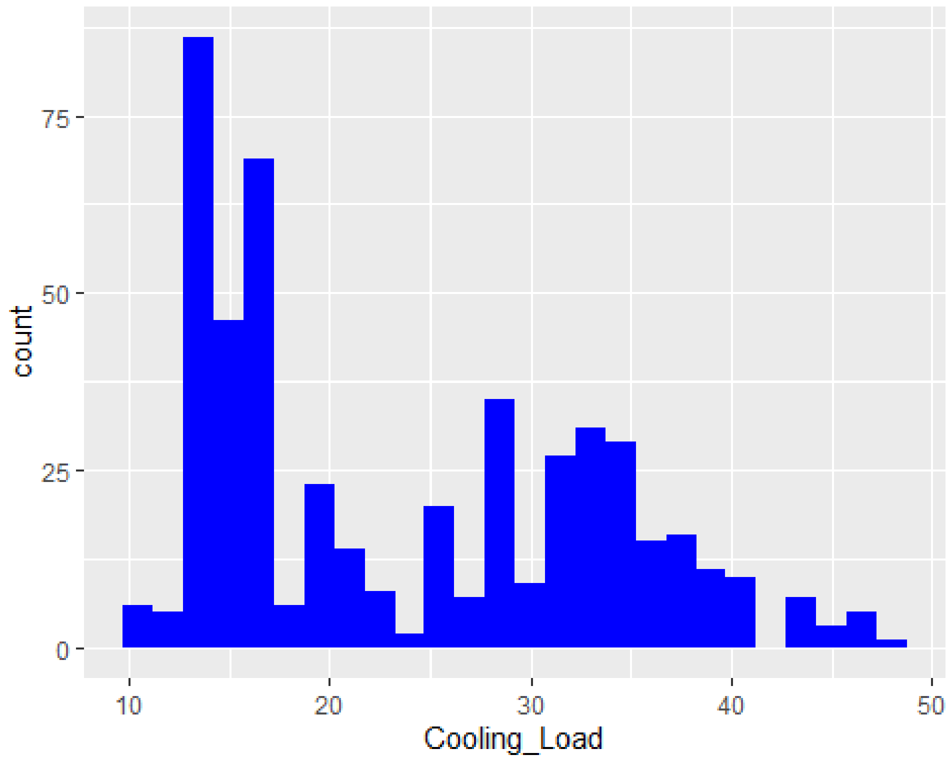
```
y.label <-as.numeric(train_set_Cl$Cooling_Load)
corrplot(cor(cbind(train_set_Cl,y = train_set_Cl$Cooling_Load)),type="upper")
```



```r
cor(train_set_Cl,train_set_Cl$Cooling_Load)

##                                 [,1]
## Relative_Compactness       0.65114680
## Surface_Area              -0.68746588
## Wall_Area                  0.39956961
## Roof_Area                 -0.86814433
## Overall_Height             0.90083963
## Orientation                0.02223762
## Glazing_Area               0.22366656
## Glazing_Area_Distribution  0.02759879
## Cooling_Load               1.00000000

# We can see that overal height has the highest correlation with cooling load
followed by relative compactnes and wall area

# Let's apply a cooling load model to see P-values
model_cl <- lm(Cooling_Load~.,data=train_set_Cl)
summary(model_cl)

##
## Call:
## lm(formula = Cooling_Load ~ ., data = train_set_Cl)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.584 -1.623 -0.354  1.339 11.203
##
## Coefficients: (1 not defined because of singularities)
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               101.772993  25.449946   3.999 7.36e-05 ***
## Relative_Compactness      -73.878750  13.757346  -5.370 1.22e-07 ***
## Surface_Area               -0.092664   0.022940  -4.039 6.23e-05 ***
## Wall_Area                   0.044749   0.009187   4.871 1.51e-06 ***
## Roof_Area                         NA         NA      NA       NA
## Overall_Height              4.310914   0.460923   9.353  < 2e-16 ***
## Orientation                 0.179801   0.127633   1.409    0.160
## Glazing_Area               15.082987   1.068838  14.112  < 2e-16 ***
## Glazing_Area_Distribution   0.007407   0.093003   0.080    0.937
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.137 on 483 degrees of freedom
## Multiple R-squared:  0.8944, Adjusted R-squared:  0.8928
## F-statistic: 584.3 on 7 and 483 DF,  p-value: < 2.2e-16
```

P-values for glazing area distribution, orientation, and roof area are all greater than 0.05. This means that the relationship between the dependent and these independent variables is not significant at the 95% certainty level. Let's drop these variables from cooling load data sets.

```r
train_set_Cl <- select (train_set_Cl, -c(Roof_Area), -c(Orientation), -
c(Glazing_Area_Distribution))

# Histogram of Cooling Load
train_set_Cl %>%
  ggplot(aes(Cooling_Load))+
  geom_histogram(binwidth = 1.5, fill = 'blue')
```

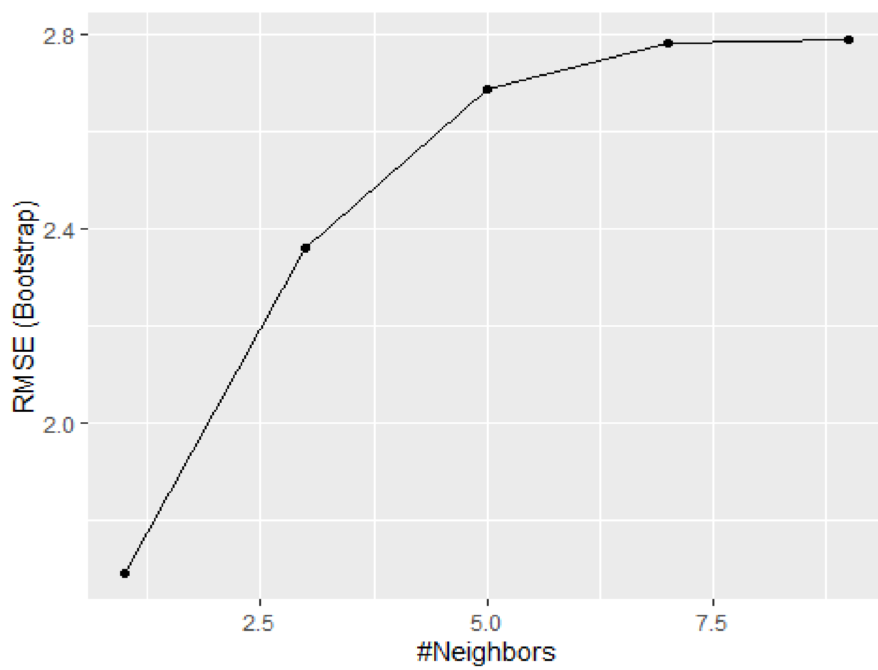


k-nearest neighbors (kNN)

Let’s start by fitting a K-nearest neighbors method (kNN) on train sets.

Considering Heating Load

```r
fit_knn <- train(Heating_Load ~ ., method = "knn",
                 tuneGrid = data.frame(k = seq(1,9,2)),
                 data = train_set_hl)

ggplot(fit_knn) #We see the number of neighbors that minimizes RMSE
```



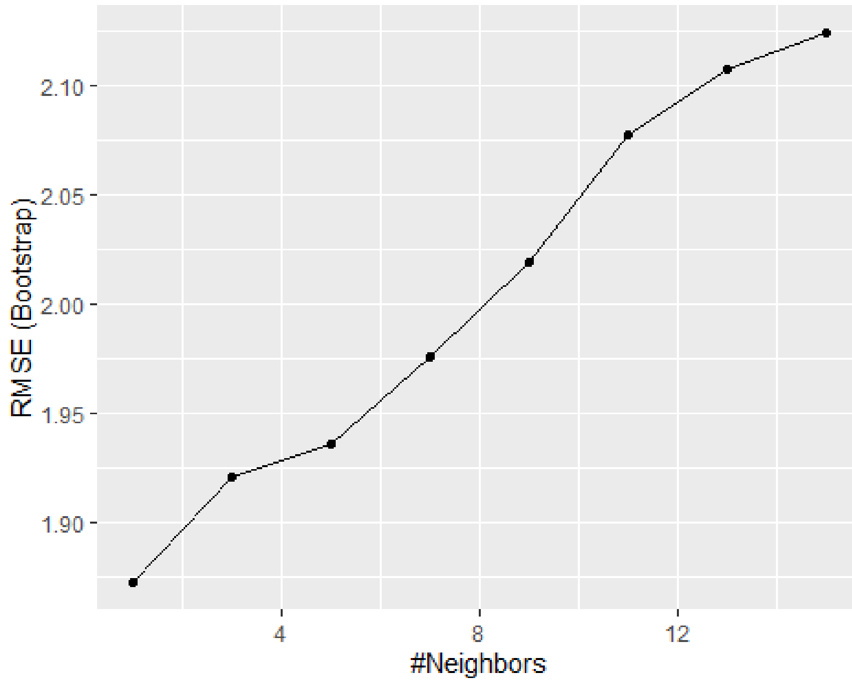
Let's change how we perform cross validation and apply 10-fold cross validation

```
control <- trainControl(method = "cv", number = 10, p = .9)
fit_knn <- train(Heating_Load ~ ., method = "knn",
                 data = train_set_hl,
                 tuneGrid = data.frame(k = seq(1,9,2)),
                 trControl = control)
fit_knn$bestTune #maximizes the accuracy

##   k
## 1 1

# Predict KNN on the test set
knn_hat <- predict(fit_knn,test_set_hl)

# Let's check the accuracy of the model
confusionMatrix(table(factor(knn_hat,
levels=min(test_set_hl$Heating_Load):max(test_set_hl$Heating_Load)),
                      factor(test_set_hl$Heating_Load,
levels=min(test_set_hl$Heating_Load):max(test_set_hl$Heating_Load))))$overall["Accu
racy"]

##  Accuracy
## 0.8089888

#The accuracy is 80%
```

Considering Cooling Load

```
fit_knn_cl <- train(Cooling_Load ~ ., method = "knn",
                    tuneGrid = data.frame(k = seq(1,15,2)),
```

```
                    data = train_set_Cl)
ggplot(fit_knn_cl)
```



Let's change how we perform cross validation and apply 10-fold cross validation

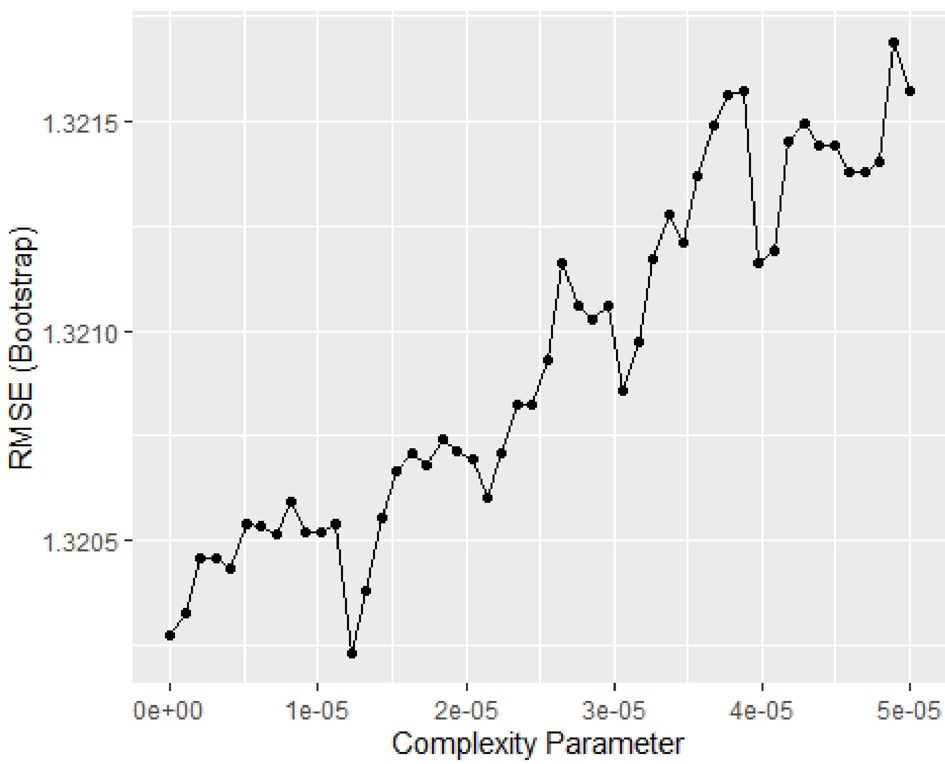
```
control <- trainControl(method = "cv", number = 10, p = .9)
fit_knn_cl <- train(Cooling_Load ~ ., method = "knn",
                    data = train_set_Cl,
                    tuneGrid = data.frame(k = seq(1,9,2)),
                    trControl = control)
fit_knn$bestTune #maximizes the accuracy

##   k
## 1 1

# Predict on the test set
knn_hat_cl <- predict(fit_knn_cl,test_set_Cl)

# Let's check the accuracy of the model
confusionMatrix(table(factor(knn_hat_cl,
levels=min(test_set_Cl$Cooling_Load):max(test_set_Cl$Cooling_Load)),
                      factor(test_set_Cl$Cooling_Load,
levels=min(test_set_Cl$Cooling_Load):max(test_set_Cl$Cooling_Load))))$overall["Accu
racy"]

##  Accuracy
## 0.8461538

#The accuracy is 84%
```

Regression trees

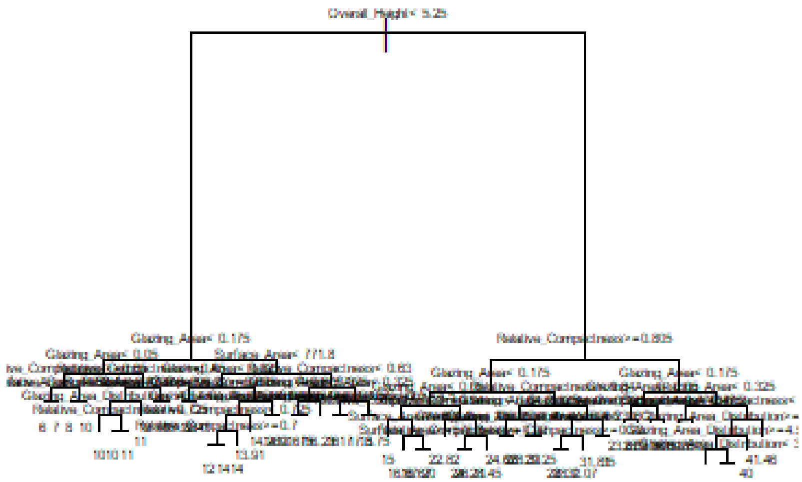
Here I will use regression tree which is a very strong method for continuous outcomes. On training datasets for more flexibility, I will find the best complexity parameter (cp). Note that cp = 0 is the most flexible value that will results in over training. cp = 0 means predictor is the original data.

Considering Heating Load

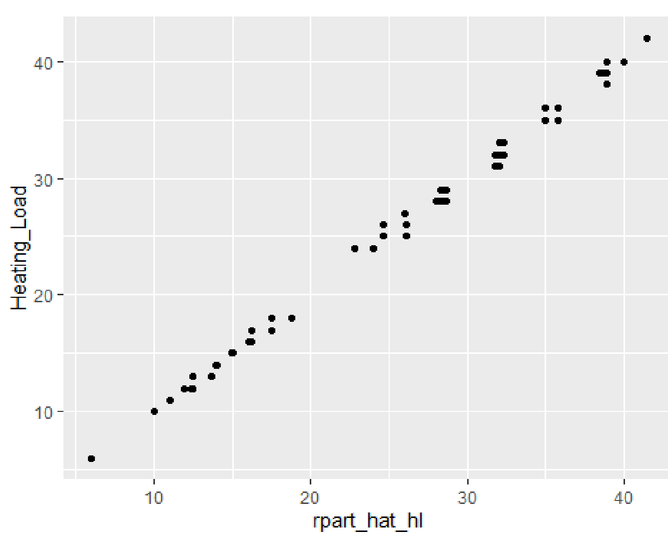
```
library(rpart)
fit_rpart <- rpart(Heating_Load ~ ., data = train_set_hl)

# To pick the best complexity problem and avoid over training:
train_rpart_hl <- train(Heating_Load ~ .,
                        method = "rpart",
                        tuneGrid = data.frame(cp = seq(0,0.00005,len = 50)),
                        data = train_set_hl)
ggplot(train_rpart_hl)
```




```
fit_rpart <- rpart(Heating_Load ~ ., data = train_set_hl, control =
rpart.control(cp = 0.000025, minsplit = 2))
plot(fit_rpart, margin = 0.01)
text(fit_rpart,cex = 0.4)
```



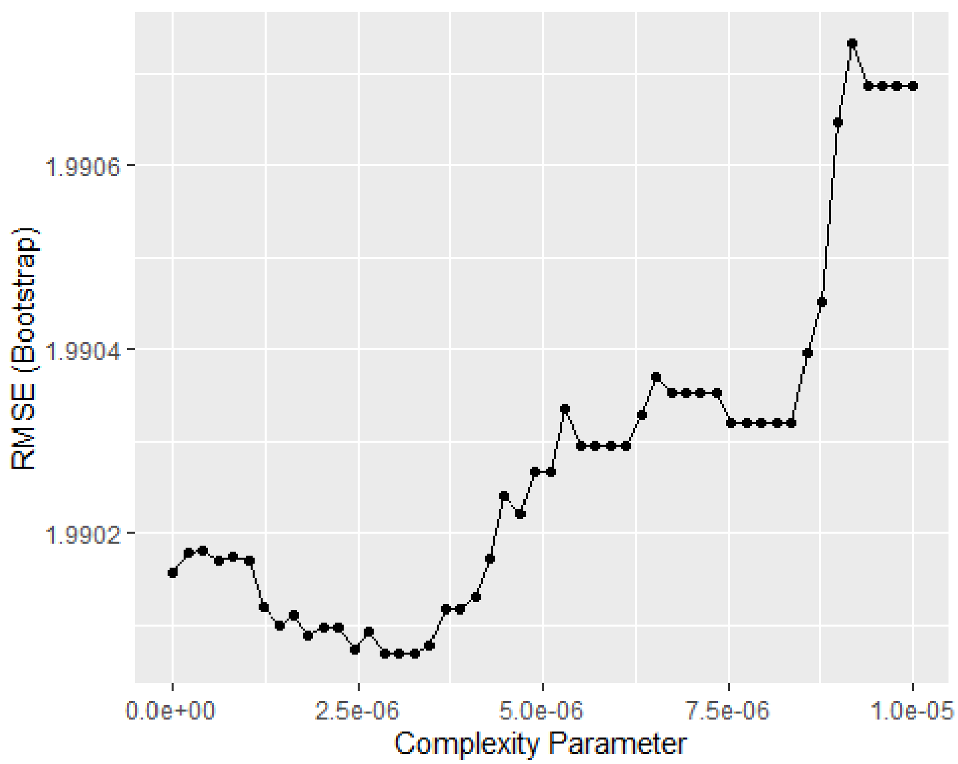

```r
# Let's apply the model on the test set
rpart_hat_hl <- predict(fit_rpart, test_set_hl)

# Let's plot predicted values versus heating load on the test set
test_set_hl %>%
  ggplot(aes(rpart_hat_hl,Heating_Load))+
  geom_point()
```



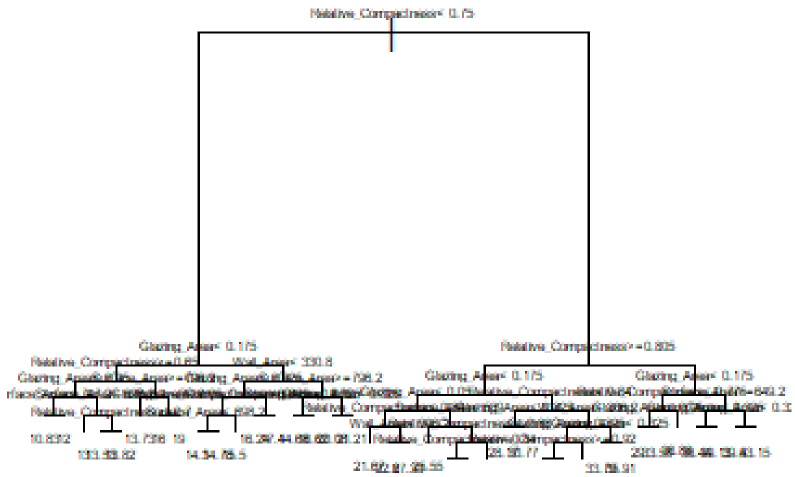
```r
# Let's check the accuracy of the model
confusionMatrix(table(factor(rpart_hat_hl,
levels=min(test_set_hl$Heating_Load):max(test_set_hl$Heating_Load)),
                      factor(test_set_hl$Heating_Load,
levels=min(test_set_hl$Heating_Load):max(test_set_hl$Heating_Load))))$overall["Accu
racy"]

##  Accuracy
## 0.9166667

# We have improvement and the accuracy is higher equal to 91%
```

Considering Cooling Load

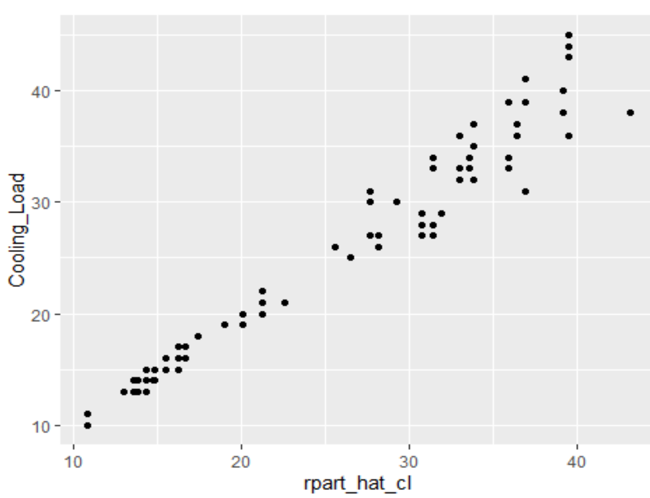
```r
fit_rpart_cl <- rpart(Cooling_Load ~ ., data = train_set_Cl)

# To pick the best complexity problem and avoid over training:
train_rpart_cl <- train(Cooling_Load ~ .,
                        method = "rpart",
                        tuneGrid = data.frame(cp = seq(0,0.00001,len = 50)),
                        data = train_set_Cl)
ggplot(train_rpart_cl)
```




```r
fit_rpart_cl <- rpart(Cooling_Load ~ ., data = train_set_Cl, control =
rpart.control(cp = 0.0000075, minsplit = 8))
plot(fit_rpart_cl, margin = 0.01)
text(fit_rpart_cl,cex = 0.4)
```



```r
# Let's apply the model on the test set
rpart_hat_cl <- predict(fit_rpart_cl, test_set_Cl)

# Let's plot predicted values versus heating load on the test set
test_set_Cl %>%
  ggplot(aes(rpart_hat_cl,Cooling_Load))+
  geom_point()
```



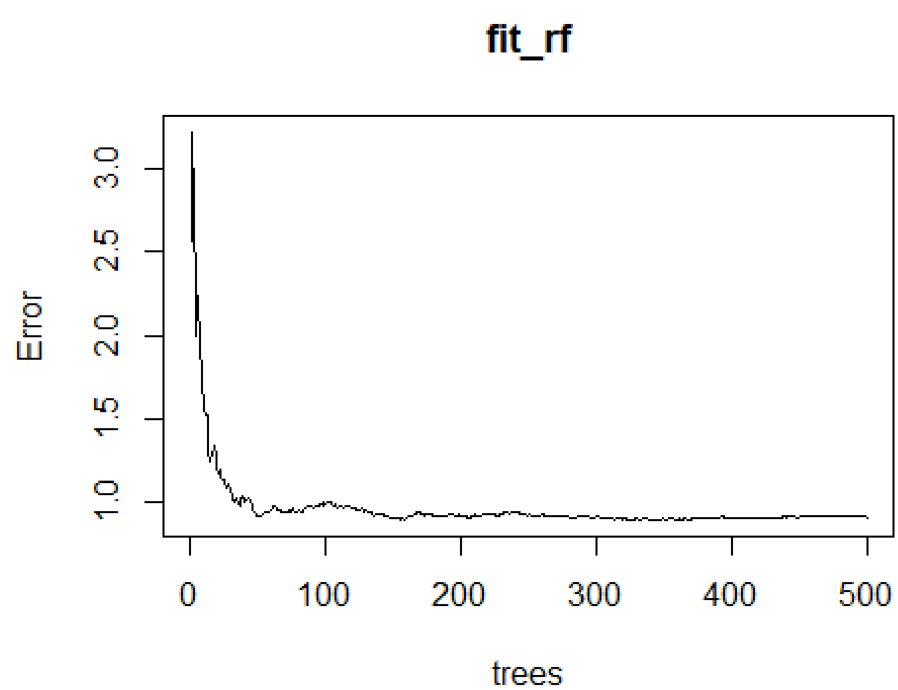
```r
# Let's check the accuracy of the model
confusionMatrix(table(factor(rpart_hat_cl,
levels=min(test_set_Cl$Cooling_Load):max(test_set_Cl$Cooling_Load)),
                      factor(test_set_Cl$Cooling_Load,
levels=min(test_set_Cl$Cooling_Load):max(test_set_Cl$Cooling_Load))))$overall["Accu
racy"]

## Accuracy
##        1

# We have improvement and the accuracy is higher equal to 100%
```

Random Forest

Let's check anothet method. The random forest aggregates predictions made by multiple decision trees. Each decision tree in the forest is trained on a subset of the dataset (bootstrapped dataset). The bootstrap makes random different individual trees. It may improve predictions of heating load and cooling load.

Considering Heating Load

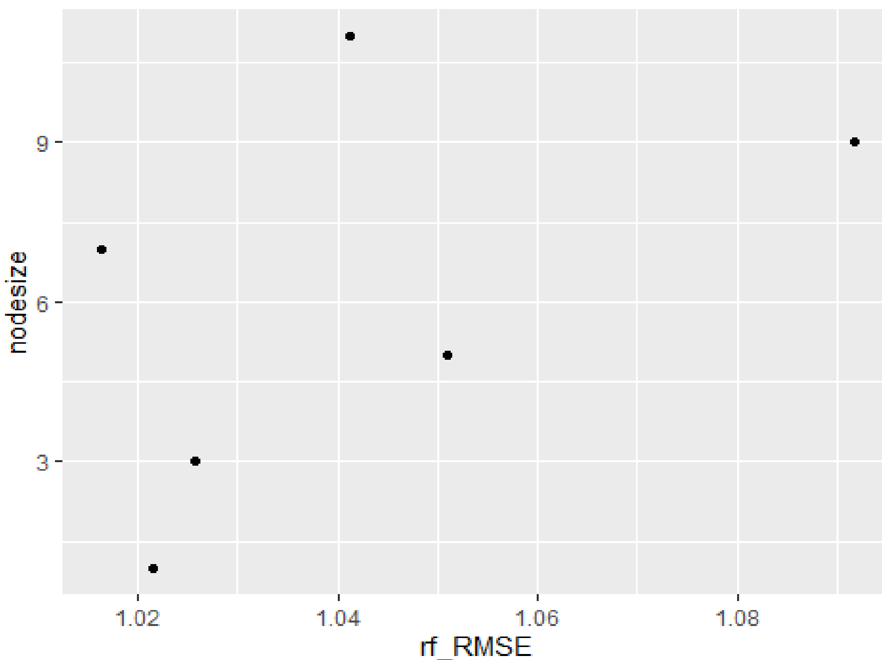
```r
fit_rf <- randomForest(Heating_Load~., data = train_set_hl)
plot(fit_rf)
```




```r
# We can see in this case the accuracy improves as we add more trees untill about
90 trees
```

We can make the estimates smoother by changing the parameters that controls the minimum number of data points in the nodes of the tree. The smaller RMSE, the smoother the final estimates will be.

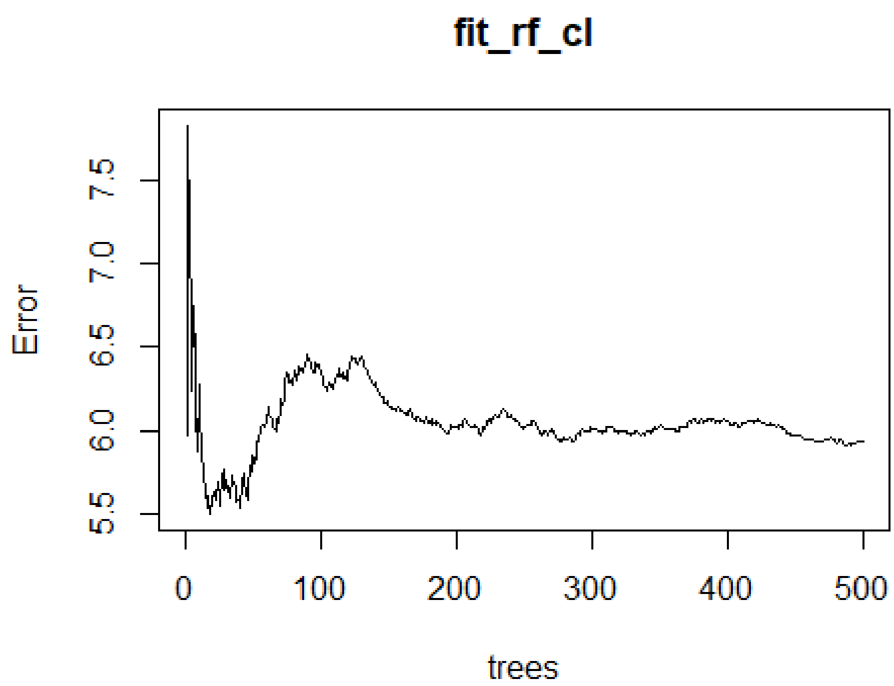
```
nodesize <- seq(1,11,2)
rf_RMSE <- sapply(nodesize, function(ns){
  train( Heating_Load ~., method = "rf", data = train_set_hl,
         tuneGrid = data.frame(mtry = 2),
         nodesize = ns)$results$RMSE
})
qplot(rf_RMSE,nodesize)
```




```
# To predict Heating Load using the optimized random forest method:
fit_rf <- randomForest(Heating_Load~., data = train_set_hl, nodesize =
nodesize[which.min(rf_RMSE)])

# Let's fit the model on the test set
rf_hat_hl <- predict(fit_rf, test_set_hl)
rf_hat_hl <- floor(rf_hat_hl) # to round results to the nearest integer

confusionMatrix(table(factor(rf_hat_hl,
levels=min(test_set_hl$Heating_Load):max(test_set_hl$Heating_Load)),
                      factor(test_set_hl$Heating_Load,
levels=min(test_set_hl$Heating_Load):max(test_set_hl$Heating_Load))))$overall["Accu
racy"]

##  Accuracy
## 0.4552846

#The accuracy is low
```

Considering Cooling Load

```
fit_rf_cl <- randomForest(Cooling_Load~., data = train_set_Cl)
plot(fit_rf_cl)
```




```
# We can see in this case the accuracy improves as we add more trees untill about 70 trees
```

We can make the estimates smoother by changing the parameters that controls the minimum number of data points in the nodes of the tree. The smaller RMSE, the smoother the final estimates will be.

```
rf_RMSE_cl <- sapply(nodesize, function(ns){
  train( Cooling_Load ~., method = "rf", data = train_set_Cl,
         tuneGrid = data.frame(mtry = 2),
         nodesize = ns)$results$RMSE
})

# To predict Cooling Load using the optimized random forest method:
fit_rf_cl <- randomForest(Cooling_Load~., data = train_set_Cl, nodesize =
nodesize[which.min(rf_RMSE_cl)])

# Let's fit the model on the test set
rf_hat_cl <- predict(fit_rf_cl, test_set_Cl)
rf_hat_cl <- floor(rf_hat_cl) # to round results to the nearest integer

confusionMatrix(table(factor(rf_hat_cl,
```

```
levels=min(test_set_Cl$Cooling_Load):max(test_set_Cl$Cooling_Load)),
                      factor(test_set_Cl$Cooling_Load,
levels=min(test_set_Cl$Cooling_Load):max(test_set_Cl$Cooling_Load))))$overall["Accu
racy"]

##  Accuracy
## 0.2276423

#The accuracy is low
```

Results

The machine learning framework is ready for validation. In this section, I will fit all three proposed machine learning models on the validation sets separately. Note that the validation set is like an unknown dataset up to now.

Validation of Regression trees

For Heating load prediction let's validate the model on the validation set for regression trees (CART)

```
rpart_hat <- predict(fit_rpart, validation_hl)
RT_HL <- confusionMatrix(table(factor(rpart_hat,
levels=min(validation_hl$Heating_Load):max(validation_hl$Heating_Load)),
                               factor(validation_hl$Heating_Load,
levels=min(validation_hl$Heating_Load):max(validation_hl$Heating_Load))))$overall["
Accuracy"]

accuracy_results <- tibble(method = "RT_HL", Accuracy = RT_HL)
accuracy_results %>% knitr::kable()
```

method	Accuracy
RT_HL	0.8909091

For Cooling load prediction let's validate the model on the validation set for regression trees (CART)

```
rpart_hat_cl <- predict(fit_rpart_cl, validation_cl)
RT_CL <- confusionMatrix(table(factor(rpart_hat_cl,
levels=min(validation_cl$Cooling_Load):max(validation_cl$Cooling_Load)),
                               factor(validation_cl$Cooling_Load,
levels=min(validation_cl$Cooling_Load):max(validation_cl$Cooling_Load))))$overall["
Accuracy"]

accuracy_results <- bind_rows(accuracy_results,
                              tibble(method="RT_CL",
                                     Accuracy = RT_CL ))
accuracy_results %>% knitr::kable()
```

method	Accuracy
RT_HL	0.8909091
RT_CL	1.0000000

Validation of KNN

For heating load prediction let's validate the model on the validation set for KNN

```
knn_hat <- predict(fit_knn,validation_hl)
KNN_HL <- confusionMatrix(table(factor(knn_hat,
levels=min(validation_hl$Heating_Load):max(validation_hl$Heating_Load)),
                                factor(validation_hl$Heating_Load,
levels=min(validation_hl$Heating_Load):max(validation_hl$Heating_Load))))$overall["
Accuracy"]

accuracy_results <- bind_rows(accuracy_results,
                              tibble(method="KNN_HL",
                                     Accuracy = KNN_HL ))
accuracy_results %>% knitr::kable()
```

method	Accuracy
RT_HL	0.8909091
RT_CL	1.0000000
KNN_HL	0.7962963

For Cooling load prediction let's validate the model on the validation set for KNN

```
knn_hat_cl <- predict(fit_knn_cl,validation_cl)
KNN_CL <- confusionMatrix(table(factor(knn_hat_cl,
levels=min(validation_cl$Cooling_Load):max(validation_cl$Cooling_Load)),
                                factor(validation_cl$Cooling_Load,
levels=min(validation_cl$Cooling_Load):max(validation_cl$Cooling_Load))))$overall["
Accuracy"]

accuracy_results <- bind_rows(accuracy_results,
                              tibble(method="KNN_CL",
                                     Accuracy = KNN_CL ))
accuracy_results %>% knitr::kable()
```

method	Accuracy
RT_HL	0.8909091
RT_CL	1.0000000
KNN_HL	0.7962963
KNN_CL	0.8947368

Validation of Random Forest

For heating load prediction let's validate the model on the validation set with Random Forest

```
rf_hat_hl <- predict(fit_rf, validation_hl)
rf_hat_hl <- floor(rf_hat_hl)

RF_HL <- confusionMatrix(table(factor(rf_hat_hl,
levels=min(validation_hl$Heating_Load):max(validation_hl$Heating_Load)),
                               factor(validation_hl$Heating_Load,
levels=min(validation_hl$Heating_Load):max(validation_hl$Heating_Load))))$overall["
Accuracy"]

accuracy_results <- bind_rows(accuracy_results,
                              tibble(method="RF_HL",
                                     Accuracy = RF_HL))
accuracy_results %>% knitr::kable()
```

method	Accuracy
RT_HL	0.8909091
RT_CL	1.0000000
KNN_HL	0.7962963
KNN_CL	0.8947368
RF_HL	0.4220779

For cooling load prediction let's validate the model on the validation set with Random Forest

```
rf_hat_cl <- predict(fit_rf_cl, validation_cl)
rf_hat_cl <- floor(rf_hat_cl)

RF_CL <- confusionMatrix(table(factor(rf_hat_cl,
levels=min(validation_cl$Cooling_Load):max(validation_cl$Cooling_Load)),
                               factor(validation_cl$Cooling_Load,
levels=min(validation_cl$Cooling_Load):max(validation_cl$Cooling_Load))))$overall["
Accuracy"]

accuracy_results <- bind_rows(accuracy_results,
                              tibble(method="RF_CL",
                                     Accuracy = RF_CL))
accuracy_results %>% knitr::kable()
```

method	Accuracy
RT_HL	0.8909091
RT_CL	1.0000000
KNN_HL	0.7962963
KNN_CL	0.8947368

RF_HL	0.4220779
RF_CL	0.1948052

Conclusion

The available data set includes discrete continues data and the distribution of predictors and outcomes were all non-normal. The proposed machine learning models estimate heating load and cooling load with high accuracy using characteristics of buildings in case a similar dataset is available. The proposed methods were applied on the validation sets and shows high accuracy of 89% for heating load, and 100% for cooling load using regression trees followed by 79% accuracy for heating load, and 89% accuracy for cooling load using k-nearest neighbors (kNN). However, results from random forest method are not likely. The accuracy of the predicted heating load is 42% and the accuracy of the predicted cooling load is only 19% using random forest. For future studies it is recommended to train an Artificial Neural Network model. ANN has more flexibility on multi-class problems. If we approach the project as a multi-class classification problem, Multi-Class Support Vector Machine (SVM) is also another flexible method tend to perform well in a wide range of problems.

Citation:

A. Tsanas, A. Xifara: 'Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools', Energy and Buildings, Vol. 49, pp. 560-567, 2012 The dataset was created by Angeliki Xifara (angxifara '@' gmail.com, Civil/Structural Engineer) and was processed by Athanasios Tsanas (tsanasthanasis '@' gmail.com, Oxford Centre for Industrial and Applied Mathematics, University of Oxford, UK).