building_energy

Summary:

In this notebook I have leveraged the research paper published Athanasios Tsanas & Angeliki Xifara to develop a machine learning model which could take into consideration 8 different parameters for a building and determine the heating and cooling load.

Inputs are:

Relative Compactness, Surface Area, Wall Area, Roof Area, Overall Height, Orientation , Glazing area, Glazing area distribution

Reference

Research paper available at https://www.sciencedirect.com/science/article/abs/pii/S037877881200151X (https://www.sciencedirect.com/science/article/abs/pii/S037877881200151X)

Install & Load H2o Library

```
#install h2o package
#install.packages("h2o")

#load h2o package
library(h2o)
```

```
## Warning: package 'h2o' was built under R version 3.6.3
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai
##
## ----------------------------------------------------------------------
```

```
##
## Attaching package: 'h2o'
```

```
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
```

```
## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

```
#initialize h2o which will start a JVM process and return a reference to it
localH2o <- h2o.init(nthreads = -1)
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         9 hours 54 minutes
##     H2O cluster timezone:       Asia/Kolkata
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.30.0.1
##     H2O cluster version age:    2 months and 8 days
##     H2O cluster name:           H2O_started_from_R_Aditya_Jain_spq261
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   5.28 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 3.6.2 (2019-12-12)
```

Change current work directory, load data file and convert to h2o frame

```
#set the current working directory
setwd('C:\\MachineLearning\\repos\\harvardx\\Data-Science-With-R\\R-BuildingEnergy Model')

#Load the input data csv file as R dataframe
data.r <- read.csv('ENB2012_data.csv')

#convert R dataframe to H2o frame
data <- as.h2o(data.r)
```

```
## Warning in use.package("data.table"): data.table cannot be used without R
## package bit64 version 0.9.7 or higher. Please upgrade to take advangage of
## data.table speedups.
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
#see top entries in the dataframe
head(data)
```

```
##     X1    X2    X3     X4 X5 X6 X7 X8    Y1    Y2
## 1 0.98 514.5 294.0 110.25  7  2  0  0 15.55 21.33
## 2 0.98 514.5 294.0 110.25  7  3  0  0 15.55 21.33
## 3 0.98 514.5 294.0 110.25  7  4  0  0 15.55 21.33
## 4 0.98 514.5 294.0 110.25  7  5  0  0 15.55 21.33
## 5 0.90 563.5 318.5 122.50  7  2  0  0 20.84 28.28
## 6 0.90 563.5 318.5 122.50  7  3  0  0 21.46 25.38
```

Split data in training & test

```
#convert column X6 abd X8 as features
factorsList <- c("X6", "X8")
data[,factorsList] <- as.factor(data[,factorsList])

#split the dataframe into train and test dataset in a 80:20 ratio
splits <- h2o.splitFrame(data, 0.8)
train <- splits[[1]]
test <- splits[[2]]
```

Print summary of the input data to

```
#summary of the input data
summary(data.r)
```

```
##        X1               X2              X3              X4
##  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
##        X5             X6             X7              X8              Y1
##  Min.   :3.50   Min.   :2.00   Min.   :0.0000   Min.   :0.000   Min.   : 6.01
##  1st Qu.:3.50   1st Qu.:2.75   1st Qu.:0.1000   1st Qu.:1.750   1st Qu.:12.99
##  Median :5.25   Median :3.50   Median :0.2500   Median :3.000   Median :18.95
##  Mean   :5.25   Mean   :3.50   Mean   :0.2344   Mean   :2.812   Mean   :22.31
##  3rd Qu.:7.00   3rd Qu.:4.25   3rd Qu.:0.4000   3rd Qu.:4.000   3rd Qu.:31.67
##  Max.   :7.00   Max.   :5.00   Max.   :0.4000   Max.   :5.000   Max.   :43.10
##        Y2
##  Min.   :10.90
##  1st Qu.:15.62
##  Median :22.08
##  Mean   :24.59
##  3rd Qu.:33.13
##  Max.   :48.03
```

Print correlation across the features in the dataset

```
#install.packages("corrplot")
library('corrplot')
```

```
## Warning: package 'corrplot' was built under R version 3.6.3
```

```
## corrplot 0.84 loaded
```

```
corresult<- cor(data.r)

#Positive correlations are displayed in blue and negative correlations in red color. Color inten
sity and the
#size of the circle are proportional to the correlation coefficients. In the right side of the c
orrelogram,
#the legend color shows the correlation coefficients and the corresponding colors.
corrplot(corresult, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)

#install.packages("PerformanceAnalytics")
library("PerformanceAnalytics")
```

```
## Warning: package 'PerformanceAnalytics' was built under R version 3.6.3
```

```
## Loading required package: xts
```

```
## Warning: package 'xts' was built under R version 3.6.3
```

```
## Loading required package: zoo
```

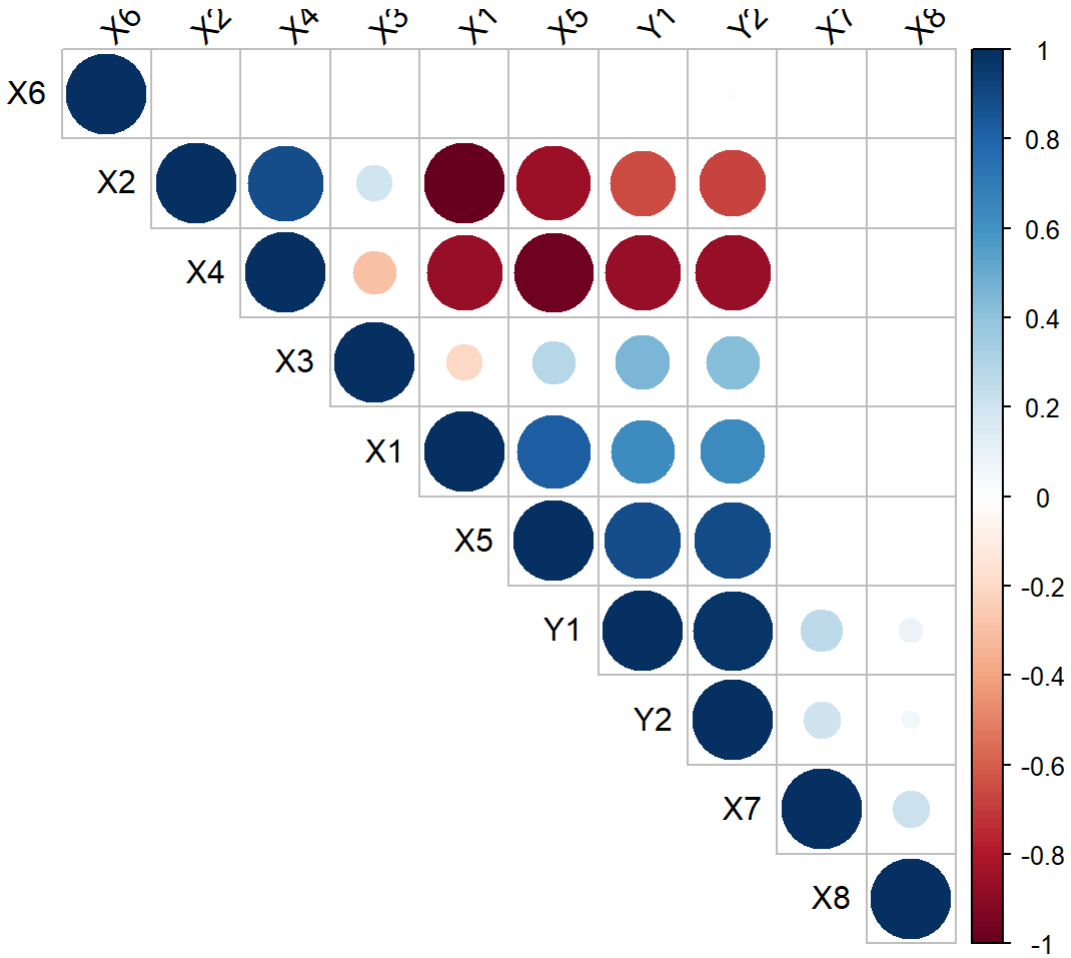
```
## Warning: package 'zoo' was built under R version 3.6.3
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

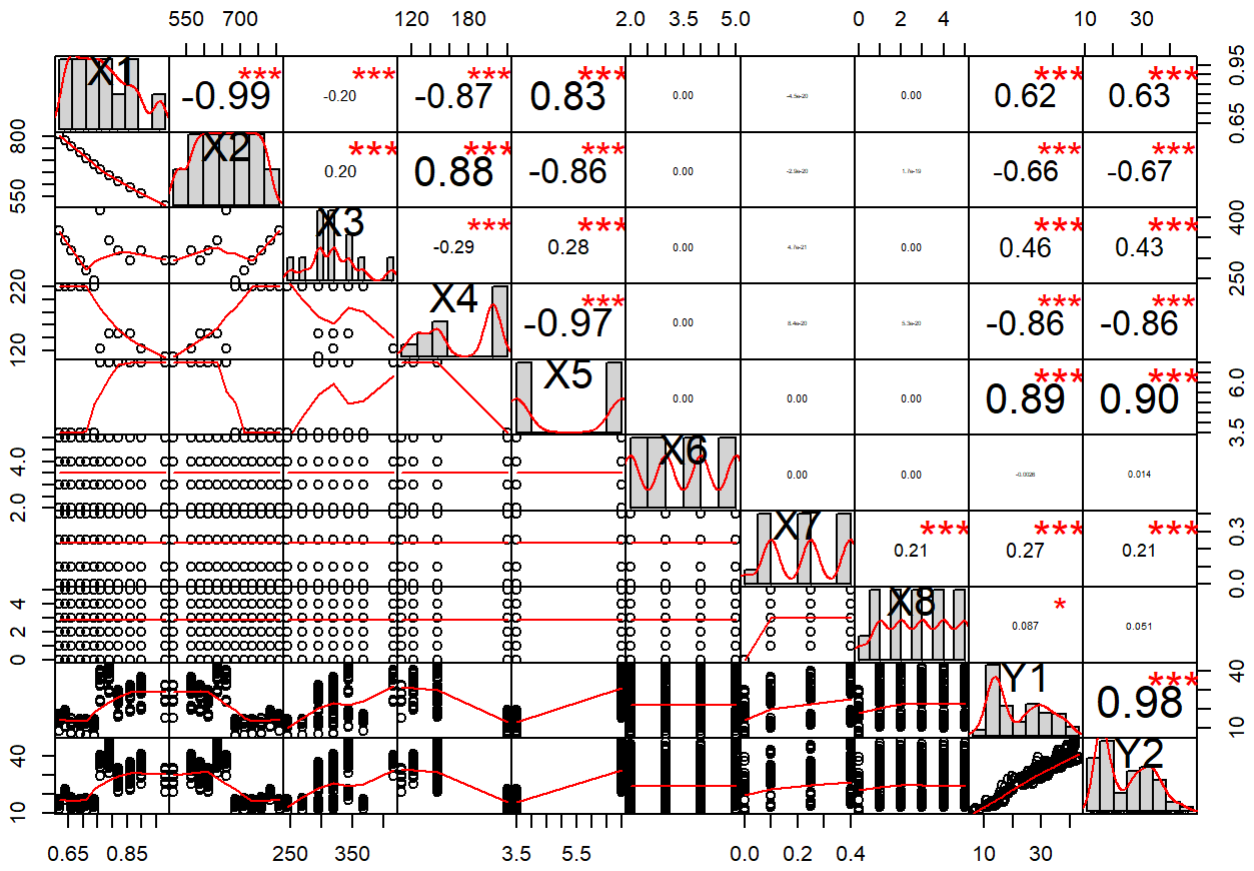
```
##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

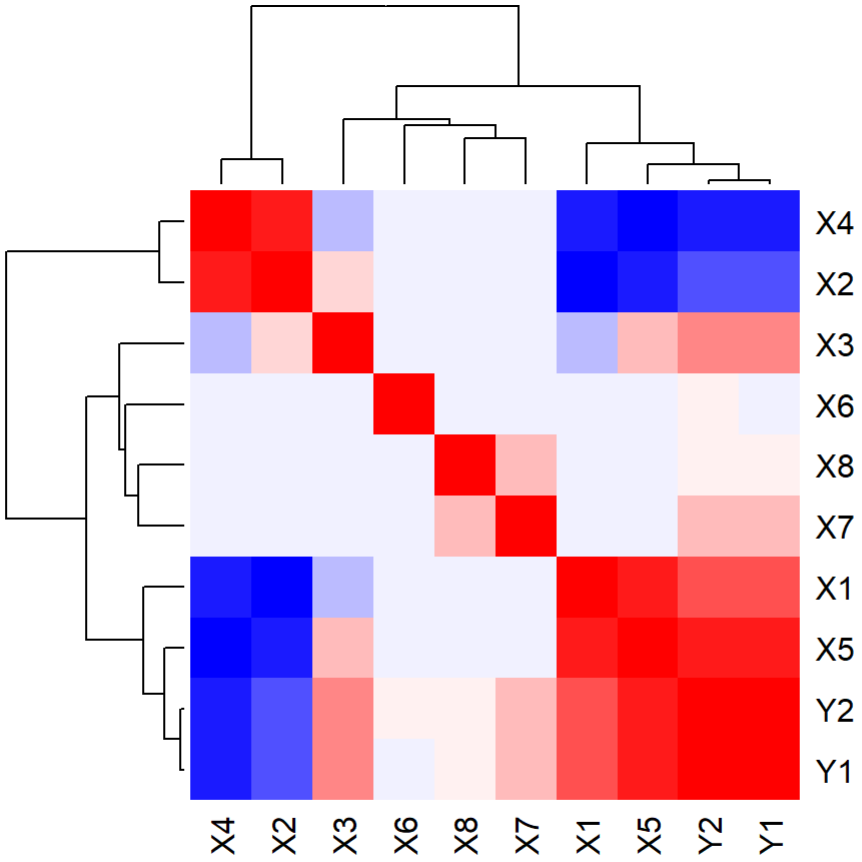


```
#In the above plot:

#  The distribution of each variable is shown on the diagonal.
#  On the bottom of the diagonal : the bivariate scatter plots with a fitted line are displayed
#  On the top of the diagonal : the value of the correlation plus the significance level as star
s
#  Each significance level is associated to a symbol : p-values(0, 0.001, 0.01, 0.05, 0.1, 1) <=
> symbols("***", "**", "*", ".", " ")
chart.Correlation(data.r, histogram=TRUE, pch=19)
```




```
# Get some colors
col<- colorRampPalette(c("blue", "white", "red"))(20)
heatmap(x = corresult, col = col, symm = TRUE)
```



Model 1: This model will try and predict the 'Cooling Load' feature

```
x <- c("X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8")
y <- "Y2"

#setup the grid to do hyperparameter tuning for a RandomForest model
g <- h2o.grid("randomForest",
              hyper_params = list(
                ntrees = c(50, 100, 120),
                max_depth = c(40, 60),
                min_rows = c(1, 2)
              ),
              x = x, y = y, training_frame = train, nfolds = 10
)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
#evaluate the outcome of the grid
g_rmse = h2o.getGrid(g@grid_id, sort_by = "rmse")
as.data.frame( h2o.getGrid(g@grid_id, sort_by = "rmse")@summary_table )
```

```
##    max_depth min_rows ntrees
## 1         40      1.0    100
## 2         40      2.0    120
## 3         40      1.0     50
## 4         60      1.0    100
## 5         40      1.0    120
## 6         60      1.0     50
## 7         60      1.0    120
## 8         60      2.0    100
## 9         40      2.0     50
## 10        60      2.0    120
## 11        40      2.0    100
## 12        60      2.0     50
##                                                    model_ids
## 1   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_5
## 2  Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_11
## 3   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_1
## 4   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_6
## 5   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_9
## 6   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_2
## 7  Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_10
## 8   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_8
## 9   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_3
## 10 Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_12
## 11  Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_7
## 12  Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_15_model_4
##                  rmse
## 1  1.7901576438902003
## 2  1.8081613546403708
## 3  1.8240502227173039
## 4  1.8339236233295912
## 5  1.8468401268785628
## 6   1.850572057925703
## 7  1.8522646548758823
## 8   1.859740293861997
## 9   1.868130986939924
## 10  1.872484780226446
## 11 1.8955921072756776
## 12 1.9189233419966236
```

```
#get the best model
best_model <- h2o.getModel(g_rmse@model_ids[[1]])

#print RMSE of the best model
h2o.rmse(best_model)
```

```
## [1] 1.839036
```

```
#predict on test dataset
perf <- h2o.performance(best_model, test)
perf
```

```
## H2ORegressionMetrics: drf
##
## MSE:  3.82088
## RMSE:  1.954707
## MAE:  1.395066
## RMSLE:  0.05989934
## Mean Residual Deviance :  3.82088
```

```
rmse_results <- data.frame(method = "Cooling load model (Test Accuracy)", RMSE = h2o.rmse(perf))
```

Model 2: This model will try and predict the ‘Heating Load’ feature

```
x <- c("X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8")
y <- "Y1"

#setup the grid to do hyperparameter tuning for a RandomForest model
g <- h2o.grid("randomForest",
              hyper_params = list(
                ntrees = c(50, 100, 120),
                max_depth = c(40, 60),
                min_rows = c(1, 2)
              ),
              x = x, y = y, training_frame = train, nfolds = 10
)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
#evaluate the outcome of the grid
g_rmse = h2o.getGrid(g@grid_id, sort_by = "rmse")
as.data.frame( h2o.getGrid(g@grid_id, sort_by = "rmse")@summary_table )
```

```
##    max_depth min_rows ntrees
## 1         40      1.0    100
## 2         40      1.0     50
## 3         60      1.0     50
## 4         60      1.0    100
## 5         40      1.0    120
## 6         60      1.0    120
## 7         40      2.0    100
## 8         60      2.0     50
## 9         40      2.0    120
## 10        60      2.0    120
## 11        40      2.0     50
## 12        60      2.0    100
##                                                    model_ids
## 1   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_5
## 2   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_1
## 3   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_2
## 4   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_6
## 5   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_9
## 6  Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_10
## 7   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_7
## 8   Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_4
## 9  Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_11
## 10 Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_12
## 11  Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_3
## 12  Grid_DRF_RTMP_sid_916b_6_model_R_1591863068788_16_model_8
##                  rmse
## 1  0.9946137871654349
## 2  1.0455227120359774
## 3  1.0543048881884387
## 4   1.061096642325836
## 5  1.0870708804080342
## 6  1.1080823610087158
## 7  1.1157345057987083
## 8  1.1254434127029633
## 9   1.136770751188993
## 10 1.1551460760667356
## 11  1.193442485677092
## 12 1.1942699499423923
```

```
#get the best model
best_model <- h2o.getModel(g_rmse@model_ids[[1]])

#print RMSE of the best model
h2o.rmse(best_model)
```

```
## [1] 0.9778967
```

```
#predict on test dataset
perf <- h2o.performance(best_model, test)
perf
```

```
## H2ORegressionMetrics: drf
##
## MSE:  0.853328
## RMSE:  0.9237576
## MAE:  0.7357144
## RMSLE:  0.04407953
## Mean Residual Deviance :  0.853328
```

```
rmse_results <- rbind(rmse_results,
                          data.frame(method = "Heating load model (Test Accuracy)", RMSE = h2o.r
mse(perf))
                          )
```

Models Summary

```
rmse_results
```

```
##                                method      RMSE
## 1 Cooling load model (Test Accuracy) 1.9547070
## 2 Heating load model (Test Accuracy) 0.9237576
```