

# Prediction of Heating and Cooling Loads in Building Design: An Application of Random Forest Based Prediction

Tristen Bristow  
capitolmotion@gmail.com  
05/01/2017

## Abstract

This study concerns the clarification of structural and physical relationships in building design, specifically as an optimization study of materials and structure as they affect energy efficiency. Prior originating work is a simulation-based study, where this work concerns the interpretation of simulation output-data already generated in that study. Random-Forest machine learning is applied to the task of prediction, and any clarification of underlying relationships is attempted via this approach.

## Import libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
```

## Load Data

```
np.random.seed(1)
wb = pd.read_excel("ENB2012_data.xlsx")
```

## Data Description

The total number of number of samples is 768, each representing a unique building shapes. Independent-factors and their units are,

- Relative Compactness (%)
- Surface Area (meters<sup>2</sup>)
- Wall Area (meters<sup>2</sup>)
- Roof Area (meters<sup>2</sup>)
- Overall Height (meters)
- Orientation (1-4)
- Glazing Area (% surface covered)
- Glazing Area Distribution (1-5)

## Data Cleaning

All column names are re-titled according source documentation:

```
wb.columns = ['Compactness', 'Surface', 'Wall', 'Roof', 'Height', 'Orientation', 'Glazing', 'Glazing Dist.', 'Heating', 'Cooling']
```

## Summary Statistics

```
print(wb.describe())
```

	Compactness	Surface	Wall	Roof	Height \
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	0.764167	671.708333	318.500000	176.604167	5.250000
std	0.105777	88.086116	43.626481	45.165950	1.75114
min	0.620000	514.500000	245.000000	110.250000	3.500000
25%	0.682500	606.375000	294.000000	140.875000	3.500000
50%	0.750000	673.750000	318.500000	183.750000	5.250000
75%	0.830000	741.125000	343.000000	220.500000	7.000000
max	0.980000	808.500000	416.500000	220.500000	7.000000

	Orientation	Glazing	Glazing Dist.	Heating	Cooling
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.500000	0.234375	2.81250	22.307195	24.587760
std	1.118763	0.133221	1.55096	10.090204	9.513306
min	2.000000	0.000000	0.000000	6.010000	10.900000
25%	2.750000	0.100000	1.75000	12.992500	15.620000
50%	3.500000	0.250000	3.00000	18.950000	22.080000
75%	4.250000	0.400000	4.00000	31.667500	33.132500
max	5.000000	0.400000	5.00000	43.100000	48.030000

Compactness	False
Surface	False
Wall	False
Roof	False
Height	False
Orientation	False
Glazing	False
Glazing Dist.	False
Heating	False
Cooling	False

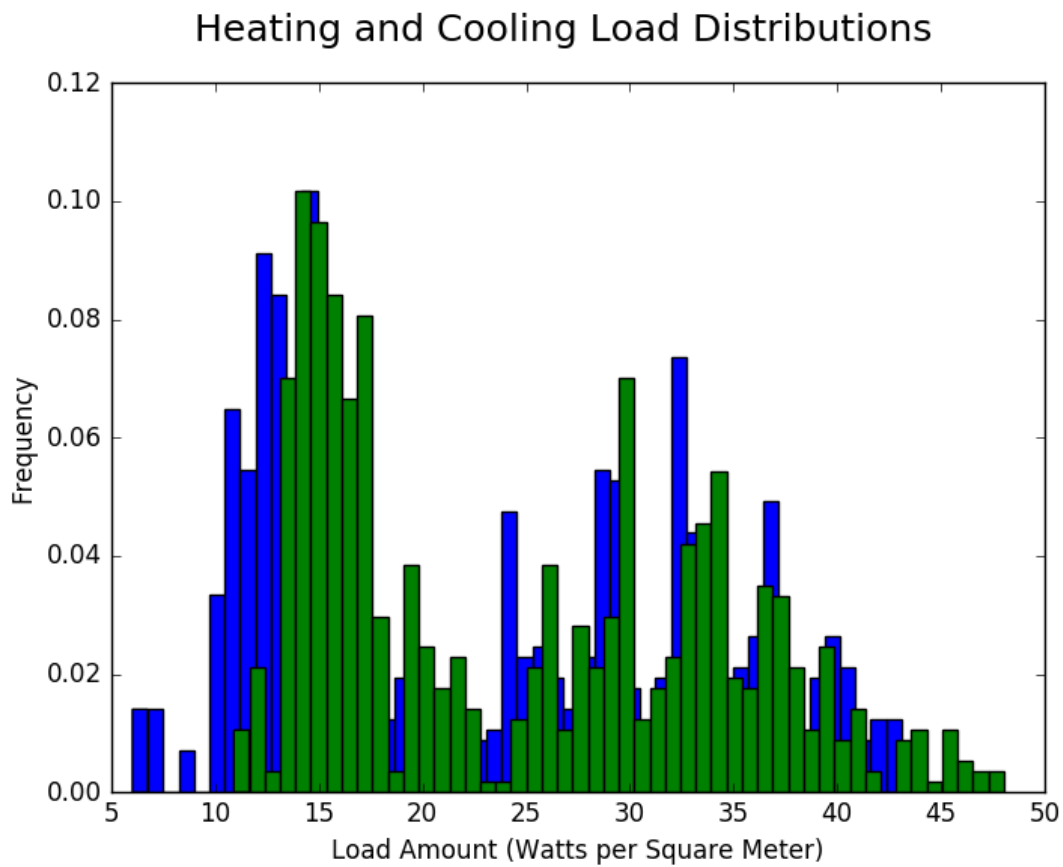
```
print(wb.isnull().any())
```

It can be seen by the above output that there are no missing values in the data.

## A Check of Output Variables for Normality

```
plt.suptitle('Heating and Cooling Load Distributions', fontsize=18)  
plt.hist(wb['Heating'], 50, normed = True)
```

```
plt.hist(wb['Cooling'], 50, normed = True)
plt.ylabel('Frequency')
plt.xlabel('Load Amount')
plt.show()
```



*Fig 1: Histogram of Heating and Cooling Load outputs over full set.*

All Factors Vs. Heating Load (Watts per Square-Meter)

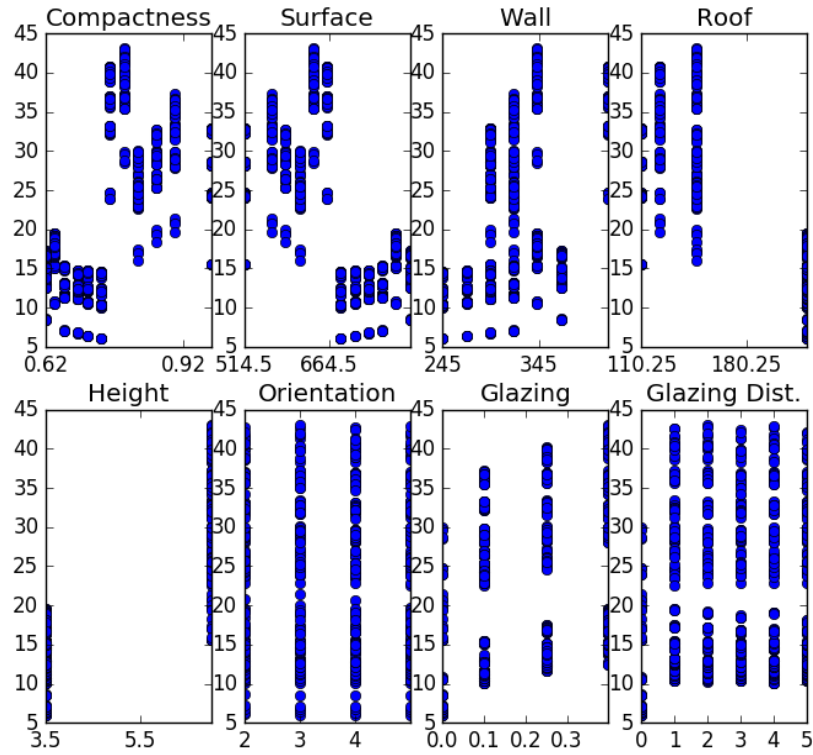


Fig 2: Factor Values/Levels vs Output (Heating Load)

All Factors Vs. Cooling Load (Watts per Square Meter)

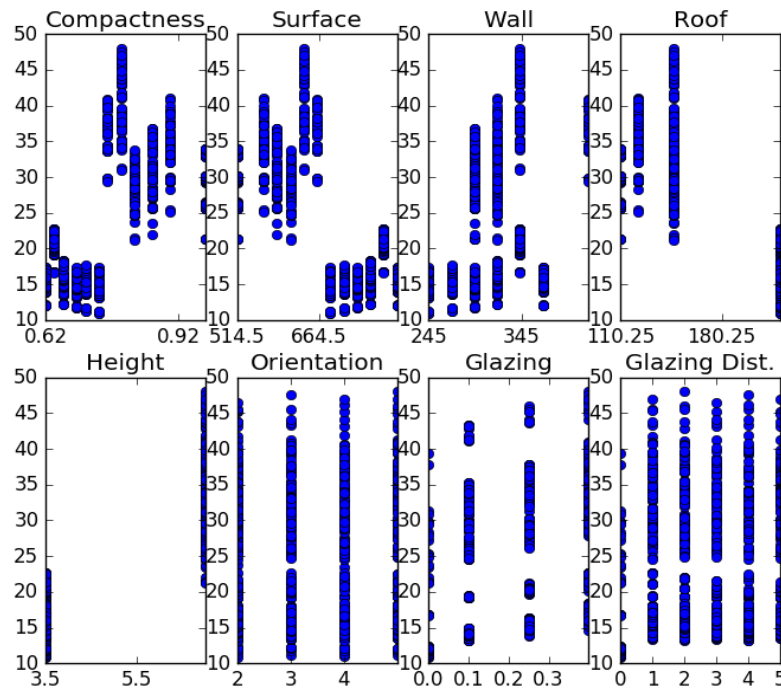


Fig 3: Factor Values/Levels vs Output (Cooling Load)

Output histograms 'Heating' and 'Cooling' distributions both show non-normality, (see fig 1), thus it is clear that a regression-based approach will not be possible.

```
plt.figure(figsize=(8, 7))
plt.suptitle('All Factors Vs. Heating Load', fontsize=18)
plt.subplot(2, 4, 1).set_title('Compactness')
plt.xticks(np.arange(min(wb['Compactness']), max(wb['Compactness']), .3))
plt.plot(wb['Compactness'], wb['Heating'], "o")
plt.subplot(2, 4, 2).set_title('Surface')
#50 - 80
plt.xticks(np.arange(min(wb['Surface']), max(wb['Surface']), 150))
plt.plot(wb['Surface'],wb['Heating'], "o")
plt.subplot(2, 4, 3).set_title('Wall')
#240-420
plt.xticks(np.arange(min(wb['Wall']), max(wb['Wall']), 100))
plt.plot(wb['Wall'],wb['Heating'], "o")
plt.subplot(2, 4, 4).set_title('Roof')
#100-240
plt.xticks(np.arange(min(wb['Roof']), max(wb['Roof']), 70))
plt.plot(wb['Roof'],wb['Heating'], "o")
plt.subplot(2, 4, 5).set_title('Height')
#3.5-7
```

```

plt.xticks(np.arange(min(wb['Height']), max(wb['Height']), 2))
plt.plot(wb['Height'],wb['Heating'], "o")
plt.subplot(2, 4, 6).set_title('Orientation')
#2-5
plt.xticks(np.arange(min(wb['Orientation']), max(wb['Orientation']), 1))
plt.plot(wb['Orientation'],wb['Heating'], "o")
plt.subplot(2, 4, 7).set_title('Glazing')
#0-.45
plt.xticks(np.arange(min(wb['Glazing']), max(wb['Glazing']), .1))
plt.plot(wb['Glazing'],wb['Heating'], "o")
plt.subplot(2, 4, 8).set_title('Glazing Dist.')
#0-5
plt.plot(wb['Glazing Dist.'],wb['Heating'], "o")

plt.show()

```

```

plt.figure(figsize=(8, 7))
plt.suptitle('All Factors Vs. Cooling Load', fontsize=18)
plt.subplot(2, 4, 1).set_title('Compactness')
plt.xticks(np.arange(min(wb['Compactness']), max(wb['Compactness']), .3))
plt.plot(wb['Compactness'], wb['Cooling'], "o")
plt.subplot(2, 4, 2).set_title('Surface')
#50 - 80
plt.xticks(np.arange(min(wb['Surface']), max(wb['Surface']), 150))
plt.plot(wb['Surface'],wb['Cooling'], "o")
plt.subplot(2, 4, 3).set_title('Wall')
#240-420
plt.xticks(np.arange(min(wb['Wall']), max(wb['Wall']), 100))
plt.plot(wb['Wall'],wb['Cooling'], "o")
plt.subplot(2, 4, 4).set_title('Roof')
#100-240
plt.xticks(np.arange(min(wb['Roof']), max(wb['Roof']), 70))
plt.plot(wb['Roof'],wb['Cooling'], "o")
plt.subplot(2, 4, 5).set_title('Height')
#3.5-7
plt.xticks(np.arange(min(wb['Height']), max(wb['Height']), 2))
plt.plot(wb['Height'],wb['Cooling'], "o")
plt.subplot(2, 4, 6).set_title('Orientation')
#2-5
plt.xticks(np.arange(min(wb['Orientation']), max(wb['Orientation']), 1))
plt.plot(wb['Orientation'],wb['Cooling'], "o")
plt.subplot(2, 4, 7).set_title('Glazing')
#0-.45
plt.xticks(np.arange(min(wb['Glazing']), max(wb['Glazing']), .1))
plt.plot(wb['Glazing'],wb['Cooling'], "o")
plt.subplot(2, 4, 8).set_title('Glazing Dist.')
#0-5
plt.plot(wb['Glazing Dist.'],wb['Cooling'], "o")

```

```
plt.show()
```

Scatter plots of independently varying factors verses measured Heating and Cooling Loads indicates a varying correlation-type. The relationships appear in some instances to be linear and others non-linear. In certain relationships, compactness, surface, and wall area verses roof and heating and cooling loads appear at times to show a discontinuous separation that can be indicative of underlying dynamics. It suggests the existence of underlying independent-factor threshold-levels that define multiple dynamic modes in the system. The modes separated, examined unto themselves, appear generally to show linear relationships with heating and cooling load outputs. These discontinuities appear to manifest at specific threshold-levels for each factor where relative linearity is otherwise observed. It is clear that a Random Forests is a well-advised approach for analysis, as the system can be described by a machine-learning method that can maximally utilize information from these discontinuities and encompass the relative simplicity of dynamic expression between them (owing to linearity and their limited number of discrete states, in addition to a limited number of potentially-relevant factors to be considered. See figures 2 and 3)

## Random Forest Modeling

As the relevant aspects of cross-validation are inherent in the Random Forest algorithm, train and test splits are unnecessary to produce a valid out-of-sample performance estimate as is any externally-applied cross-validation. Though this is the case, it appears to be a common practice to demonstrate random-forest performance via train-test data splitting, and thus is applied here to further elaborate out-of-sample performance.

Splitting data into input verses target values:

```
wb_train = wb[ : int(.75 * len(wb))]  
wb_test = wb[int(.75 * len(wb)) : ]
```

```
d_ind = wb_train.iloc[:,8]  
d_target = wb_train.iloc[:,8:10]
```

```
t_ind = wb_test.iloc[:,8]  
t_target = wb_test.iloc[:,8:10]
```

```
mdl = RandomForestRegressor(n_estimators = 8,  
                           max_features = 'auto',  
                           max_depth = None,  
                           min_samples_split = 2,
```

```
min_samples_leaf = 1,  
min_weight_fraction_leaf = 0,  
max_leaf_nodes = None,  
n_jobs = -1)
```

```
mdl.fit(d_ind, d_target)  
result = mdl.predict(t_ind)
```

```
res = mdl.score(t_ind, t_target)  
print(res*100)
```

Output:

97.7352900775

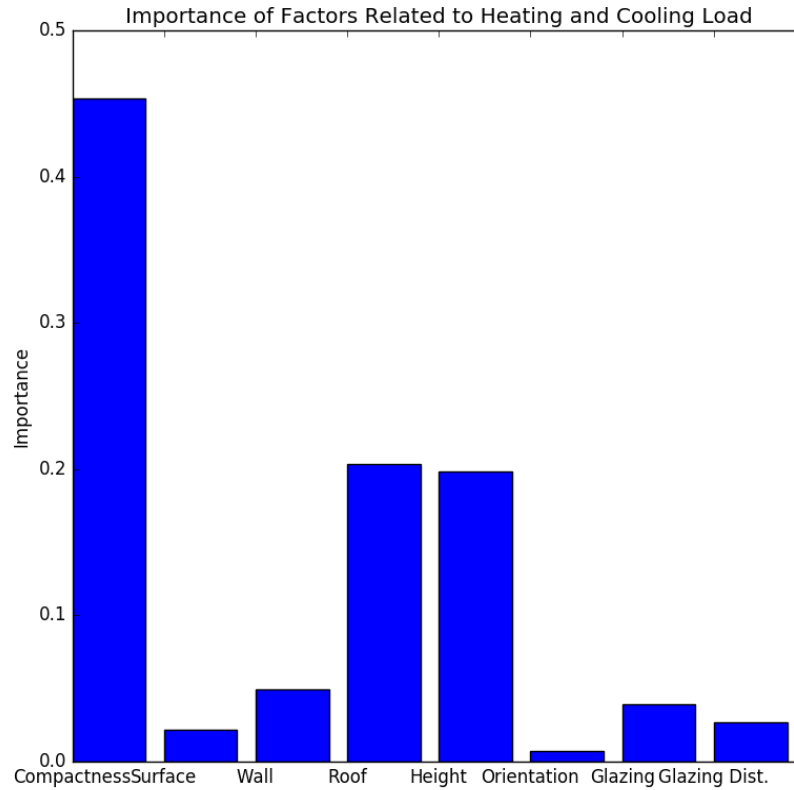
Performance on the test data is measured at 98% accuracy. With a well-fitted model, it is possible to establish the relative importance of the included factors by their occurrence across many fittings (in this case, 10 are attempted).

```
importance = mdl.feature_importances_  
name = list(d_ind)
```

```
# print("Type      Importance")  
# for i in range(8):  
#     print (name[i], "      ", importance[i])
```

```
y_pos = np.arange(len(name))  
plt.figure(figsize=(9,10))  
plt.bar(y_pos, importance)  
plt.xticks(y_pos, name)  
plt.ylabel('Importance')  
plt.title('Importance of Factors Related to Heating and Cooling Load')  
plt.show()
```





*Fig 4: Importance graph of model factors in Random Forest prediction.*

## Discussion

Out of sample prediction accuracy is measured at 98%. Results indicate the most important factors related to Heating and Cooling Loads are Compactness and Height. These values differ significantly from the others. Surface-Area and Roof-Area can also appear in many cases as important factors, and over many runs they show not to be as stable as the other two (they appear in many model fits, but not all). This relative-importance of model factors should be interpreted as efficiently-deduced for the purposes of predictive modeling (see fig 4). It is highly recommended that established design conclusions follow from the direct utilization of this model for prediction.

## Conclusion

The results of this study indicate the existence of significant design features that can inform a practical perspective. The Random Forest model is highly effective for the ends of Heating and Cooling Load-prediction, and can be used to predict these quantities provided all relevant feature-values are included.