INTELLIGENT ESTIMATION OF HEATING AND COOLING LOAD REQUIREMENTS OF BUILDINGS

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

B.Tech

in

Production Engineering

By

Tushar Gautam (114113089)

PRODUCTION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI-620015

APRIL 2017

BONAFIDE CERTIFICATE

This is to certify that the project titled **INTELLIGENT ESTIMATION OF HEATING AND COOLING LOAD REQUIREMENTS OF BUILDINGS** is a bonafide record of the work done by

Tushar Gautam (114113089)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Production Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the year 2016-2017.

Dr.K.Panneerselvam	**Dr.E.S.Gopi**	**Dr.M.Duraiselvam**
Guide	Co-Guide	Head of the Department

Project Viva-voce held on _____

Internal Examiner **External Examiner**

ABSTRACT

This project studies the effect of various input variables (relative compactness, surface area, wall area, roof area, overall height, orientation, glazing area, glazing area distribution) on two output variables, namely heating load (HL) and cooling load (CL) of buildings. Reports suggest that building energy consumption has steadily increased over the past decades worldwide [1], [2], and heating, ventilation and air conditioning (HVAC), which have a catalytic role in regulating the indoor climate, account for most of the energy use in the buildings. Therefore, one way to alleviate the ever increasing demand for additional energy supply is to have more energy-efficient building designs with improved energy conservation properties. When it comes to efficient building design, the computation of the heating load (HL) and the cooling load (CL) is required to determine the specifications of the heating and cooling equipment needed to maintain comfortable indoor air conditions. Various Machine Learning approach (Regression, ANN etc) is to be studied, implemented, compared and visualised [3] on the datasets obtained from Machine Learning Repository, Centre for Machine Learning and Intelligent Systems, UCI. Finally, a model with the best accuracy is proposed.

Keywords: Building energy evaluation; Heating load; Cooling load; Regression; ANN

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to Dr.K.Pannerselvam, for reviewing my work and constant support throughout the project. I place on record, my sincere thank you to the Dr.M.Duraiselvam and Dr.S.Vinodh for their continuos encouragement to pursue the project in the first place.

I am also grateful to Dr. E.S.Gopi, Assistant Professor, in the Department of Electronics and Communication Engineering. I am extremly thankful and indebted to him for sharing expertise, sincere and valuable guidance and encouragement extended to me. Furthermore, taking a course on Pattern Recognition (EC459) by Dr. E.S. Gopi helped significantly.

I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture.

TABLE OF CONTENTS

Title	Page No.
ABSTRACT	**ii**
ACKNOWLEDGEMENTS	**iii**
TABLE OF CONTENTS	**iv**
LIST OF TABLES	**vi**
LIST OF FIGURES	**vii**
1 Introduction	**1**
1.1 Simulation tools approach	2
1.2 Machine learning approach	2
2 Review of Literature	**3**
3 Data exploration	**5**
3.1 Dataset description	5
3.1.1 Mapping	5
3.2 Probability density	5
3.2.1 Histograms	6
3.2.2 Conclusion	7
3.3 Correlation	7
3.3.1 Scatter plot	7
3.3.2 Spearman rank correlation coefficient	9
3.3.3 Correlation Matrix for features(X)	10
3.3.4 Conclusion	12
4 Machine learning approach	**13**
4.1 Linear models	13
4.1.1 Ridge Regression	13
4.2 Non-Linear models	14
4.2.1 Artificial Neural Networks	14
5 Code Snippets	**17**
6 Results and Discussions	**25**
6.1 Model accuracy	25

6.1.1 Linear vs Non-linear model 25

7 Summary and Conclusions 30

7.1 Summary 30

7.2 Conclusions 30

7.3 Scope for Future Work 30

REFERENCES 31

LIST OF TABLES

3.1 Features 5
3.2 Responses 5
3.3 Association strength estimated using the Spearman rank correlation coefficient of the eight input variables (X1. . . X8) with heating load (y1). 10
3.4 Association strength estimated using the Spearman rank correlation coefficient of the eight input variables (X1. . . X8) with cooling load (y2). 10

LIST OF FIGURES

3.1 Cooling load (y1) 6
3.2 Heating load (y2) 6
3.3 Rel. compactness (X1) 6
3.4 Surface area (X2) 6
3.5 Wall area (X3) 6
3.6 Roof area (X4) 6
3.7 Overall height (X5) 7
3.8 Orientation (X6) 7
3.9 Glazing area (X7) 7
3.10 Glazing area distribution (X8) 7
3.11 Scatter plot grid between features (X) 8
3.12 Scatter plot grid between features (X) vs responses (Y) 8
3.13 Spearman rank correlation coeff. for various features 9
3.14 Correlation matrix shown as Color Map 11

4.1 Intutively, non-linear model fits better. 14
4.2 Neurons in human brain 15
4.3 Artificial neuron 15
4.4 Artificial neuron with various layers 15

6.1 Models accuracy : Linear (Ridge) vs Non-linear (ANN) 25
6.2 Target CL, Output CL and corresponding error (target-output) plot . . . 26
6.3 Target HL, Output HL and corresponding error (target-output) plot . . . 27
6.4 Target CL, Output CL and corresponding error (target-output) plot . . . 28
6.5 Target HL, Output HL and corresponding error (target-output) plot . . . 29

CHAPTER 1

INTRODUCTION

Globally the building sector accounts for more electricity use than any other sector, 42 per cent. No wonder considering that we spend more than 90 per cent of our time in buildings. With increasing urbanization, higher in developing countries, the number and size of buildings in urban areas will increase, resulting in an increased demand for electricity and other forms of energy commonly used in buildings. Africa's rate of urbanization of 3.5 per cent per year is the highest in the world, resulting in more urban areas with bigger populations, as well as the expansion of existing urban areas. There are currently 40 cities in Africa with populations of more than a million and it is expected that by 2015 seventy cities will have populations of one million or more.

In many developing countries there is normally very little margin between existing power supply and electricity demand. With increasing electricity demand, new generation needs to be brought in. Although renewable sources of electricity such as hydro, geothermal or wind provide electricity at a much lower cost, their capital outlay is large, they are complex and take much longer to implement. Diesel based generation is usually brought in the short term to meet this demand, which results in increased cost of electricity.

Investments in energy efficiency in a building can be compared with the cost of capital investments necessary on the supply side of the energy system to produce a similar amount of peak capacity or annual energy production. Usually, the capital costs of efficiency are lower than comparable investments in increased supply and there are no additional operating costs of efficiency compared to substantial operating costs for supply-side options. In addition, energy efficiency investments generally have much shorter lead times than energy supply investments, a particularly important consideration in countries where the demand for energy services is growing rapidly.

One consistent quality in the building sector is that it is subject to a high degree of regulation. Building codes often influence material use and appliance standards that have a significant effect on energy efficiency. Regulatory regimes, to the extent that they exist, may therefore provide a pathway to improve efficiency for both building construction and a variety of building appliances.

Reports [1], [2] suggest that heating, ventilation and air conditioning (HVAC), which have a catalytic role in regulating the indoor climate, account for most of the energy use in the buildings. Therefore, one way to alleviate the ever increasing demand for additional energy supply is to have more energy-efficient building designs with improved energy conservation properties. When it comes to efficient building design, the computation of the heating load (HL) and the cooling load (CL) is required to determine the specifications of the heating and cooling equipment needed to maintain comfortable indoor air conditions.

1.1 SIMULATION TOOLS APPROACH

Building energy simulation tools are currently widely used to analyze or forecast building energy consumption in order to facilitate the design and operation of energy efficient buildings. Simulation tools are used extensively across diverse disciplines because they enable experimentation with parameters that would otherwise be infeasible, or at least very difficult to control in practice [4]

1.2 MACHINE LEARNING APPROACH

Using advanced dedicated building energy simulation software may provide reliable solutions to estimate the impact of building design alternatives; however this process can be very time-consuming and requires user expertise in a particular program. Moreover, the accuracy of the estimated results may vary across different building simulation software. Hence, in practice many researchers rely on machine learning tools to study the effect of various building parameters (e.g. compactness) on some variables of interest (e.g. energy) because this is easier and faster if a database of the required ranges of variables is available. Using statistical and machine learning concepts has the distinct advantage that distilled expertise from other disciplines is brought in the EPB domain, and by using these techniques it is extremely fast to obtain answers by varying some building design parameters once a model has been adequately trained. Moreover, statistical analysis can enhance our understanding offering quantitative expressions of the factors that affect the quantity (or quantities) of interest that the building designer or architect may wish to focus on.

In this report, various suitable machine learning models (linear and non-linear) have been explored to predict HL and CL.

CHAPTER 2

REVIEW OF LITERATURE

A. Tsanas, A. Xifara (2012) [5] developed a statistical machine learning framework using a range of diverse input variables to study heating load and cooling load requirements of a building. It demonstrated that it is possible to accurately estimate HL with only 0.5 points deviation and CL with 1.5 points deviation from the ground truth (the simulated results). These findings are particularly compelling given the accurate prediction, and also because we can easily infer the output variables in a matter of few seconds without requiring the painstaking design of a new building in a simulation tool.

A. Hani and T. Koiv (2012) [6] analyzes the thermal and electrical energy consumptions for 40 Residential, 7 Educational and 44 Public buildings located in warm summer continental climate. Information about reconstructions, heating source, internal air temperature, air exchange rate and domestic hot water production is tabulated.

Dong, B., Cao, C., & Lee, S. E. (2005) [7] This paper presents support vector machines (SVM), a new neural network algorithm, to forecast building energy consumption in the tropical region. The objective of this paper is to examine the feasibility and applicability of SVM in building load forecasting area. Four commercial buildings in Singapore are selected randomly as case studies. Weather data including monthly mean outdoor dry-bulb temperature (T0), relative humidity (RH) and global solar radiation (GSR) are taken as three input features. Mean monthly landlord utility bills are collected for developing and testing models. Finally, all prediction results are found to have coefficients of variance (CV) less than 3% and percentage error (%error) within 4%.

S.S.K. Kwok, R.K.K. Yuen, E.W.M. Lee (2011) [8] This paper discusses the use of the multi-layer perceptron (MLP) model, one of the artificial neural network (ANN) models widely adopted in engineering applications, to estimate the cooling load of a building. The training samples used include weather data obtained from the Hong Kong Observatory and building-related data acquired from an existing prestigious commercial building in Hong Kong that houses a mega complex and operates 24 h a day. The paper also discusses the practical difficulties encountered in acquiring building-related data. In contrast to other studies that use ANN models to predict building cooling load, this paper includes the building occupancy rate as one of the input parameters used to determine building cooling load. The results demonstrate that the building occupancy

rate plays a critical role in building cooling load prediction and significantly improves predictive accuracy.

Yu, Zhun and Haghighat, Fariborz and Fung, Benjamin C.M. and Yoshino, Hiroshi (2010) [9] This paper reports the development of a building energy demand predictive model based on the decision tree method. The developed model estimates the building energy performance indexes in a rapid and easy way. This method is appropriate to classify and predict categorical variables: its competitive advantage over other widely used modeling techniques, such as regression method and ANN method, lies in the ability to generate accurate predictive models with interpretable flowchart-like tree structures that enable users to quickly extract useful information.

CHAPTER 3

DATA EXPLORATION

3.1 DATASET DESCRIPTION

The dataset contains eight attributes (or features, denoted by X1...X8) and two responses (or outcomes, denoted by y1 and y2).

Table 3.1: Features

Relative compactness (X1)
Surface area (X2)
Wall area (X3)
Roof area (X4)
Overall height (X5)
Orientation (X6)
Glazing area (X7)
Glazing area distribution (X8)

Table 3.2: Responses

Cooling load (y1)
Heating load (y2)

3.1.1 Mapping

The aim is to use the eight features to predict each of the two responses.

$$f: \begin{array}{l} X \to Y \\ X \mapsto f(X) \end{array} \tag{3.1}$$

3.2 PROBABILITY DENSITY

The first step in most data analysis applications is the exploration of the statistical properties of the variables. This is typically achieved by plotting the probability densities, which succinctly summarize each variable for visualization. One way to obtain an empirical non-parametric density estimate is by using histograms. Although histograms are considered crude for most advanced statistical applications, they have the great advantage of making no prior assumptions regarding the distribution of the examined variable and are very simple to compute. *Often, this preliminary step can reveal whether the variable follows a Gaussian (normal) distribution, which is characterized by a unimodal peak in the middle of the variable's possible range of values, is completely symmetric*, and is particularly useful because a large number of mathematical functions are applicable [10].

3.2.1 Histograms

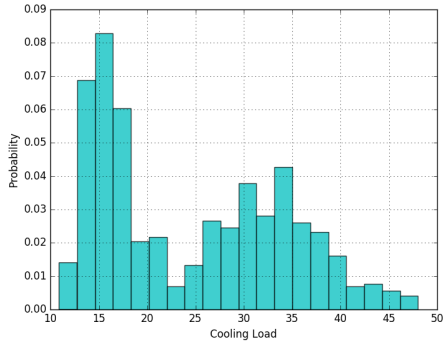

Figure 3.1: Cooling load (y1)

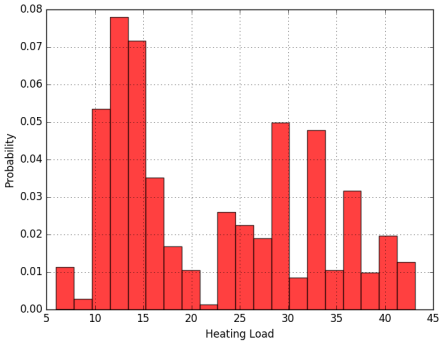

Figure 3.2: Heating load (y2)

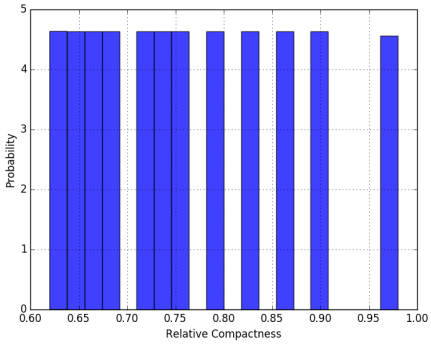

Figure 3.3: Rel. compactness (X1)

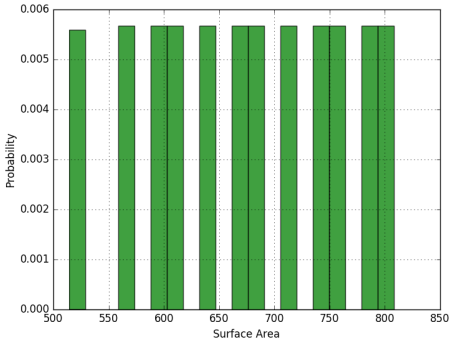

Figure 3.4: Surface area (X2)

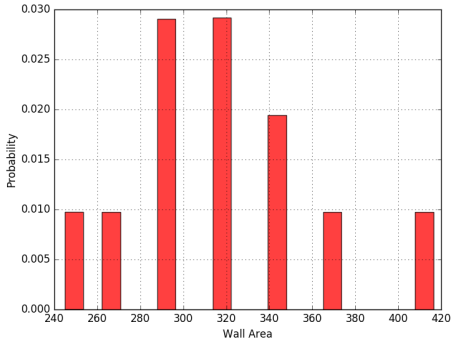

Figure 3.5: Wall area (X3)

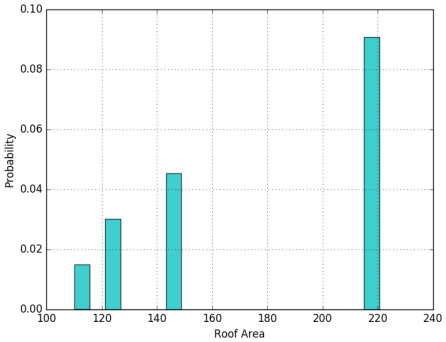

Figure 3.6: Roof area (X4)

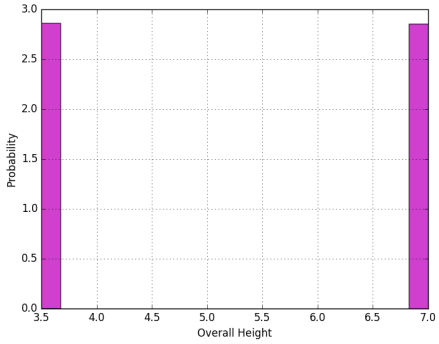

Figure 3.7: Overall height (X5)

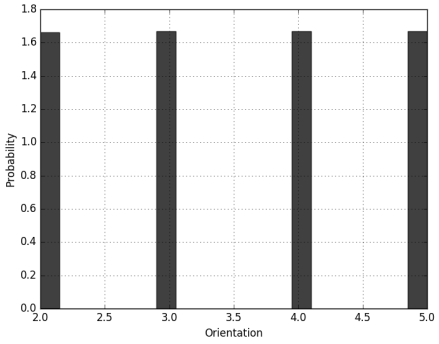

Figure 3.8: Orientation (X6)

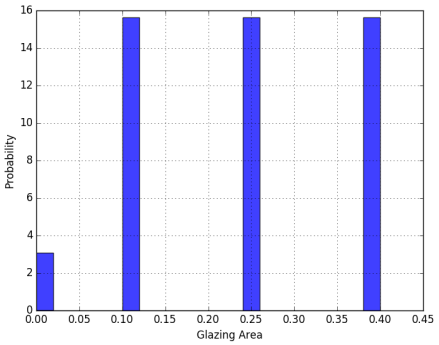

Figure 3.9: Glazing area (X7)

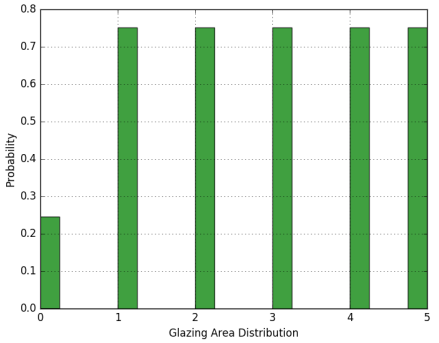

Figure 3.10: Glazing area distribution (X8)

3.2.2 Conclusion

We observe that there is no unimodal peak in the middle or the symmetry. Hence, the data is non-Gaussian.

3.3 CORRELATION

3.3.1 Scatter plot

Scatter plots are similar to line graphs in that they use horizontal and vertical axes to plot data points. However, they have a very specific purpose. Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation. For simplicity, scatter plots often use normalized data (i.e. all the variables are normalized to lie between 0 and 1) to facilitate comparison between measures that possibly span orders of magnitude different ranges of values. Correlation plot between different features and between features and responses have been shown below:

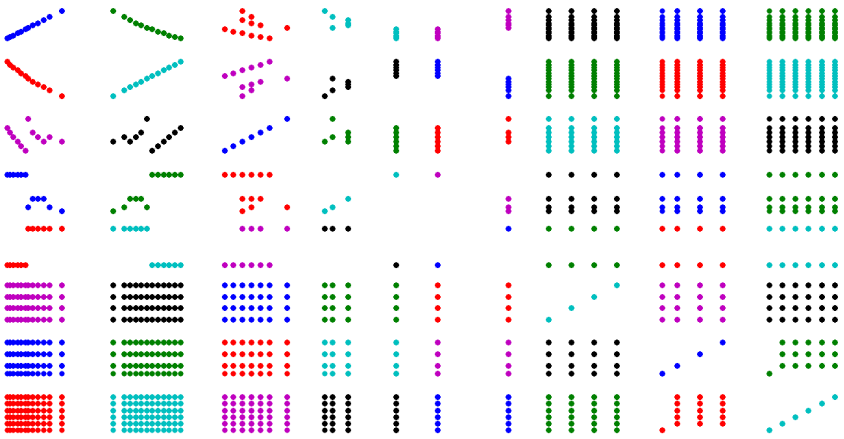

Figure 3.11: Scatter plot grid between features (X)

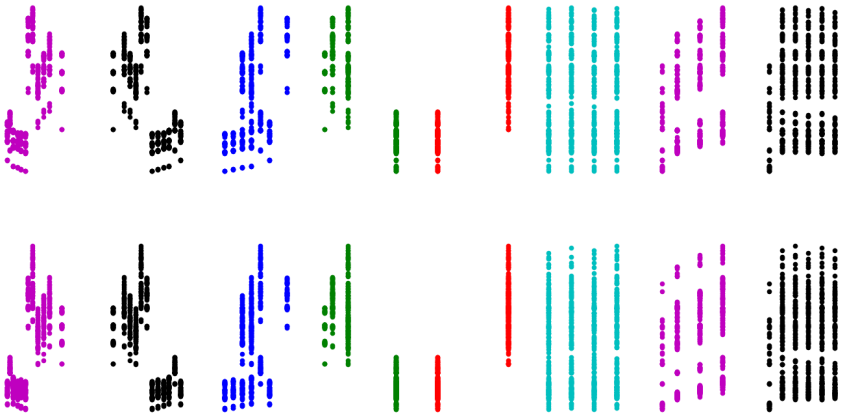

Figure 3.12: Scatter plot grid between features (X) vs responses (Y)

3.3.2 Spearman rank correlation coefficient

As concluded from the histogram plot above, the data is non-Gaussian, so we use the Spearman rank correlation coefficient to obtain a statistical metric regarding the association strength of each input variable with each of the two outputs. The Spearman rank correlation coefficient can characterize general monotonic relationships and lies in the range [-1 1], where negative sign indicates inversely proportional and positive sign indicates proportional rela- tionship, whilst the magnitude denotes how strong this relationship is.

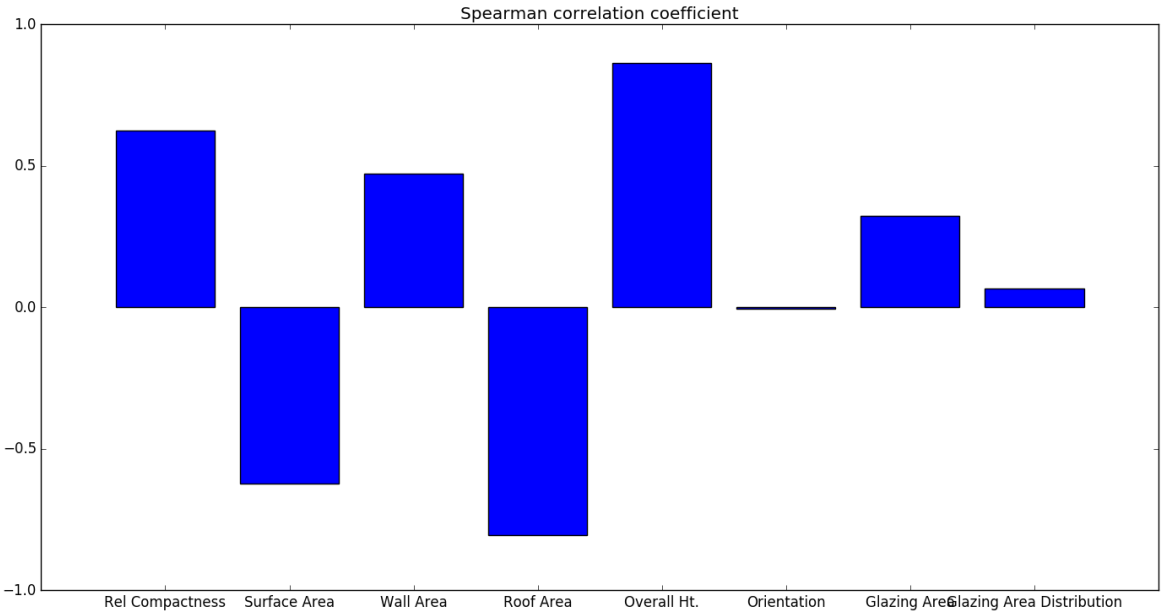

Figure 3.13: Spearman rank correlation coeff. for various features

Table 3.3: Association strength estimated using the Spearman rank correlation coefficient of the eight input variables (X1. . . X8) with heating load (y1).

Features	Spearman rank correlation coefficient
X1	0.623
X2	-0.623
X3	0.471
X4	-0.806
X5	0.861
X6	-0.004
X7	0.323
X8	0.068

Table 3.4: Association strength estimated using the Spearman rank correlation coefficient of the eight input variables (X1. . . X8) with cooling load (y2).

Features	Spearman rank correlation coefficient
X1	0.651
X2	-0.651
X3	0.416
X4	-0.803
X5	0.865
X6	0.018
X7	0.289
X8	0.046

3.3.3 Correlation Matrix for features(X)

$$\mathrm{Corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_x \sigma_y} = \frac{\mathrm{E}[(X-\mu_x)(Y-\mu_y)]}{\sigma_x \sigma_y} \tag{3.2}$$

$$\begin{bmatrix} 1.0 & -1.0 & -0.254 & -0.870 & 0.869 & 0.002 & 0.003 & 0.003 \\ -1.0 & 1.0 & 0.254 & 0.870 & -0.869 & -0.002 & -0.004 & -0.003 \\ -0.254 & 0.254 & 1.0 & -0.195 & 0.221 & -0.001 & -0.001 & -0.001 \\ -0.870 & 0.870 & -0.195 & 1.0 & -0.937 & -0.003 & -0.003 & -0.003 \\ 0.869 & -0.869 & 0.221 & -0.937 & 1.0 & 0.002 & 0.002 & 0.002 \\ 0.002 & -0.002 & -0.001 & -0.003 & 0.002 & 1.0 & -0.003 & -0.003 \\ 0.003 & -0.004 & -0.002 & -0.003 & 0.002 & -0.003 & 1.0 & 0.184 \\ 0.003 & -0.003 & -0.001 & -0.003 & 0.002 & -0.003 & 0.184 & 1.0 \end{bmatrix} \tag{3.3}$$

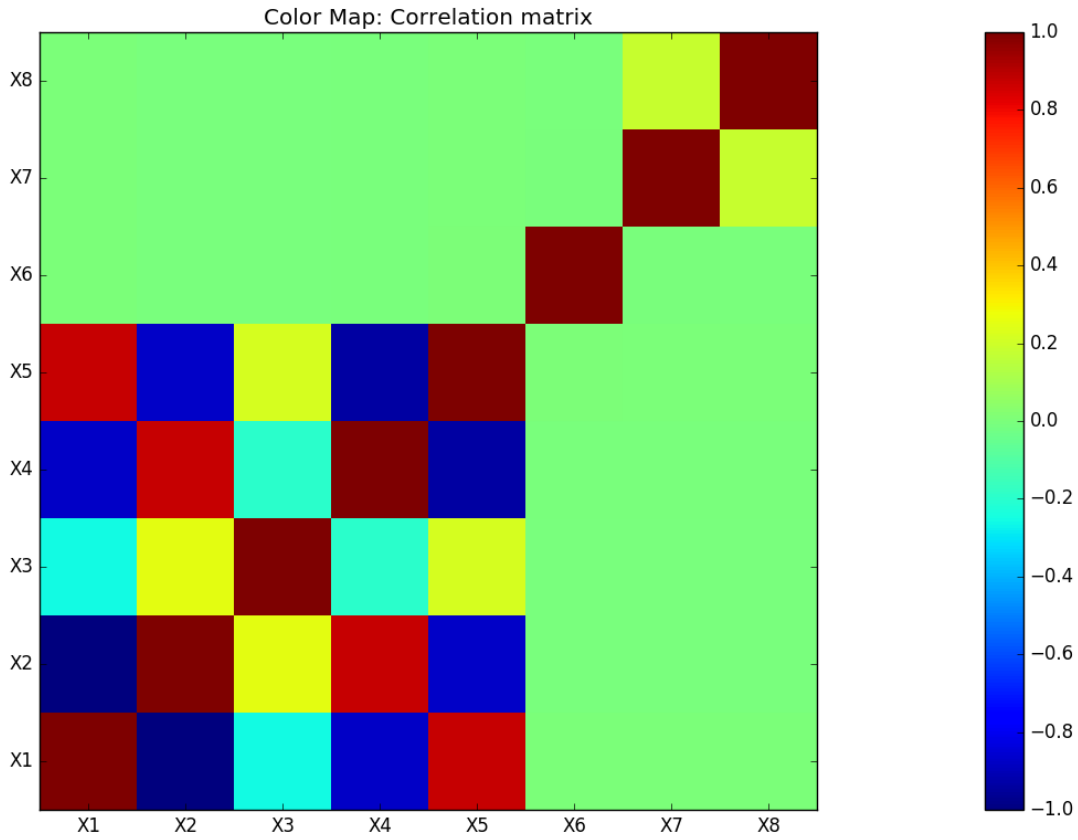

Figure 3.14: Correlation matrix shown as Color Map

3.3.4 Conclusion

Figure 3.3 presents the empirical probability distributions of all the input and output variables. These distributions demonstrate that none of the variables follows the normal distribution. Figure 3.12 displays the scatter plots for each of the (normalized) input variables with each of the two output variables. These scatter plots and Color Map show that any functional relationship of the input variables and the output variables is not trivial. This suggests that we can reasonably expect that classical learners such as linear regression (linear models) may fail to find an accurate mapping of the input variables to the output variables. Therefore, these plots intuitively justify the need to experiment with complicated learners such as Artificial Neural Networks (non linear models).

CHAPTER 4

MACHINE LEARNING APPROACH

4.1 LINEAR MODELS

In statistics, linear regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Linear Regression fits a linear model with coefficients $w = (w_1, ..., w_p)$ to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation. Mathematically it solves a problem of the form:

$$\min_{w} ||Xw - y||_2{}^2$$

However, coefficient estimates for Ordinary Least Squares rely on the independence of the model terms. When terms are correlated and the columns of the design matrix X have an approximate linear dependence, the design matrix becomes close to singular and as a result, the least-squares estimate becomes highly sensitive to random errors in the observed response, producing a large variance.

4.1.1 Ridge Regression

Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of coefficients. The ridge coefficients minimize a penalized residual sum of squares,

$$\min_{w} ||Xw - y||_2{}^2 + \alpha||w||_2{}^2$$

Here, $\alpha \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity.

4.2 NON-LINEAR MODELS

When the data has a non-linear relationship between dependent and independent variables, then non-linear models tend to perform better than linear models. Artificial Neural Network (ANN) models employ data-driven, self-adaptive methods, and can be used to perform nonlinear modeling without the need for prior knowledge about the relationship between input and output variables. They have been proven to be universal function approximators, and are capable of approximating highly nonlinear system behaviors by constructing behaviors on the basis of historical system data. The most commonly used supervised neural network is MLP, which we shall investigate a little more in further sections.

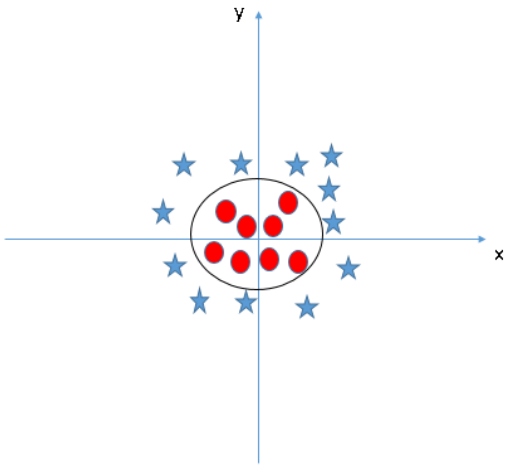

Figure 4.1: Intutively, non-linear model fits better.

4.2.1 Artificial Neural Networks

Artificial Neural Networks are a class of Machine learning algorithms inspired from the neurons of the human body (fig 4.2). This is biologically inspired machine learning algorithm. A neuron is a single unit that has several inputs with associated weights and an activation function. The weights are used to represent the strength of each input. Depending on the amount of activation, the neuron produces its own activity and sends this along its outputs.

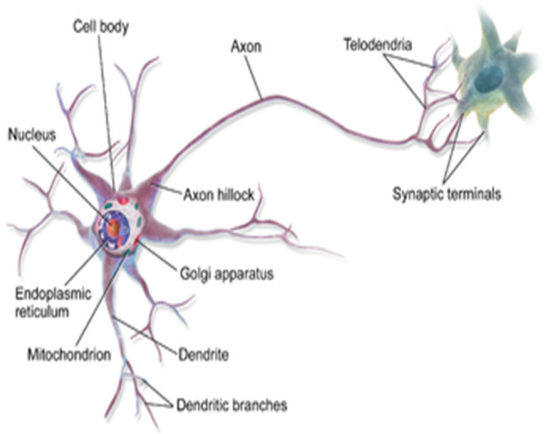

Figure 4.2: Neurons in human brain

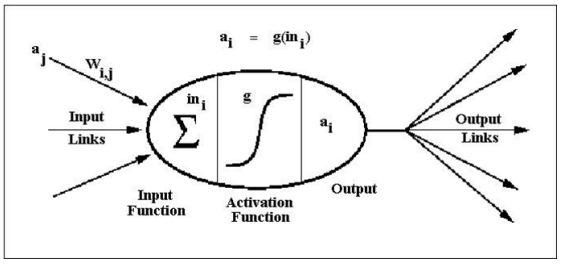

Figure 4.3: Artificial neuron

The artificial equivalent (fig 4.3) of a neuron is a node that receives a set of weighted inputs, processes their sum with its activation function ϕ and passes the result of the activation function to nodes further down the graph. We can then form a network by chaining these nodes together. Usually this is done in layers that is one layer's outputs are connected to the next layer's inputs (fig 4.4).

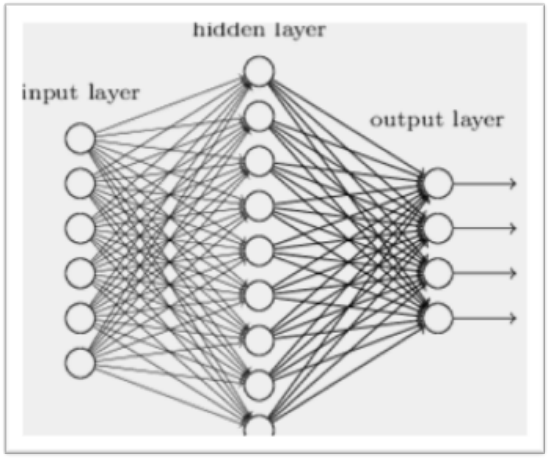

Figure 4.4: Artificial neuron with various layers

Training

The ANN is trained using Backpropagation algorithm. Backpropagation (BP) is a traditional training algorithm used for the MLP model. It feeds back the prediction errors from the output layer to the input layer and the weights of the links between the neurons are adjusted according to the BP algorithm. Upon completion of the weight adjustments, a new prediction is carried out to evaluate a new prediction error for the next epoch of weight adjustments. These procedures are repeated numerous times until a satisfactory prediction result is achieved. This is a method for updating the weights by propagating the error backwards, starting from the output layer, until each neuron has an associated error value which roughly represents its contribution to the original output.
Here the Loss function is SSE (sum-squared error).

$$f(x) = \sigma(Wx + b) \tag{4.1}$$

I have used the sigmoid activation function which is

$$\sigma(z) = e^z/(e^z + 1) \tag{4.2}$$

CHAPTER 5

CODE SNIPPETS

Below is the implementation in Python programming language. Suitable cross validation (CV), a standard statistical resampling technique has been used. Specifically, the dataset is split into a training subset with which the learner is trained, and a testing subset which is used to assess the learner's generalization performance. Typically some percentage of the data is left out for testing the learner, and this is known as K-fold CV, where K is usually 5 or 10. In this study, I've used 5-fold CV. The model parameters are derived using the training subset, and errors are computed using the testing subset (out-of-sample error or testing error). For statistical confidence, the training and testing process is repeated 100 times with the dataset randomly permuted in each run prior to splitting in training and testing subsets.

```
# -*- coding: utf-8 -*-
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import numpy as np
import pandas as pd

from itertools import cycle
from scipy.stats import spearmanr
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score



class Data(object):
    """
        Class for pre-processing data.
    """
    def __init__(self, filename="data.csv", cold=0):
        self.label = ["Rel Compactness", "Surface Area",
                      "Wall Area", "Roof Area",
                      "Overall Ht.", "Orientation", "Glazing Area",
                      "Glazing Area Distribution", "Heating Load",
                      "Cooling Load"]
```

```
        self.data = pd.read_csv(filename)
        self.cold = cold
        self.X = self.data.ix[:,0:8].as_matrix()
        self.y = self.data.ix[:,8:10].as_matrix()
        scaler = StandardScaler()
        self.X_train, self.X_test, self.y_train, self.y_test = train_test_s
            self.X,self.y,test_size=0.4,random_state=0)
        scaler.fit(self.X_train)
        scaler.fit(self.X)
        self.X_train = scaler.transform(self.X_train)
        self.X_test = scaler.transform(self.X_test)
        self.X = scaler.transform(self.X)

    def get_sp_rank(self):
        """
            Return Spearman rank-order correlation coefficient and p-value
            to test for non-correlation.
        """

        return [(spearmanr(self.data.ix[:, i:i+1], self.data.ix[:, 8:9]),
                 spearmanr(self.data.ix[:, i:i+1], self.data.ix[:, 9:10]))
                 for i in range(len(self.label)-2)]



def cross_validator(func):
    def inner_wrap(self):
        est, est_cv = func()
        X = self.X
        y = self.y
        alphas = np.logspace(-4, -0.5, 30)
        scores = []
        scores_std = []
        n_folds = 5

        for alpha in alphas:
            est.alpha = alpha
            this_scores = cross_val_score(est, X, y, cv=n_folds, n_jobs=1)
            scores.append(np.mean(this_scores))
            scores_std.append(np.std(this_scores))

        scores, scores_std = np.array(scores), np.array(scores_std)
        est_cv.alphas = alphas
        est_cv.random_state = 0
        k_fold = KFold(n_folds)
        scores = []
        for k, (train, test) in enumerate(k_fold.split(X, y)):
            est_cv.fit(X[train, 0:8], y[train, self.cold: 1+self.cold])
            scores.append(est_cv.score(X[test, 0:8],
                                       y[test, self.cold: 1+self.cold]))

```

```
80          return np.mean(scores)*100, est_cv
81
82      return inner_wrap
83
84
85  class Ridge_M(Data, object):
86      """
87          Ridge regression model.
88      """
89      def __init__(self):
90          Data.__init__(self)
91          self.acc, self.est_cv = Ridge_M.fit(self)
92
93      @staticmethod
94      @cross_validator
95      def fit():
96          est = linear_model.Ridge()
97          est_cv = linear_model.RidgeCV()
98          return est, est_cv
99
100     @property
101     def accuracy(self):
102         return self.acc
103
104     def get_prediction(self):
105         return self.est_cv.predict(self.X)
106
107
108 class ANN(Data, object):
109     def __init__(self):
110         Data.__init__(self)
111         self.reg = MLPRegressor(solver='lbfgs', alpha=1e-5,
112                                 hidden_layer_sizes=(5, 2), random_state=1)
113         self.reg.fit(self.X_train,
114                      self.y_train[:,self.cold:self.cold+1].ravel())
115
116     @property
117     def weights(self):
118         return self.reg.coefs_
119
120     @property
121     def accuracy(self):
122         return self.reg.score(self.X_test,
123                 self.y_test[:,self.cold:self.cold+1].ravel())*100
124
125     def get_prediction(self):
126         return self.reg.predict(self.X_test)
127
128
129 # PLOT GRAPHS
130
```

```
131 def plot_model_accuracy(models):
132     X = range(len(models))
133     Y = map(lambda m: m[1].accuracy, models)
134     cycol = cycle('bgrcmk').next
135 
136     for i in range(len(Y)):
137         plt.bar(i, Y[i], 0.5, align='center', color=cycol())
138 
139     for x, y in zip(X,Y):
140         plt.text(x, y+0.05, '%.2f' % y, ha='center', va= 'bottom')
141 
142     labels = [m[0] for m in models]
143     plt.xticks(X, labels)
144     plt.xlabel('Regression models')
145     plt.title('Accuracy (%)')
146     plt.ylim(80,100)
147     plt.show()
148 
149 
150 def plot_feature_correlation(data):
151     """
152         Scatter plot of X vs X and Y vs X.
153 
154         X1  Relative Compactness
155         X2  Surface Area
156         X3  Wall Area
157         X4  Roof Area
158         X5  Overall Height
159         X6  Orientation
160         X7  Glazing Area
161         X8  Glazing Area Distribution
162         y1  Heating Load
163         y2  Cooling Load
164     """
165 
166     cycol = cycle('bgrcmk').next
167     nf = len(data.label) - 2
168 
169     plt.figure(1)   # X vs X
170     for i in range(nf):
171         for j in range(nf):
172             plt.subplot(nf, nf, i*nf+j+1)
173             plt.axis('off')
174             plt.scatter(data.data.ix[:, j:j+1].as_matrix(),
175                         data.data.ix[:, i:i+1].as_matrix(),
176                         color = cycol())
177 
178     plt.figure(2)   # Y vs X
179     for i in range(nf):
180         clr = cycol()
181         plt.subplot(2, 8, i+1)
```

```
182             plt.axis('off')
183             plt.scatter(data.data.ix[:, i:i+1].as_matrix(),
184                         data.data.ix[:, nf:nf+1].as_matrix(),
185                         color = clr)
186 
187             plt.subplot(2, 8, i+9)
188             plt.axis('off')
189             plt.scatter(data.data.ix[:, i:i+1].as_matrix(),
190                         data.data.ix[:, nf+1:nf+2].as_matrix(),
191                         color = clr)
192 
193         plt.show()
194 
195 
196     def plot_pd(data):
197         """
198             Plot probablility densities.
199         """
200         cycol = cycle('bgrcmk').next
201         nf = len(data.label)
202 
203         for i in range(nf):
204             plt.figure(i+1)
205             pdf, bins, patches = plt.hist(data.data.ix[:, i:i+1].as_matrix(),
206                                           20, normed=1, facecolor=cycol(), alpha=0.75)
207             print np.sum(pdf * np.diff(bins))   # sums to 1
208 
209             plt.xlabel(data.label[i])
210             plt.ylabel('Probability')
211             plt.grid(True)
212 
213             plt.show()
214 
215 
216     def plot_spearman(sp_result, data):
217         print [x[0].correlation for x in sp_result]
218         plt.bar(range(len(sp_result)), [x[0].correlation for x in sp_result],
219             tick_label=data.label[:8], align="center")
220         plt.title("Spearman correlation coefficient")
221         plt.show()
222 
223     def get_corr_matrix(data):
224         """
225             Compute the correlation matrix for the dataset
226         """
227         nf = len(data.label)-2
228         corr_matrix = []
229         for i in range(nf):
230             cov = []
231             for j in range(nf):
232                 cov.append(spearmanr(data.data.ix[:, i:i+1].as_matrix(),
```

```
                        data.data.ix[:, j:j+1].as_matrix()).correlation)
        corr_matrix.append(cov)

    return corr_matrix

def plot_corr_matrix(corr_matrix):
    fig = plt.figure(figsize=(20, 20))
    ax = fig.add_subplot(111)
    ax.set_title('Color Map: Correlation matrix')
    plt.imshow(corr_matrix, aspect='auto',
               interpolation='none', origin='lower')
    ax.set_aspect('equal')
    labels = ["X"+str(i) for i in range(9)]
    labels.extend(["y1", "y2"])
    ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
    ax.yaxis.set_major_locator(ticker.MultipleLocator(1))
    ax.set_xticklabels(labels)
    ax.set_yticklabels(labels)
    cax = fig.add_axes([0,0.1,0.95,0.80])
    cax.get_xaxis().set_visible(False)
    cax.get_yaxis().set_visible(False)
    cax.patch.set_alpha(0)
    cax.set_frame_on(False)
    plt.colorbar(orientation='vertical')
    plt.show()

def plot_output_target_graph(y_output_hl, y_output_cl, y_target, name):
    X = range(len(y_target[:, 0:1]))
    dpi = 500
    b = 20
    h = 20

    # Plot HL
    fig = plt.figure(figsize=(b, h), dpi=dpi)

    ax = fig.add_subplot(311)
    ax.set_title('Target')
    plt.xticks(range(1, len(X)+1))
    plt.plot(X, y_target[:, 0:1], 'b+', label='Target HL')
    plt.ylabel("Responses")
    plt.legend(loc='best')

    ax = fig.add_subplot(312)
    ax.set_title('Output')
    plt.xticks(range(1, len(X)+1))
    plt.plot(X, y_output_hl, 'ro', label='Output HL')
    plt.ylabel("Responses")
    plt.legend(loc='best')


    ax = fig.add_subplot(313)
```

```
284         ax.set_title('Error')
285         plt.ylabel("Error")
286         plt.xticks(range(1, len(X)+1))
287         if name=="ann":
288             plt.bar(X, y_target[:, 1:2].ravel()-y_output_hl ,
289                     tick_label=range(1, len(X)+1), align="center")
290         else:
291             plt.bar(X, y_target[:, 1:2]-y_output_hl ,
292                     tick_label=range(1, len(X)+1), align="center")
293         plt.legend(loc='best')
294         fig.savefig("to_%s_hl.png"%(name), dpi=dpi)
295         print "Saved to_%s_hl.png"%(name)
296         # plt.show()
297
298
299         # Plot CL
300         fig = plt.figure(figsize=(b, h), dpi=dpi)
301
302         ax = fig.add_subplot(311)
303         ax.set_title('Target')
304         plt.xticks(range(1, len(X)+1))
305         plt.plot(X, y_target[:, 1:2], 'b+', label='Target CL')
306         plt.ylabel("Responses")
307         plt.legend(loc='best')
308
309         ax = fig.add_subplot(312)
310         ax.set_title('Output')
311         plt.xticks(range(1, len(X)+1))
312         plt.plot(X, y_output_cl, 'ro', label='Output CL')
313         plt.ylabel("Responses")
314         plt.legend(loc='best')
315
316
317         ax = fig.add_subplot(313)
318         ax.set_title('Error')
319         plt.ylabel("Error")
320         plt.xticks(range(1, len(X)+1))
321         if name=="ann":
322             plt.bar(X, y_target[:, 1:2].ravel()-y_output_cl ,
323                     tick_label=range(1, len(X)+1), align="center")
324         else:
325             plt.bar(X, y_target[:, 1:2]-y_output_cl ,
326                     tick_label=range(1, len(X)+1), align="center")
327
328         plt.legend(loc='best')
329         fig.savefig("to_%s_cl.png"%(name), dpi=dpi)
330         print "Saved to_%s_cl.png"%(name)
331         # plt.show()
332
333
334 def main():
```

```
335     data = Data(cold=0)
336     ridge = Ridge_M()
337     ann = ANN()
338     models = [("Ridge", ridge), ("ANN", ann)]
339 
340     for model_name, model_obj in models:
341         print model_name, model_obj.accuracy
342     sp_result = data.get_sp_rank()
343     corr_matrix = get_corr_matrix(data)
344     print corr_matrix
345     plot_corr_matrix(corr_matrix)
346     plot_feature_correlation(data)
347     plot_model_accuracy(models)
348     plot_pd(data)
349     plot_spearman(sp_result, data)
350 
351     ann_y_output_hl = ann.get_prediction()
352     ridge_y_output_hl = ridge.get_prediction()
353     data.cold = 1
354     ann = ANN()
355     ridge = Ridge_M()
356     _ = ridge.accuracy
357     ann_y_output_cl = ann.get_prediction()
358     ridge_y_output_cl = ridge.get_prediction()
359     plot_output_target_graph(ann_y_output_hl,
360                              ann_y_output_cl,
361                              data.y_test,
362                              "ann")
363     plot_output_target_graph(ridge_y_output_hl,
364                              ridge_y_output_cl,
365                              data.y,
366                              "ridge")
367 
368 
369 
370 if __name__ == "__main__":
371     main()
```

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 MODEL ACCURACY

6.1.1 Linear vs Non-linear model

Cross validation

One of the main reasons for using cross-validation instead of using the conventional validation (e.g. partitioning the data set into two sets of 70% for training and 30% for test) is that there is not enough data available to partition it into separate training and test sets without losing significant modelling or testing capability. In these cases, a fair way to properly estimate model prediction performance is to use cross-validation as a powerful general technique. In summary, cross-validation combines (averages) measures of fit (prediction error) to derive a more accurate estimate of model prediction performance.

Accuracy plot

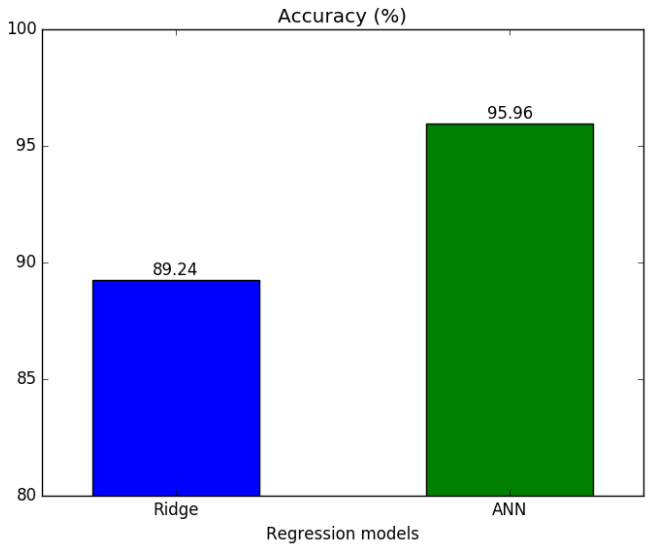

Figure 6.1: Models accuracy : Linear (Ridge) vs Non-linear (ANN)

As observed before, the data has non-linear relationship with responses. The probability distributions demonstrate that none of the variables follows the normal distribution. Furthermore, the scatter plots show that any functional relationship of the input variables and the output variables is not trivial. This suggests that we can reasonably expect that classical learners such as linear regression (linear models) may fail to find an accurate mapping of the input variables to the output variables. Therefore, these insights intuitively show the need for non-linear models. This claim can be justified by comparing the fitting of a linear model (Ridge regression) and non-linear model (ANN) shown below:

Ridge

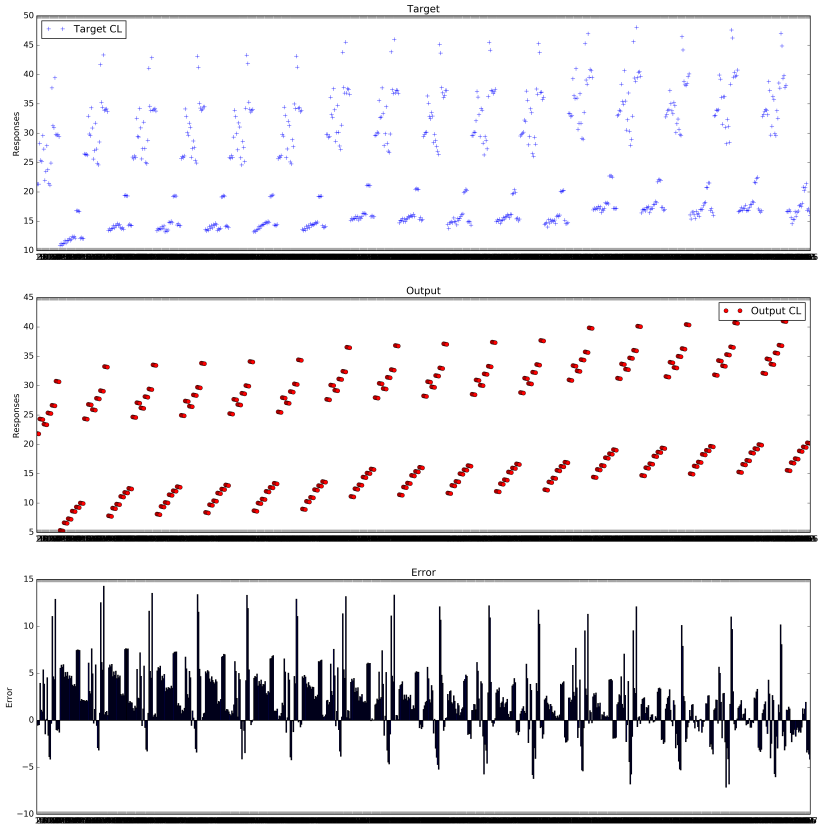

Figure 6.2: Target CL, Output CL and corresponding error (target-output) plot

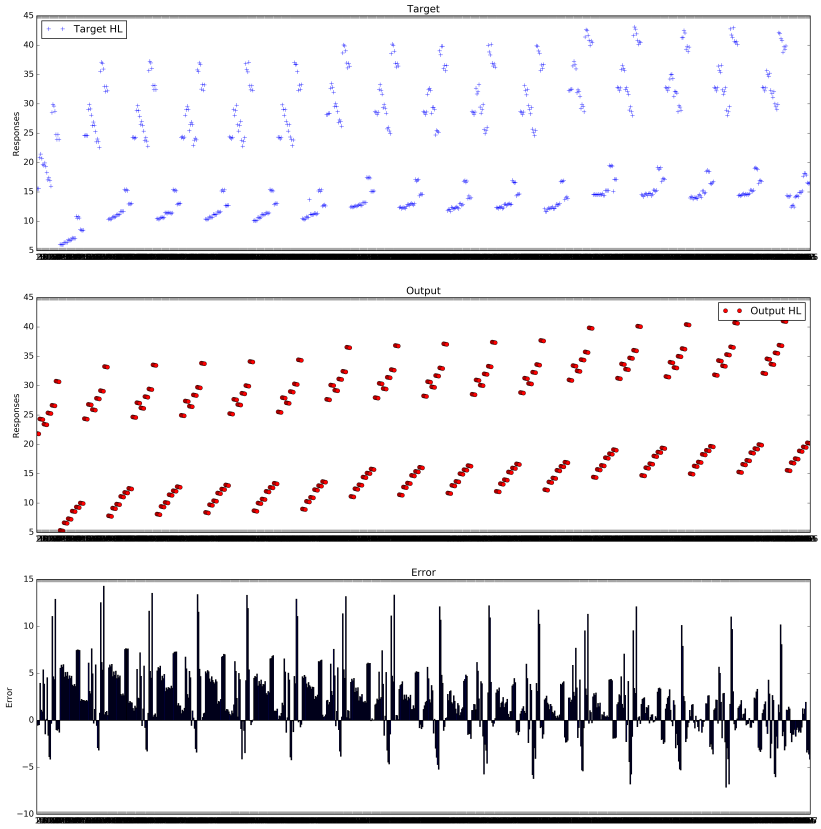

Figure 6.3: Target HL, Output HL and corresponding error (target-output) plot

ANN

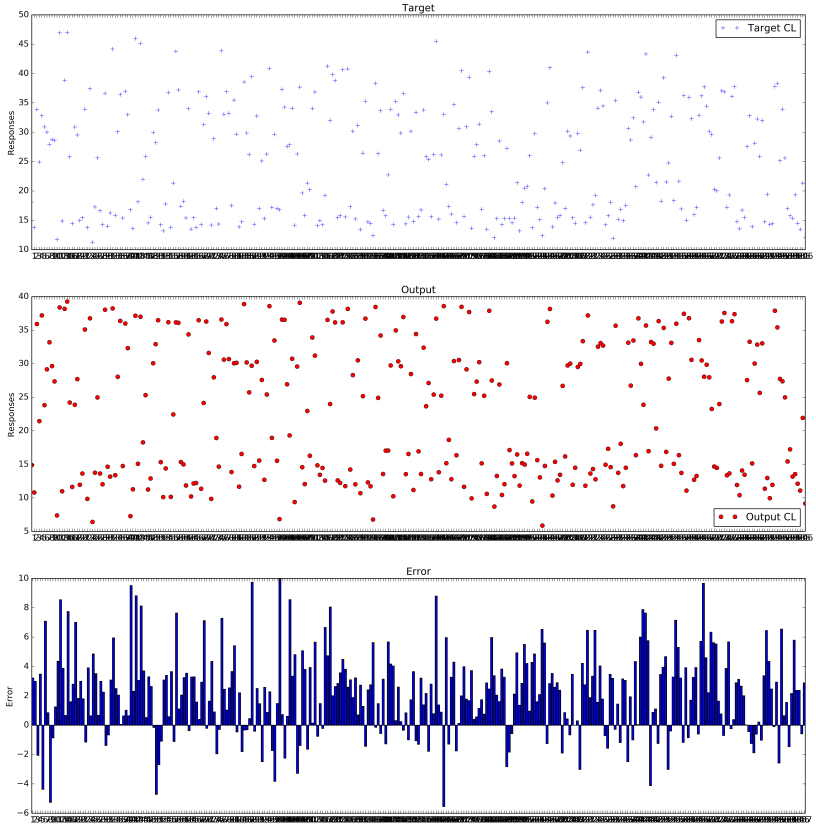

Figure 6.4: Target CL, Output CL and corresponding error (target-output) plot

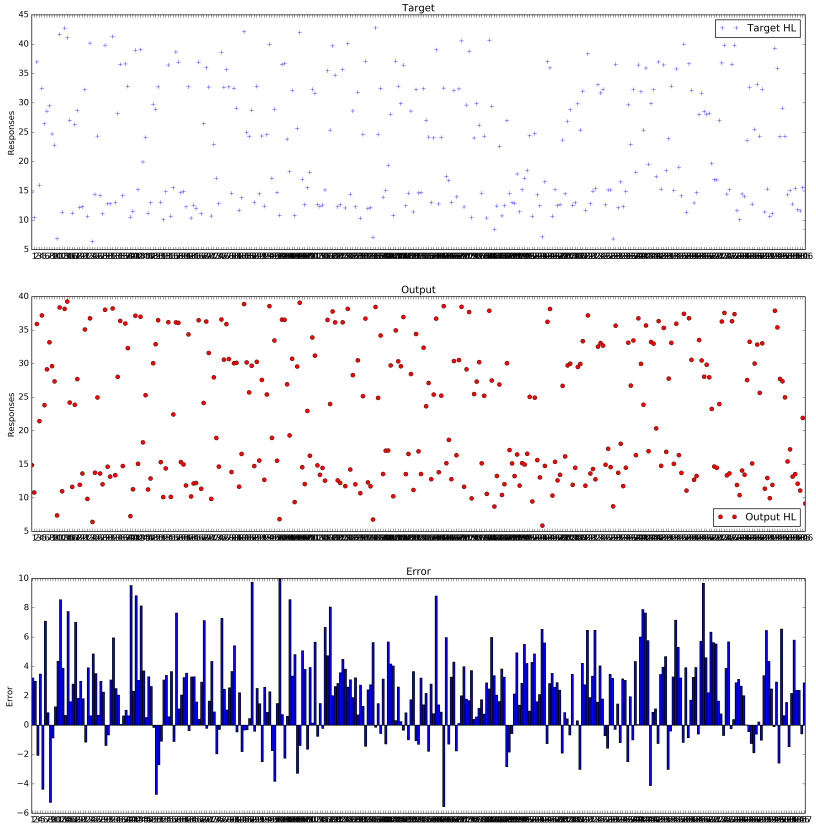

Figure 6.5: Target HL, Output HL and corresponding error (target-output) plot

CHAPTER 7

SUMMARY AND CONCLUSIONS

7.1 SUMMARY

This project investigates the modeling of building heating and cooling load through an intelligent approach that takes into account the Relative Compactness, Surface Area, Wall Area, Roof Area, Overall Height, Orientation, Glazing Area and Glazing Area Distribution. These findings are particularly compelling given the accurate prediction, and also because we can easily infer the output variables in a matter of few seconds without requiring the painstaking design of a new building in a simulation tool. Further exploration of the statistical relationship between eight input variables and the two output variables was done.

7.2 CONCLUSIONS

The ANN massively outperformed Linear regression in finding an accurate functional relationship between the input and output variables. Classical regression settings may fail to account for multi-collinearity, where variables appear to have large magnitude but opposite side sign coefficients with regard to predicting the response.

7.3 SCOPE FOR FUTURE WORK

Improvement in results can be made by taking into account factors like Building occupancy and climate conditions data. For eg. usually larger cooling load occurs between April and September. Furthermore, the effects of occupant's presence on a building's energy consumption vary: people give off heat and pollutants that add to the building's internal gains and then require more cooling load. Hence, consideration should be given to increasing the level of accuracy via looking for the parameters which could not only simulate the occupant presence but also reflect their behaviors in the building.

REFERENCES

[1] L. Perez-Lombard, J. Ortiz, and C. Pout. A review on buildings energy consumption information, energy and buildings. 40(3):394–398, 2008.

[2] W.G. Cai, Y. Wu, Y. Zhong, and H. Ren. China building energy consumption: situation challenges and corresponding measures. 37(6):2054–2059, 2009.

[3] Hunter and J. D. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[4] Crawley, Drury B., Hand, Jon W., Michael Kummert, Griffith, and Brent T. Contrasting the capabilities of building energy performance simulation programs. *Building and Environment*, 43(4):661–673, 2008.

[5] A. Tsanas and A. Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. 49:560–567, 2012.

[6] A. Hani and T. Koiv. Energy consumption monitoring analysis for residential, educational and public buildings. 3(3):231–238, 2012.

[7] B. Dong, Cao, C., Lee, and S. E. Applying support vector machines to predict building energy consumption in tropical region. 37(5):545–553, 2005.

[8] S.S.K. Kwok, R.K.K. Yuen, and E.W.M. Lee. An intelligent approach to assessing the effect of building occupancy on building cooling load prediction. 2011.

[9] Yu, Zhun, Haghighat, Fariborz, Fung, Benjamin C.M., Yoshino, and Hiroshi. A decision tree method for building energy demand modeling. 42(10):1637–1646, 2010.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Press, 2006.