Data mining Project
Report
On

Assessment of heating and cooling load requirements of building using machine learning techniques.

M. Tech. in Mechanical Engineering

by

Group: 20
Jainil Shah (193100071)
Abhijeet Sarkar(193100068)
Saurabh Mandaokar(193100081)

Indian Institute of Technology Bombay,
Powai, Mumbai – 400076.
Oct, 2019

Problem statement :

The building energy consumption has steadily increased over the past decades worldwide and heating, ventilation and air conditioning (HVAC), which have a catalytic role in regulating the indoor climate accounts for most of the energy use in the buildings. One of the ways to deal with such issue is to have 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.

Using advanced dedicated building energy simulation software may provide some degree of reliable solutions but requires user expertise in a particular program. Moreover, the accuracy of the estimated results may vary across different building simulation software. Hence, in practice researchers rely on machine learning tools to study the effect of various building parameters on some variables of interest (e.g. energy).

This project is intended to propose a model that best describes the heating and cooling loads for the building parameters.

Data description :

• Link to the source https://archive.ics.uci.edu/ml/datasets/Energy+efficiency
• Number of observations are 768
• Number of features are 8 and there are 2 outputs

Taking the elementary cube (3.5 × 3.5 × 3.5) 12 building forms were generated where each building form is composed of 18 elements (elementary cubes). The simulated buildings were generated using Ecotect. All the buildings have the same volume, which is 771.75 m3, but different surface areas and dimensions. building materials are the same.
assumptions for the data:
. The simulation assumes that the buildings are in Athens, Greece, residential with seven persons, and sedentary activity (70W).
The internal design conditions were set as follows: clothing: 0.6 clo, humidity: 60%, air speed: 0.30 m/s, lighting level: 300 Lux.
For the thermal properties we used mixed mode with 95% efficiency, thermostat range 19 oC – 24 oC, with 15-20 hours of operation on weekdays and 10-20 hours on weekends.

features:

X1: Relative Compactness
X2: Total Surface Area
X3: Wall Area
X4: Roof Area
X5: Overall Height
X6: Orientation
X7: Glazing area
X8: Glazing area distribution

Outputs:

y1: Heating Load
y2: Cooling Load

Null values checking:

First we checked whether there are any null value cells present in the raw data and none of them were found as shown below:

```
# searching for null values
df.isnull().sum()
```

```
X1    0
X2    0
X3    0
X4    0
X5    0
X6    0
X7    0
X8    0
Y1    0
Y2    0
dtype: int64
```

Checking for collinearity as well as multi-collinearity with outputs variables included :

X6, X7 and X8 show no correlation because they are class labels:

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
X1	1.000	-0.992	-0.204	-0.869	0.828	0.000	0.000	0.000	0.622	0.634
X2	-0.992	1.000	0.196	0.881	-0.858	0.000	0.000	-0.000	-0.658	-0.673
X3	-0.204	0.196	1.000	-0.292	0.281	0.000	-0.000	0.000	0.456	0.427
X4	-0.869	0.881	-0.292	1.000	-0.973	0.000	-0.000	-0.000	-0.862	-0.863
X5	0.828	-0.858	0.281	-0.973	1.000	0.000	0.000	0.000	0.889	0.896
X6	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	-0.003	0.014
X7	0.000	0.000	-0.000	-0.000	0.000	0.000	1.000	0.213	0.270	0.208
X8	0.000	-0.000	0.000	-0.000	0.000	0.000	0.213	1.000	0.087	0.051
Y1	0.622	-0.658	0.456	-0.862	0.889	-0.003	0.270	0.087	1.000	0.976
Y2	0.634	-0.673	0.427	-0.863	0.896	0.014	0.208	0.051	0.976	1.000

fig: multicollinearity between features and correlations with outputs.

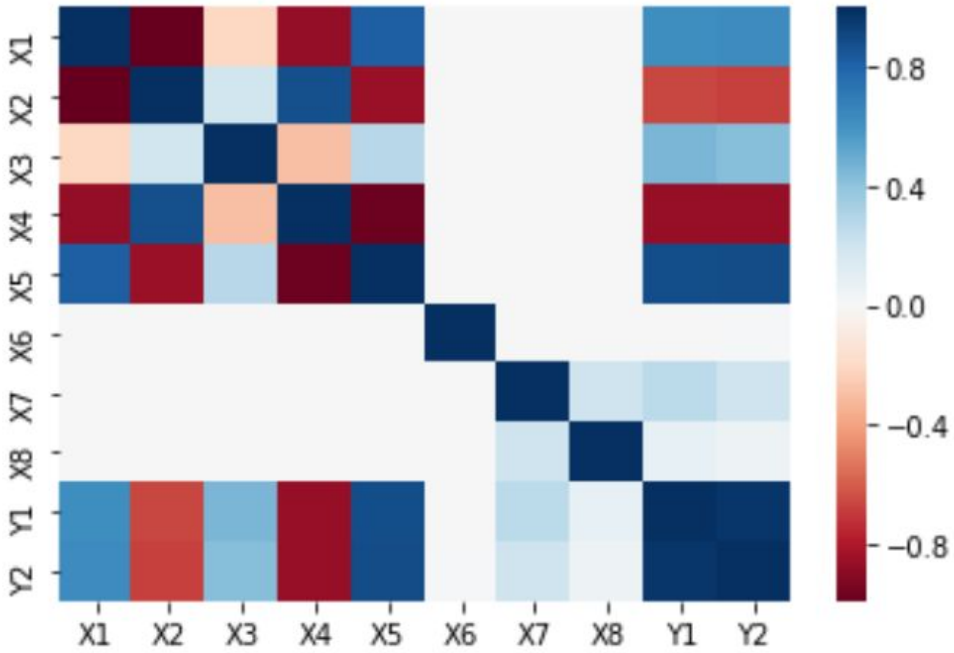


fig: heatmap showing correlations pictorially.

RAW DATA PROCESSING

Knocking off 10% of data :

This would randomly empty 77 cells from each column of the features columns. Re - knocking of the data is prevented by code using if else loop. There is no knocking on the outputs Y1 and Y2 as they being the outputs. This knocked data is saved in a new csv file.

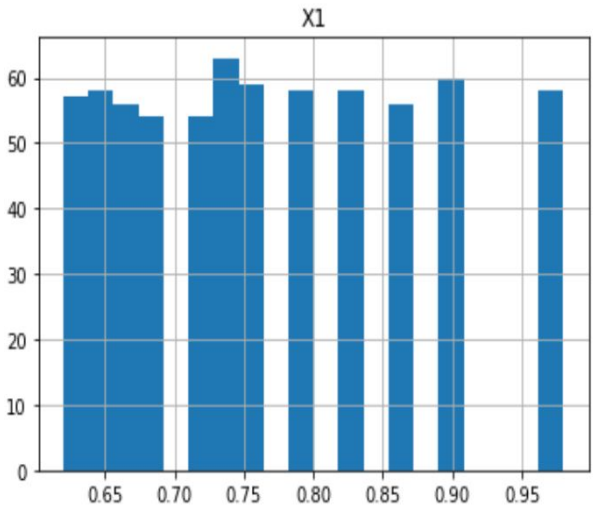
```
df.head(10)
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
0	0.98	NaN	294.0	110.25	7.0	2.0	0.0	0.0	15.55	21.33
1	0.98	NaN	294.0	110.25	7.0	3.0	0.0	0.0	15.55	21.33
2	0.98	514.5	294.0	110.25	7.0	4.0	0.0	0.0	15.55	21.33
3	0.98	NaN	294.0	110.25	7.0	5.0	0.0	0.0	15.55	21.33
4	0.90	563.5	NaN	122.50	7.0	2.0	0.0	NaN	20.84	28.28
5	0.90	563.5	318.5	122.50	7.0	3.0	0.0	0.0	21.46	25.38
6	0.90	563.5	318.5	122.50	7.0	4.0	0.0	0.0	20.71	25.16
7	0.90	563.5	318.5	122.50	7.0	NaN	0.0	0.0	19.68	29.60
8	0.86	588.0	294.0	147.00	NaN	NaN	0.0	0.0	19.50	27.30
9	0.86	588.0	294.0	147.00	7.0	3.0	NaN	NaN	19.95	21.97

IMPUTATION of new data:

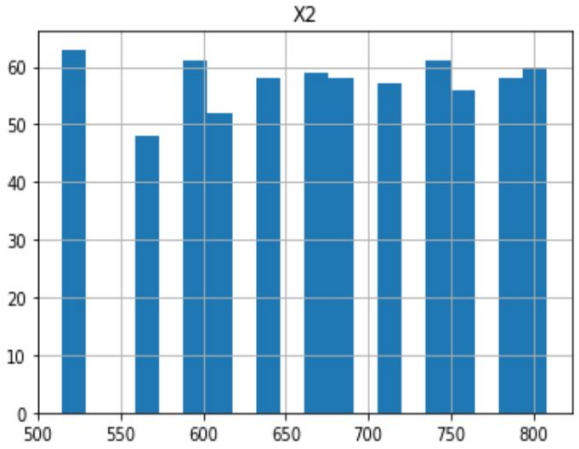
X1 analysis : Relative Compactness

```
count    768.000000
mean       0.686224
std        0.249986
min        0.000000
25%        0.640000
50%        0.740000
75%        0.820000
max        0.980000
Name: X1, dtype: float64
```



X2 analysis : surface area

```
count    768.000000
mean     604.939453
std      218.608806
min        0.000000
25%      588.000000
50%      661.500000
75%      735.000000
max      808.500000
Name: X2, dtype: float64
```



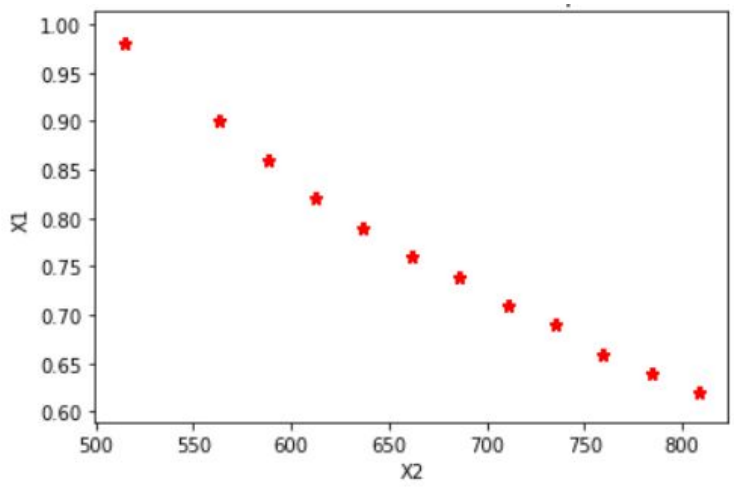
Making data with no nan values: Even though we have 768 rows there must be some rows in which there are no nan value in any cell. We found 322 of such rows, this data was named as pure data.

```
pure_X.describe()
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
count	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000	322.000000
mean	0.768665	668.271739	316.597826	175.836957	5.250000	3.459627	0.238975	2.869565	22.139441	24.370124
std	0.108296	89.046533	45.375233	46.016693	1.752724	1.151966	0.135094	1.539229	9.976683	9.259425
min	0.620000	514.500000	245.000000	110.250000	3.500000	2.000000	0.000000	0.000000	6.050000	10.900000
25%	0.690000	588.000000	294.000000	122.500000	3.500000	2.000000	0.100000	2.000000	12.970000	15.500000
50%	0.750000	673.750000	318.500000	183.750000	5.250000	3.000000	0.250000	3.000000	18.940000	21.895000
75%	0.860000	735.000000	343.000000	220.500000	7.000000	5.000000	0.400000	4.000000	31.542500	32.927500
max	0.980000	808.500000	416.500000	220.500000	7.000000	5.000000	0.400000	5.000000	42.740000	47.010000

Data imputation of X1

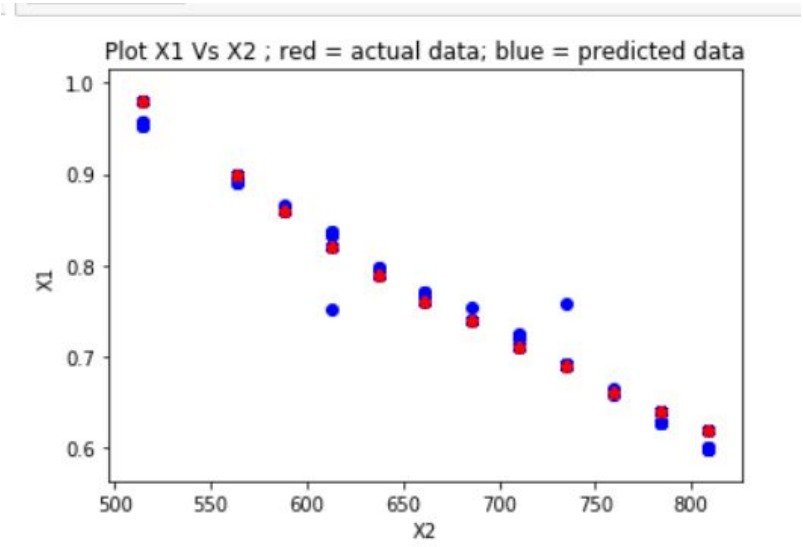
1)Multiple linear Regression: We plotted a graph between X1 and X2 . It is fairly evident from visual analysis of the plot that both of them show a linear relationship.



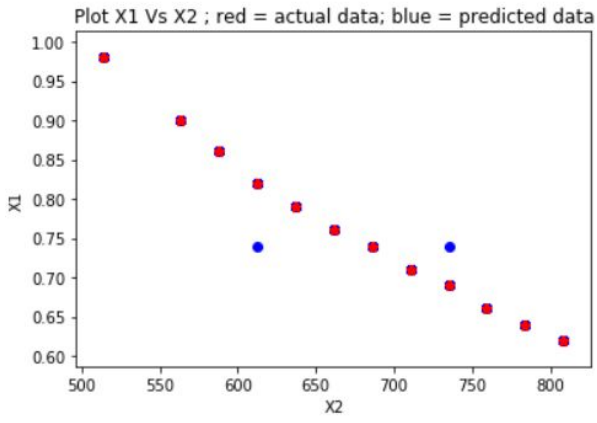
X1 also showed good correlation with X2, X4 and X5 and also with outputs.
First we fit the model using all feature then we did feature selection and found out X1 predictions are best when we use features X2 as well as Y1.Then we trained an MLR model using X2 and Y1 values. To try to predict the X1 using all features and output.
The NaN values which are common in X1 and X2 are filled with mean value of X2 so that we can find all the null values of X1 approx.
This model was used to predict nan values of column x1 in the knocked data dataframe.

OLS Regression Results

Dep. Variable:	X1	R-squared:	0.985
Model:	OLS	Adj. R-squared:	0.985
Method:	Least Squares	F-statistic:	2.071e+04
Date:	Sun, 27 Oct 2019	Prob (F-statistic):	0.00
Time:	18:00:14	Log-Likelihood:	1806.4
No. Observations:	616	AIC:	-3607.
Df Residuals:	613	BIC:	-3593.
Df Model:	2		
Covariance Type:	nonrobust		

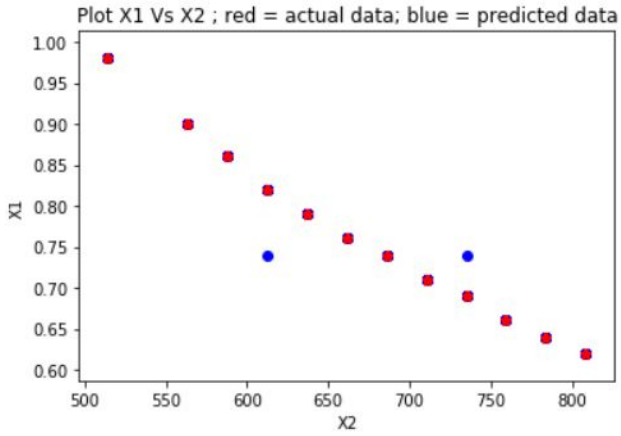


2)Decision Tree Regressor:

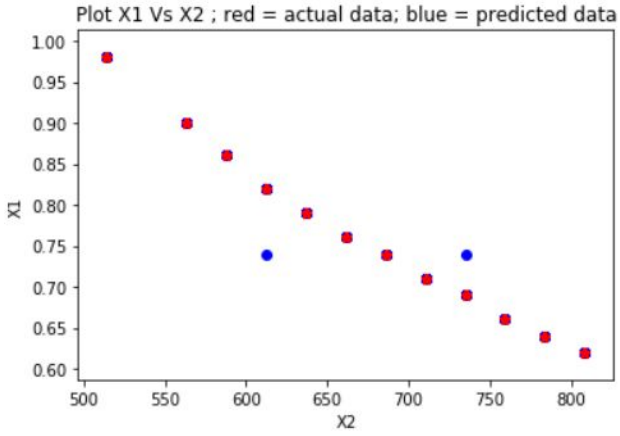


3)KNN Regressor:

In this KNN we used k= 5 to predict values of X1.The predicted values of X1 are compared with that of original data to check the prediction accuracy.



4)Random Forest:



Method	Multiple Linear Regression	Decision Trees Regressor	KNN(K = 5)	Random Forest
Accuracy	0.997561	0.99896292	0.99896	0.99916

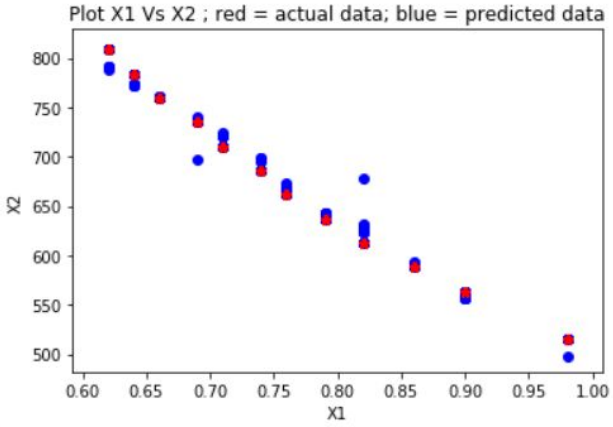
Data imputation of X2 by using modified X1 and Y1

1) Multilinear Regression:

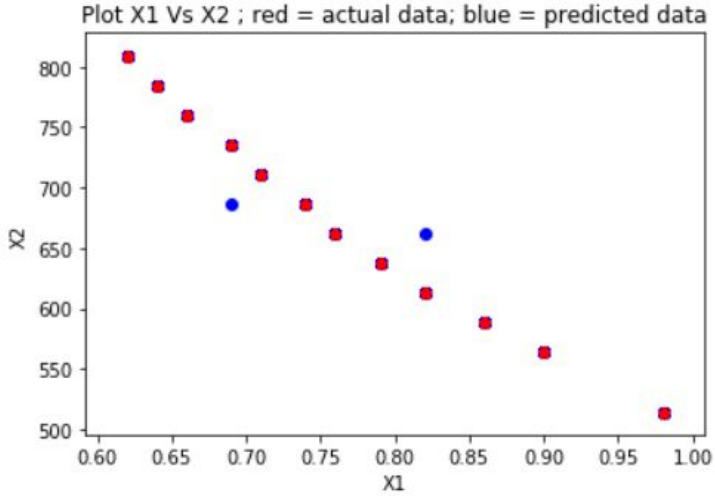
Model fitted by using X1 and Y1 after doing feature selection

OLS Regression Results

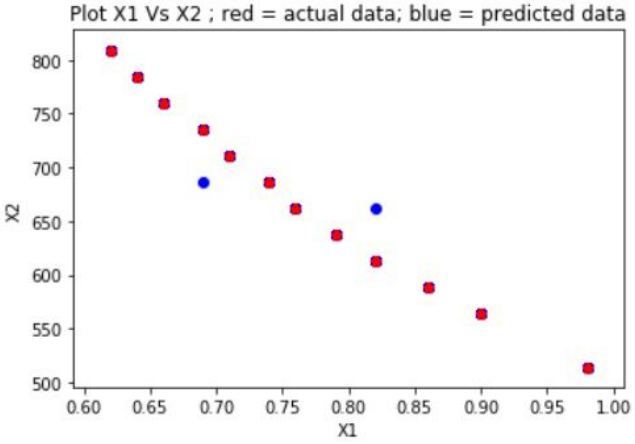
Dep. Variable:	X2	R-squared (uncentered):	0.939
Model:	OLS	Adj. R-squared (uncentered):	0.939
Method:	Least Squares	F-statistic:	5344.
Date:	Sun, 27 Oct 2019	Prob (F-statistic):	0.00
Time:	19:03:51	Log-Likelihood:	-4516.9
No. Observations:	691	AIC:	9038.
Df Residuals:	689	BIC:	9047.
Df Model:	2		
Covariance Type:	nonrobust		



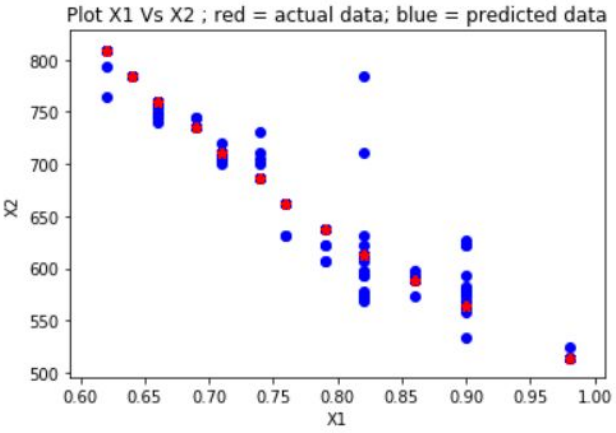
2) Decision Tree Regression:



3)Random Forest Regression:



4)KNN Regression:



Method	Multiple Linear Regression	Decision Tree Regressor	KNN (K = 5)	Random Forest
Accuracy	0.9978	0.9991	0.98791	0.99919

Imputation in X8:

For column X8:

for X8 the values shows signifies classes with values 0 , 1 , 2 , 3 , 4 , 5 as class labels.

0 means no glazing area

1 means uniform with 25% glazing on each side

2 means north:55% on the north side and 15% on each of the other side

3 means east:55% on the east side and 15% on each of the other sides

4 means south:55% on the south side and 15% on each of the other sides

5 means west:55% on the west side and 15% on each of the other sides.

Since these are classes with some specific values so we can't compute it using KNN or regression or any other method to impute the missing data.

Confusion matrix is used to compare data from the original file and the corresponding imputed data column.

'accuracy_score' is used to see model accuracy as well as accuracy off predictions compared to original data before knocking.

since X8 is a class data encoded as 0,1,2,3,4 and 5 as class labels.

We used Y1 and Y2 ,as our explanatory features for imputations in X8.

X8 predictions using LDA:

```
cm(df_ori.X8, X8)
```

```
array([[ 46,   2,   0,   0,   0,   0],
       [  1, 135,   0,   0,   4,   4],
       [  0,   6, 129,   0,   4,   5],
       [  0,   9,   0, 122,   2,  11],
       [  0,   5,   0,   0, 135,   4],
       [  0,   5,   0,   0,   4, 135]],
```

```
asc(df_ori.X8, X8)
```

```
0.9140625
```

Where cm is confusion matrix and asc is accuracy.
prediction accuracies as compared to original data = 0.227

X8 predictions using QDA:

```
cm(df_ori.X8, X8)
```

```
array([[ 46,   1,   0,   0,   1,   0],
       [  1, 137,   0,   5,   0,   1],
       [  0,   6, 129,   5,   1,   3],
       [  0,   7,   0, 130,   0,   7],
       [  0,   5,   0,   6, 130,   3],
       [  0,   5,   0,   6,   0, 133]],
```

```
asc(df_ori.X8, X8)
```

```
0.91796875
```

model accuracy = 0.917 and prediction accuracy as compared to the original data is 0.234

X8 predictions using KNN with k = 5:

```
cm(df_ori.X8, X8)
```

```
array([[ 47,   1,   0,   0,   0,   0],
       [  0, 141,   1,   0,   1,   1],
       [  0,   5, 132,   1,   4,   2],
       [  0,   4,   6, 125,   2,   7],
       [  0,   6,   3,   3, 131,   1],
       [  0,   1,   2,   5,   3, 133]],
```

```
asc(df_ori.X8, X8)
```

```
0.9231770833333334
```

X8 predictions using KNN with k = 10:

```
cm(df_ori.X8, X8)
```

```
array([[ 47,   1,   0,   0,   0,   0],
       [  0, 141,   2,   0,   0,   1],
       [  0,   4, 134,   2,   3,   1],
       [  0,   5,   3, 125,   7,   4],
       [  0,   2,   4,   2, 132,   4],
       [  0,   2,   1,   5,   3, 133]],
```

```
asc(df_ori.X8, X8)
```

```
0.9270833333333334
```

X8 predictions using ANN:

```
Wall time: 25.5 s
```

```
cm(df_ori.X8, X8)
```

```
array([[ 46,   0,   1,   0,   0,   1],
       [  0, 139,   0,   3,   0,   2],
       [  0,   9, 133,   0,   0,   2],
       [  0,  16,   4, 123,   0,   1],
       [  0,   7,   4,   2, 130,   1],
       [  0,   4,   3,   4,   0, 133]],
```

```
asc(df_ori.X8, X8)
```

```
0.9166666666666666
```

ANN was used to predict X8 values in which we experimented the results with various number of hidden layers and in each case we either used 5 or 10 nodes in each layer. The number of iterations were 100,200,500 and 1000.
We found out that the best combination for more accuracy of prediction was the ANN with 5 hidden layer , each layer consisting of 10 neurons . the number of iterations was 200, which took 25.5 seconds for computation.

X8 predictions using decision trees:

```
Wall time: 193 ms

In [109]: cm(df_ori.X8, X8)

Out[109]: array([[ 48,   0,   0,   0,   0,   0],
                 [  0, 140,   1,   1,   1,   1],
                 [  0,   3, 132,   2,   6,   1],
                 [  0,   4,   4, 124,   2,  10],
                 [  0,   4,   5,   1, 132,   2],
                 [  0,   2,   2,   4,   1, 135]], dtype=int64)

In [110]: asc(df_ori.X8, X8)

Out[110]: 0.92578125
```

preciseness to original data = 0.92578

In the below table we considered 2 accuracies:

1. The model accuracy considering x_train , y_train itself as the testing data.
2. The model accuracy on the original data set with unknocked values.

Method to predict X8	**Model accuracy (of fitting data)**	**Model accuracy (on original unknocked data)**
LDA	0.23733	0.22786
QDA	0.24023	0.23437
KNN with K = 5	0.43415	0.41406
KNN with K = 10	0.35021	0.34244
ANN	0.24023	0.23177
Decision trees	0.99276	0.91927

therefore to predict X8 values we would consider decision trees because of its predicted classes for X8 are very accurate.

Imputation of X6:

X6 using LDA :

the predictors used to predict X6 are Y1 and Y2.
accuracy as compared to original data = 0.2669
model accuracy = 0.9231

```
([[172,  13,   0,   7],
  [  0, 186,   0,   6],
  [  3,  11, 170,   8],
  [  3,   8,   0, 181]],
```

X6 using QDA :

confusion matrix for QDA predictions
accuracy as compared to original data = 0.2708
model accuracy = 0.9231

X6 Using ANN with 10 neurons in hidden layer and 5 hidden layer:

confusion matrix is as shown , model accuracy = 0.9231
but predictions accuracy with original data is 0.3346

Imputation of X6 using Decision Tree classifier.

model accuracy = 0.925
but predictions accuracy with original data is 0.919

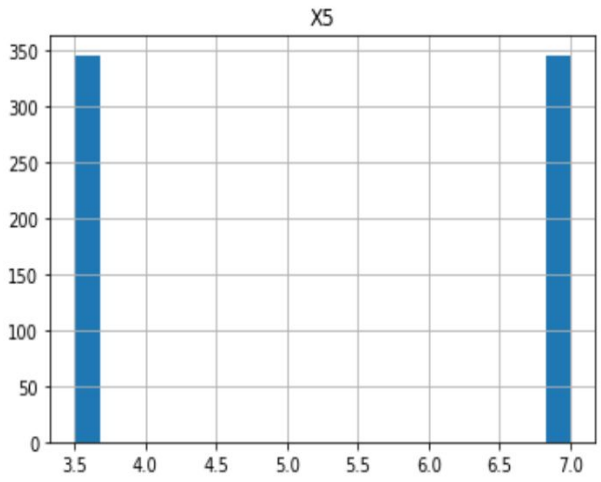
Method to predict X6	Accuracy when Compared to Original X6	Model accuracy (on fitted data)
LDA	0.923	0.266
QDA	0.923	0.2708
ANN	0.923	0.3346
Decision trees classifier	0.925	0.991

decision trees model was chosen for X6 imputation as it predicts classes with greater accuracy.

Imputation of X5:

X5 analysis : overall height

```
count    768.000000
mean       4.739583
std        2.294605
min        0.000000
25%        3.500000
50%        3.500000
75%        7.000000
max        7.000000
Name: X5, dtype: float64
```



X5 takes only two values 7 and 3.5 as overall heights.

1)X5 using KNN:

X5 consists of two values 7 and 3.5 which depicts ceiling height. We first encoded these values using label encoder .

Then we first predicted these values using KNN classifier. The training was done on a by removing all nan value cells from the knocked data. The prediction accuracy using KNN classifier is quite good as shown below.

```
# Finding the model accuracy by original data
new_X_train = df1.drop(['X3','X6','X7','X8','X5'],axis=1)
```

```
[[384   0]
 [  1 383]]
```

```
In [47]: print(asc(encoded_df1_X5,encoded_X5_knn_X5))
```

```
0.9986979166666666
```

X5 using logistic regression:

The features used for predicting X5 are X1,X2,X4,Y1and Y2.

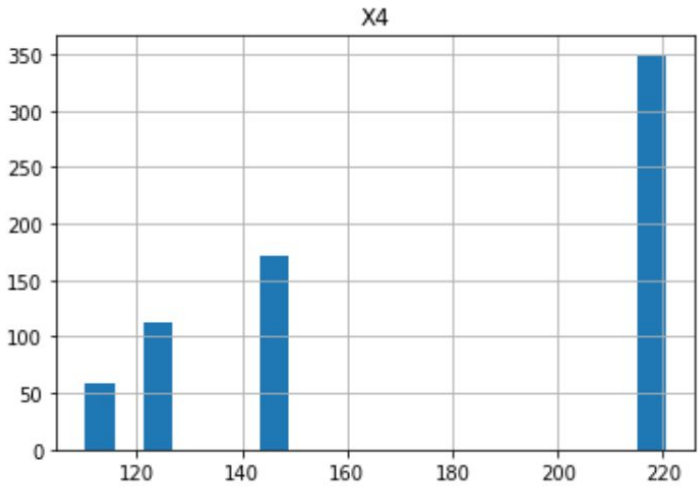
The prediction accuracy and confusion matrix is shown below.

```
In [52]: X5_lr.X5[X5_lr.X5.isnull()]= predicted_X5_values_lr
         encoded_X5_lr_X5 = lab_enc.fit_transform(X5_lr.X5)
         print(confusion_matrix(encoded_df1_X5,encoded_X5_lr_X5))

         [[384   0]
          [  5 379]]

In [53]: print(asc(encoded_df1_X5,encoded_X5_lr_X5))

         0.9934895833333334
```

Imputation of X4:

X4 analysis: roof area

```
count    768.000000
mean     158.404622
std       68.079273
min        0.000000
25%      122.500000
50%      147.000000
75%      220.500000
max      220.500000
Name: X4, dtype: float64
```



the features selected for predictions of X4 are X1,X2,X5,Y1 and Y2.

X4 imputation using KNN:

The prediction values accuracies were obtained by comparing with original data.

```
r2_score(df1.X4,knn_df_x4.X4)
```

```
: 0.9999846547314578
```

Knn regression with K = 5 gave highly accurate data predictions.

X7 imputation:

X7 values tell us the glazing are as a percentage of floor area . the X4 column has already imputed data. Here (floor area = roof area) is considered. And so to use exact X7 values as glazing area we multiplied X7 column elements with the X4 (roof area) column. These changes were made in the data frame for calculations. The training of the models used for X7 imputation was done using the rows with no nan values .
X7 imputation uses X1,X2,X4,X5,Y1,Y2 as its features.

X7 imputation using multi linear regression:

OLS Regression Results

Dep. Variable:	y	R-squared:	0.479
Model:	OLS	Adj. R-squared:	0.474
Method:	Least Squares	F-statistic:	104.8
Date:	Thu, 24 Oct 2019	Prob (F-statistic):	2.06e-93
Time:	18:16:46	Log-Likelihood:	-3026.5
No. Observations:	691	AIC:	6067.
Df Residuals:	684	BIC:	6099.
Df Model:	6		
Covariance Type:	nonrobust		

Here we see that the R-squared value for the OLS regression is very less. But the predicted values for X7 are very much closer to the actual values as shown below.

```
5]: r2_score(df1.X7, X7_mlr.X7)
```

```
5]: 0.9430062882864373
```

X7 imputation using KNN Regressor:

R2_score was calculated with predicted data and original data , and predictions are highly accurate.

```
: r2_score(df1.X7, X7_knn.X7)
```

```
: 0.9986850321395776
```

X7 imputation using KNN Classifier:

X7 values are encoded using label encoder as follows.

```
lab_enc = preprocessing.LabelEncoder()
encoded_knn_y_train = lab_enc.fit_transform(knn_y_train)
```

Confusion matrix :

```
[[ 48   0   0   0]
 [  0 240   0   0]
 [  0   0 240   0]
 [  0   0   0 240]]
```

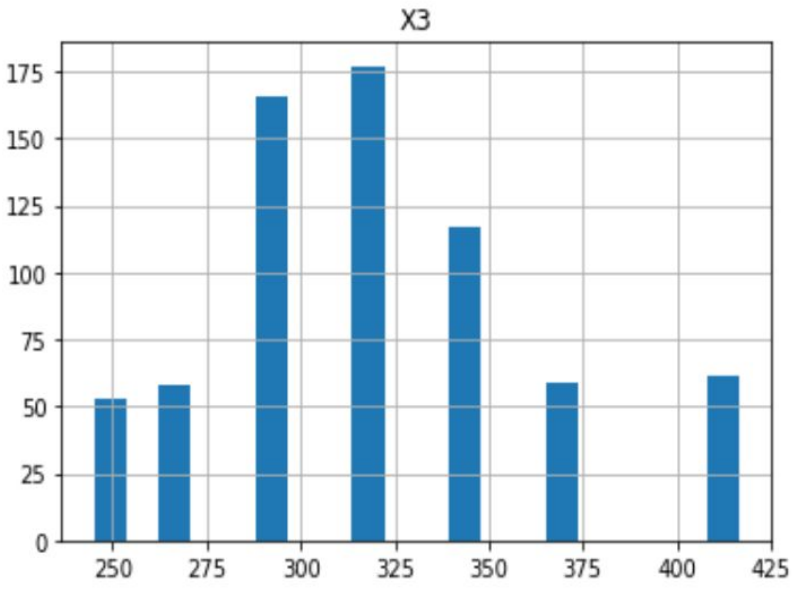
the confusion matrix shows very accurate prediction for X7

Knn classifier proves to be a better imputation model than imputations considering the encoded data because the confusion matrix shows 100% match with original data.

Imputation of X3:

X3 analysis: wall area

```
count    768.000000
mean     286.726562
std      104.434103
min        0.000000
25%      269.500000
50%      318.500000
75%      343.000000
max      416.500000
Name: X3, dtype: float64
```



For X3 we get almost normal curve , we can use mean here as the deviation is very less. Mode is also mean

.

For imputation in X3 we used features X1,X2,X4,X5,X7 and Y1 and Y2.

Imputation of X3 by linear regression:

OLS Regression Results

Dep. Variable:	X3	R-squared:	0.996
Model:	OLS	Adj. R-squared:	0.996
Method:	Least Squares	F-statistic:	2.659e+04
Date:	Thu, 24 Oct 2019	Prob (F-statistic):	0.00
Time:	19:19:57	Log-Likelihood:	-1652.4
No. Observations:	691	AIC:	3321.
Df Residuals:	683	BIC:	3357.
Df Model:	7		
Covariance Type:	nonrobust		

X3 imputation results:

imputation method	R2 score
linear regression	0.999
decision tree regressor	1
KNN regressor	1

MODEL GENERATION:

we first normalised and applied Principal component analysis so that the data is reduced to a 2D array. this will help us in data visualisation .

```
pca = PCA(n_components=2)
normalizer = Normalizer(copy=False)
X_norm = normalizer.fit_transform(X)
X_reduced = pca.fit_transform(X_norm)
```

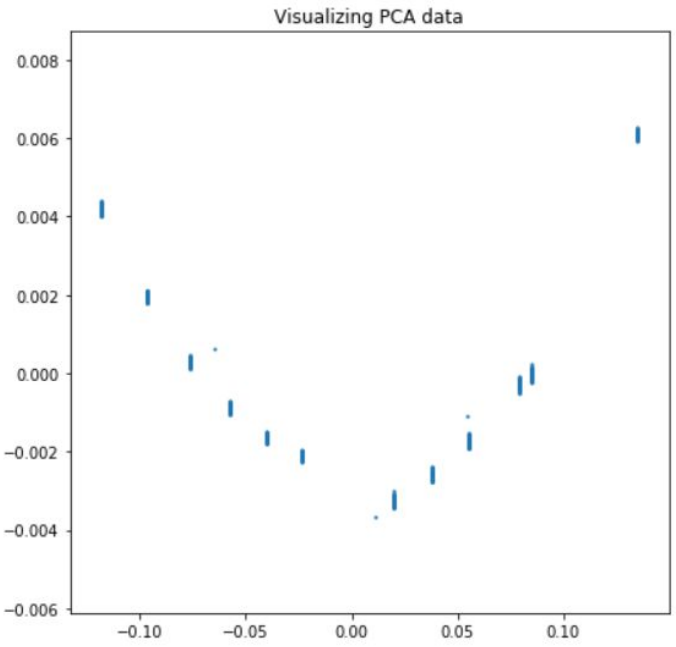


fig: visualisation of data using PCA.

Here we can see that whole data is very separate and divided approximately in 12 clusters

Reducing dimensions of training and testing data sets:

To fit the appropriate model and to select model we need to reduce dimensions of the training data and testing set .

we made reduced data sets for training and testing as follows:

1. for heating load:

explanatory features = X2,X3,X4,X5,X6,X7,X8. which is reduced to X1_train_reduced and X1_test_reduced as shown below.

for heating load

```
# for X1_train
pca = PCA(n_components=1)
normalizer = Normalizer(copy=False)
X1_train_norm = normalizer.fit_transform(X1_train)
X1_train_reduced = pca.fit_transform(X1_train_norm)

# for X1_test
X1_test_norm = normalizer.transform(X1_test)
X1_test_reduced = pca.transform(X1_test_norm)
```

2. for cooling load :

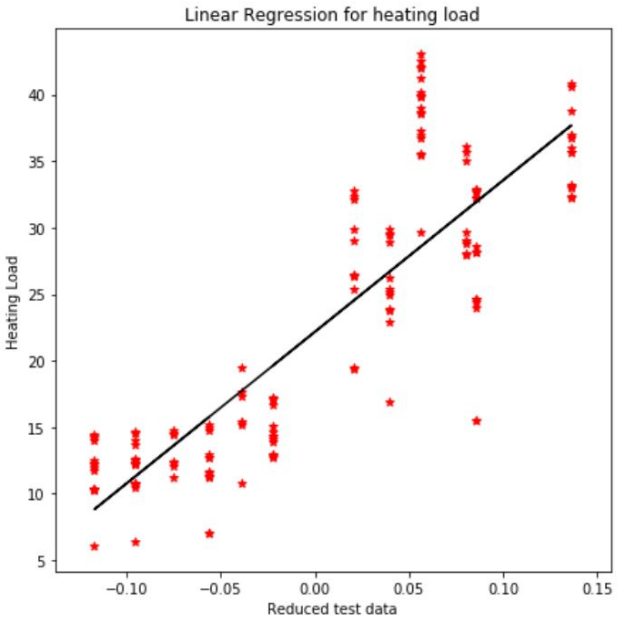
explanatory features = X2,X3,X4,X5,X6,X7,X8. which is reduced to X2_train_reduced and X2_test_reduced as shown below.

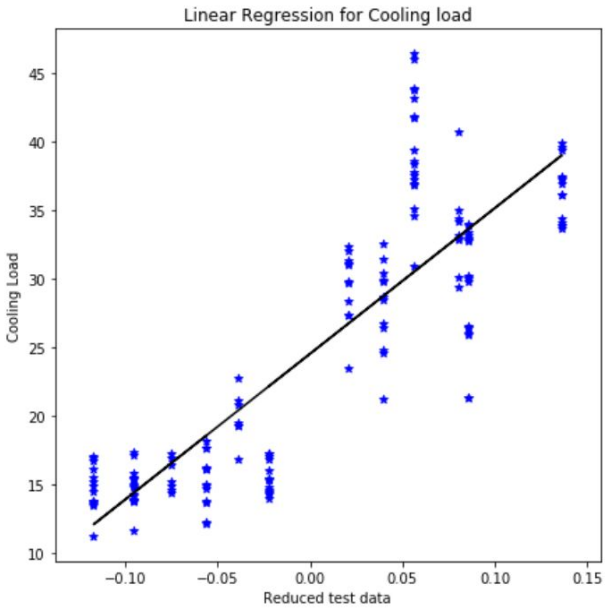
for cooling load

```
# for X2_train
pca = PCA(n_components=1)
normalizer = Normalizer(copy=False)
X2_train_norm = normalizer.fit_transform(X2_train)
X2_train_reduced = pca.fit_transform(X2_train_norm)

# for X1_test
X2_test_norm = normalizer.transform(X2_test)
X2_test_reduced = pca.transform(X2_test_norm)
```

trying to fit a linear regression on reduced data sets:

a linear model on reduced data sets for prediction of heating load as well as cooling load.





it is clearly seen from the above figures that a linear regression will be unable to predict the heating load properly. this is also evident from below OLS summary. the R-squared values are too less to be acceptable.

```
Model summary for Heating Load
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.128
Model:                            OLS   Adj. R-squared (uncentered):              0.126
Method:                 Least Squares   F-statistic:                              89.64
Date:                Fri, 25 Oct 2019   Prob (F-statistic):                    6.09e-20
Time:                        09:31:14   Log-Likelihood:                         -2788.7
No. Observations:                 614   AIC:                                      5579.
Df Residuals:                     613   BIC:                                      5584.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
```

```
Model summary for cooling Load
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.095
Model:                            OLS   Adj. R-squared (uncentered):              0.094
Method:                 Least Squares   F-statistic:                              64.39
Date:                Fri, 25 Oct 2019   Prob (F-statistic):                    5.22e-15
Time:                        09:31:14   Log-Likelihood:                         -2847.6
No. Observations:                 614   AIC:                                      5697.
Df Residuals:                     613   BIC:                                      5702.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
```

fitting KNN regression on the reduced data sets:

we checked the model accuracies for various values of ‘K’ for knn regression. in a loop we tested accuracies for predictions from till K = 30. and it was found out that for K = 28 we got a marginally high accuracy for both heating load and cooling load.

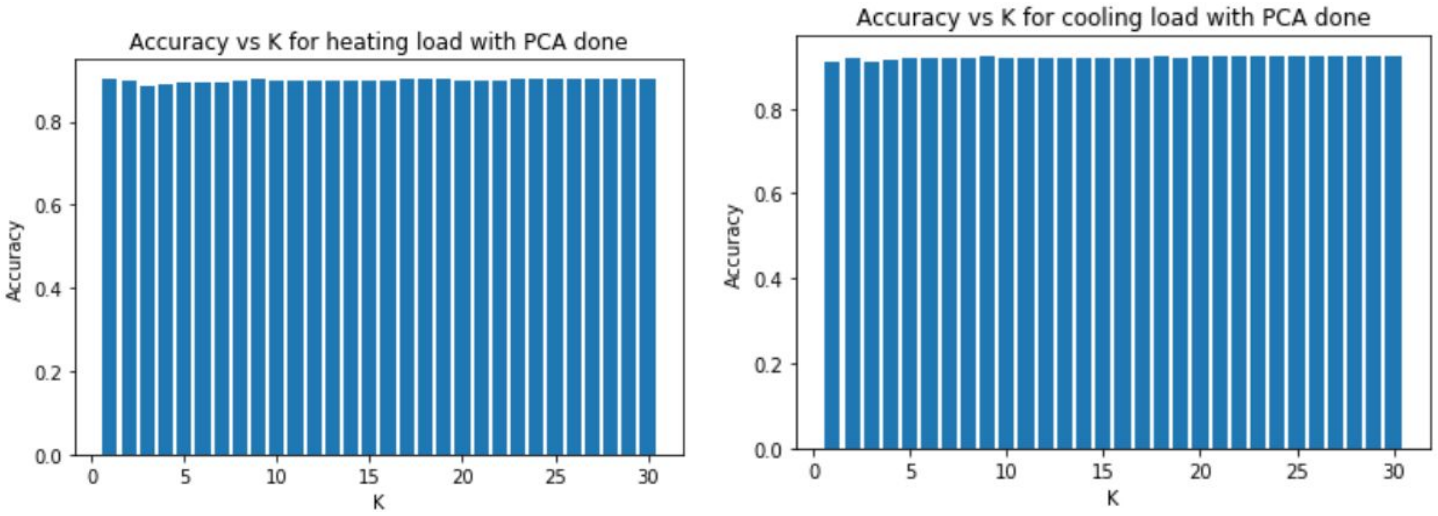


fig: KNN prediction accuracies for various ‘K’ values for heating and cooling load.

a plot of knn results is shown below. the heating load data is in red and cooling load is in blue. the predictions are shown in black colour.

model accuracy in case of KNN :
1. for heating load = 0.9043

2. for cooling load = 0.9250

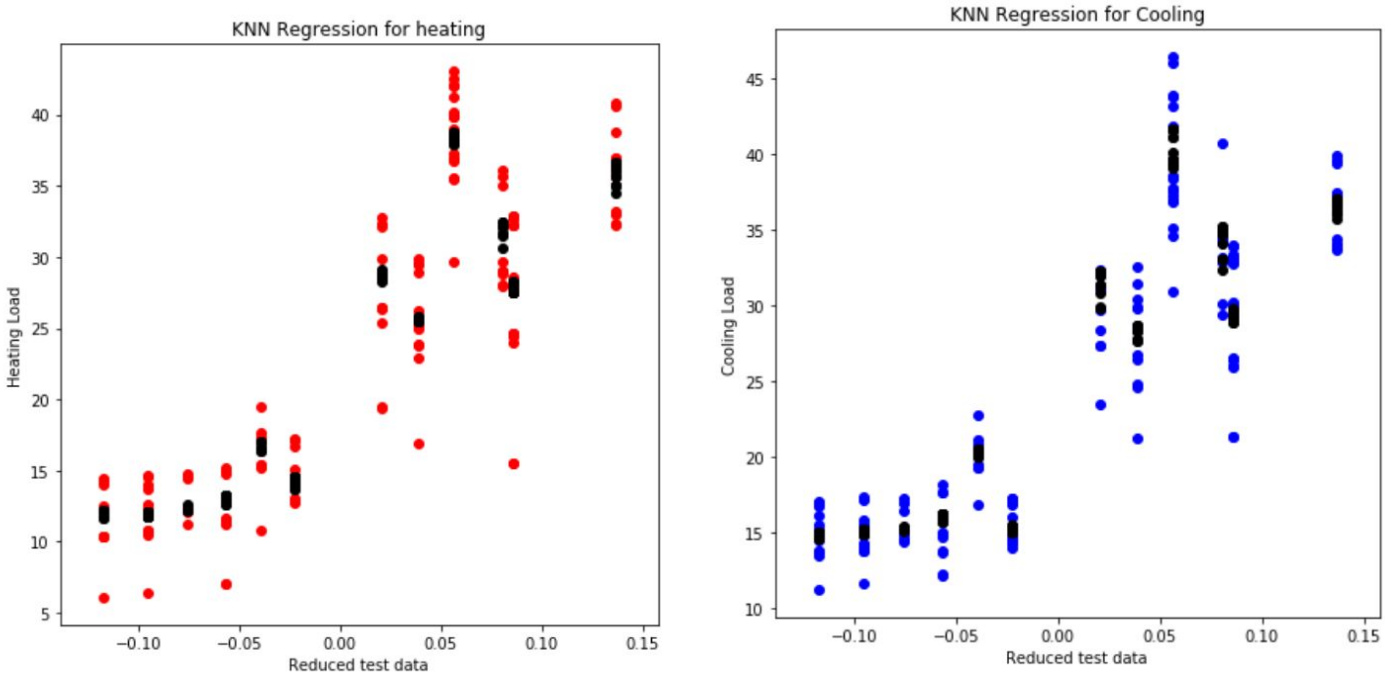
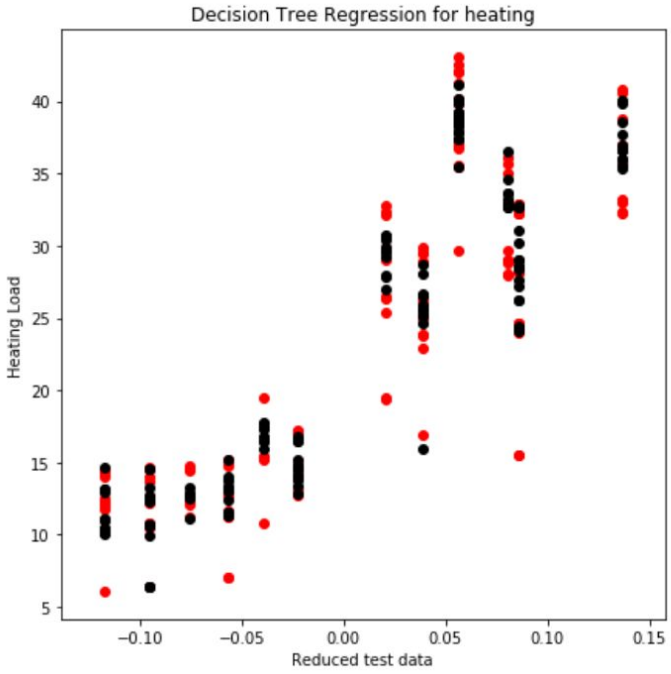
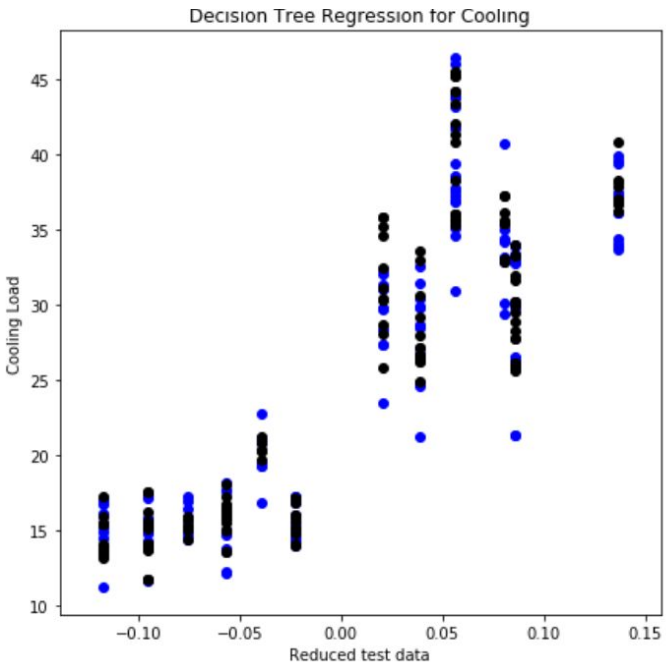


fig: KNN regression predictions for heating and cooling loads.

fitting decision trees on reduced data sets:

a plot of Decision trees results is shown below. the heating load data is in red and cooling load is in blue.
the predictions are shown in black colour.



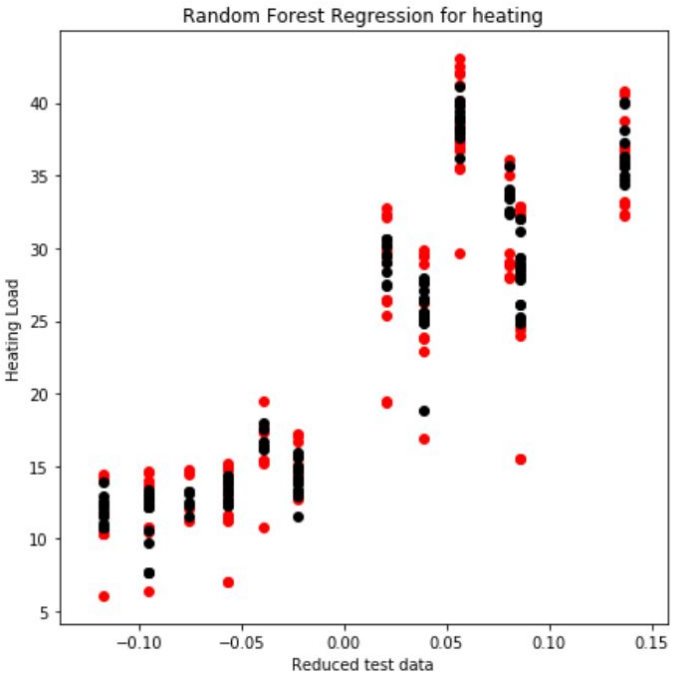


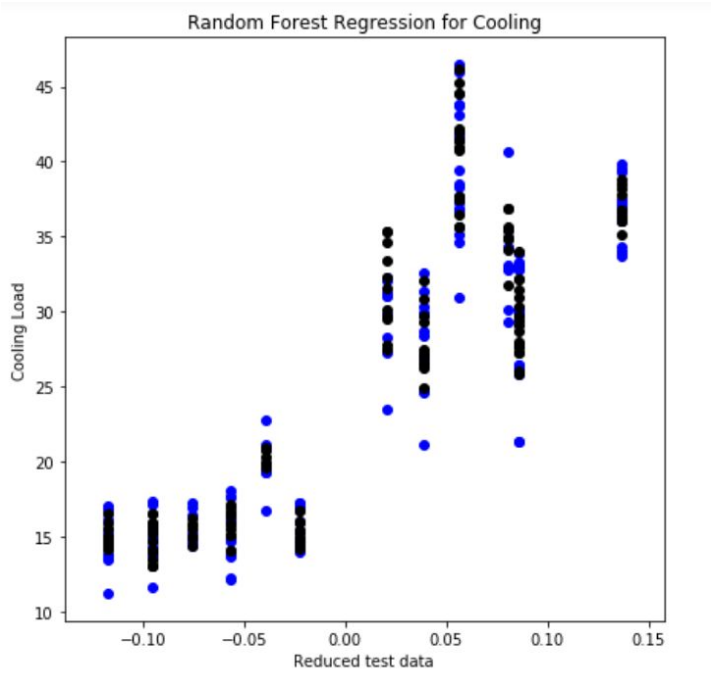
model accuracy in case of decision trees :

1. for heating load = 0.8708
2. for cooling load = 0.9056

fitting random forests in the reduced data sets:

a plot of random forest results is shown below. the heating load data is in red and cooling load is in blue. the predictions are shown in black colour.





model accuracy in case of random forests :

1. for heating load = 0.8842
2. for cooling load =0.9191

prediction model	output	r2_score
linear regression	heating load	0.7268
	cooling load	0.729
KNN regression	heating load	0.897
	cooling load	0.921
Decision trees	heating load	0.872
	cooling load	0.907
random forests	heating load	0.885
	cooling load	0.913

table : summaries for models based on reduced data.

Since all KNN, Decision Tree and Random Forest gave almost the same and decent results, so applying these methods for the non reduced data. And rejecting Linear Regression due to very low R squared value.

Fitting Non reduced imputed Data in selected 3 models

1. fitting knn regression on imputed data:

all the features were used to predict the outputs . first he ‘K’ value for KNN regression was decided by comparing model accuracies with for each k value upto K = 30.
it was found that for k = 5 we get maximum accuracy for predictions of heating and cooling loads as shown below.

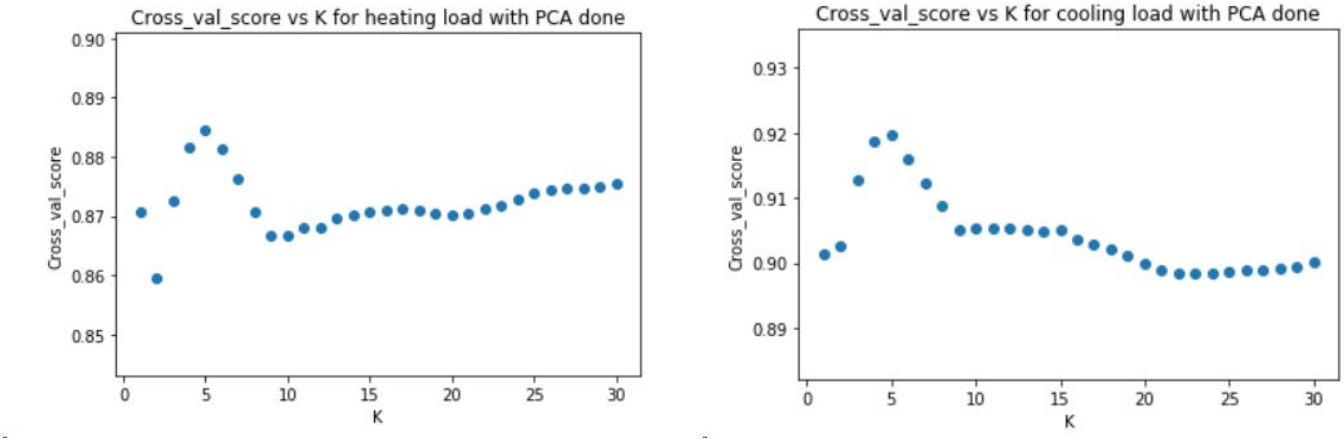


fig: Knn accuracies for various k values.

fitting Decision trees on imputed data:

in the case of decision trees , it was required to know that at what optimum depth the predictions will be accurate . for this purpose we checked accuracy of fit of decision trees for a range of depth values in a loop and the result was plotted as a graph as shown below.

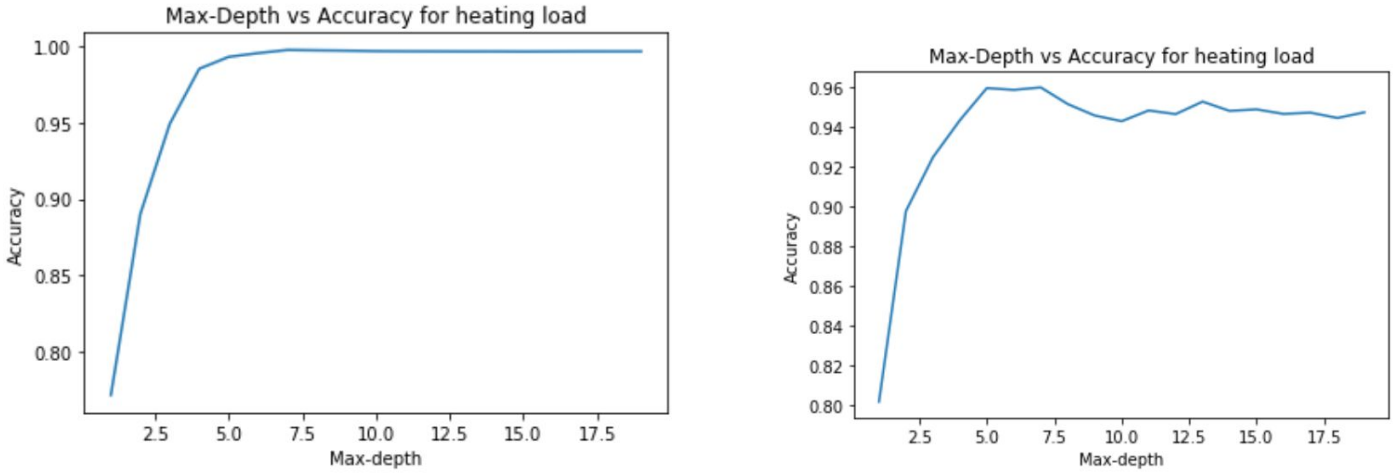


fig: plot of model accuracy by decision tree at various depth values.

it is seen that at a depth of 7 we get highest accuracy of the decision tree predictions.

model accuracy in case of decision trees at a depth of 7 :

1. for heating load = 0.997
2. for cooling load =0.959

fitting random forests on imputed data:

similar to the case of decision trees, it was required to know that at what optimum depth the predictions will be accurate . for this purpose we checked accuracy of fit of decision trees for a range of depth values in a loop and the result was plotted as a graph as shown below.

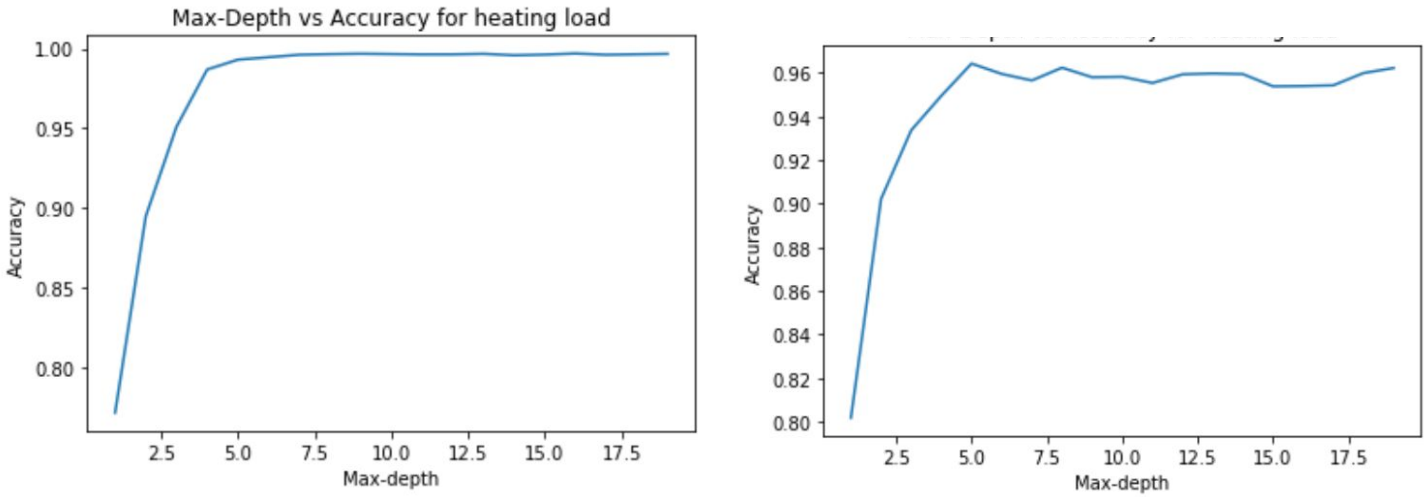


fig: model accuracies by random forests at various depths for heating and cooling load respectively.

it is seen that at a depth of 16 we get highest accuracy for random forest predictions of heating load. at a depth of 5 we get highest accuracy of prediction for cooling load .
Accuracy for heating data model is 0.9964
Accuracy for cooling data model is 0.9630

prediction model	**output**	**r2_score**	**RMSE**	**cross validation score**
KNN regression	heating load	0.946	2.4369	0.896 at k=5
	cooling load	0.952	2.115	0.912 at k=5
Decision trees regressor	heating load	0.996	0.623	0.969
	cooling load	0.944	2.287	0.919
random forests	heating load	0.996	0.656	0.978
	cooling load	0.963	1.994	0.943

table : summaries for models based on imputed data.

Conclusions:

- Imputation of X1 (relative compactness): Since the correlation of X1 is very high with X2 and outputs, hence its imputation is done with X2 (mean in NaN values temporarily because of distribution such that mean is approximately equal to mode), Y1 using feature selection. Multiple Linear Regression, Decision Tree Regressor, KNN regressor and Random Forest regressor were used to impute data. But on comparing with the original data, Random Forest gave high of 0.999, i.e almost all NaN values were correctly imputed.
- Imputation of X2 x: Using the above imputed values of X1 and using Y1, X2 was imputed to replace the temporary mean values. To execute this we use different techniques like Multi linear regression, Decision tree regressor, KNN regressor and Random forest regressor. Out of which Random Forest regressor was used as it gave almost equal data compared to original data i.e. with 0.999 accuracy means almost all NaN values were imputed correctly.
- Imputation of X5 (overall height): Here X1(imputed), X2(imputed), X4(mode in NaN values temporarily), Y1 and Y2 were used. But in data creation from software, only two values 7 and 3.5 meters were taken. So to impute we used this as classes, to execute this we used Level Encoder for X5 data. After this techniques like KNN classifier and Logistic regression were used. But KNN classifier imputed data were very similar to the original data, giving the accuracy of 0.998.
- Imputation of X4 (roof area): Here X5, X1, X2 (imputed) and Y1 and Y2 are used for the imputation replacing the modal values which were filled earlier. For this only KNN regressor was used, since we get the imputed values almost equal to original values, as the accuracy was 0.999, so no other methods were used.
- Imputation of X7 (glazing area percentage): Here we used X1, X2, X4, X5, Y1 and Y2 to impute X7. Now X7 is having 4 values namely, so we used KNN regression and KNN Classifier. For using KNN classifier, first Level Encoder was used and then KNN classifier was used, which gave the imputed data exactly equal to the original data, i.e accuracy was 1.
- Imputation of X3 (Wall Area): Here X1, X2, X4, X5, X7, Y1 and Y2 were used to impute data using decision tree regressor as it gave accuracy 1 compared to original data. means all the imputed value were equal to the original value.
- Imputation of X6: X6 had no correlation with except with Y1 and Y2. Also X6 is a class. So X6 was imputed with LDA, QDA, KNN with optimum value of K, Decision tree Classifier, ANN with optimum value of number of hidden layers and optimum number of neurons in hidden layer. And at last Decision tree was used as it gave the highest accuracy of 0.92578 compared to the original data.
- Imputation of X8: Similar to X6, X8 laso had no correlation with except with Y1 and Y2. Also X8 is a class. So X6 was imputed with LDA, QDA, KNN with optimum value of K, Decision tree Classifier, ANN with optimum value of number of hidden layers and optimum number of neurons in hidden layer. And at last Decision tree was used as it gave the highest accuracy of 0.92578 compared to original data.
- After imputing all the data, we wanted to find which model could fit better and visualize the data. For this PCA technique was used to reduce the dimension and visualize the model by

plotting the graph. From this we got to know that data was scattered in 12 different clusters. And thus it was not a good choice to use linear regression.

- We tried fitting linear regression to the data but the R squared value was very low and this was also visible from the plot. After this we tried fitting KNN with optimum K value, Decision tree regressor and random forest to the reduced data and the accuracy for all very almost the same.
- Now these three techniques were used to fit the original data, and also 10 fold cross validation was used for good sampling. Also decision tree regressor and random forest regressor was used with optimum depth.
- Using all these, we found that Decision tree is best fitted model for Heating load with accuracy of 0.9964 and lowest RMSE. And Random forest was best fitted for cooling load with accuracy of 0.963 with lowest RMSE.

learnings from the project:

1. The above mentioned analysis can be effectively used to plan and optimise the energy consumption of a new building during its design phase itself.
2. The kind of model we use for fitting on an appropriate data is largely dependent on the size of data, its distribution and the predictors involved.
3. The scope of use of machine learning models is vast and it is surprising to find correlations between some of the predictors which do not seem to correlate intuitively.
4. Importance of machine learning techniques in developing business tactics.
5. Machine learning can be used to predict the outcomes without actually performing the task or experiment.