Final Project
Predictive Modeling

MA 5790

Group 2

Karen Fernandes
Sneha Karki

Instructor:
Dr. Quiying Sha

Date of Submission: December 14, 2023

TABLE OF CONTENTS

Abstract

The goal of the project was to explore different regression models and predict energy consumption data in a household. The project covers important steps of the data modeling life cycle including data preprocessing, exploratory data analysis, and the application of predictive models. It includes visualization, missing value handling, data transformation, outlier removal, and model training for various algorithms. The evaluation employs metrics like RMSE and R-squared. Despite some commented-out sections and typos, the script provides a comprehensive framework for predictive modeling and insights into feature engineering.

Variable Introduction and Definitions

The dataset was taken from Kaggle. It includes monitored temperature and humidity, as well as logged energy data every 10 minutes for the several appliances in a house. Additionally, weather data integrated from Chievres Airport (nearby) is part of the dataset. The key factors include the importance of WSN insights into temperature and humidity, and the impact of weather parameters such as pressure, temperature, and wind speed on predictions. The specific areas of focus for the data collection were the kitchen, laundry, and living room.

Variable Name	**Description**
Appliances	energy use in Wh
Lights	energy use of light fixtures in the house in Wh
T1	Temperature in kitchen area, in Celsius
RH_1	Humidity in kitchen area, in %
T2	Temperature in living room area, in Celsius
RH_2	Humidity in living room area, in %
T3	Temperature in laundry room area
RH_3	Humidity in laundry room area, in %
T4	Temperature in office room, in Celsius
RH_4	Humidity in office room, in %
T5	Temperature in bathroom, in Celsius
RH_5	Humidity in bathroom, in %
T6	Temperature outside the building (north side), in Celsius
RH_6	Humidity outside the building (north side), in %
T7	Temperature in ironing room , in Celsius
RH_7	Humidity in ironing room, in %
T8	Temperature in teenager room 2, in Celsius
RH_8	Humidity in teenager room 2, in %
T9	Temperature in parents room, in Celsius
RH_9	Humidity in parents room, in %
To	Temperature outside (from Chievres weather station), in Celsius
Pressure	(from Chievres weather station), in mm Hg

RH_out	Humidity outside (from Chievres weather station), in %
Wind speed	(from Chievres weather station), in m/s
Visibility	(from Chievres weather station), in km
Tdewpoint	(from Chievres weather station), Â°C
rv1	Random variable 1, nondimensional
rv2	Random variable 2, nondimensional

```
> head(app)
                 date Appliances lights    T1     RH_1   T2     RH_2    T3
1 2016-01-11 17:00:00         60     30 19.89 47.59667 19.2 44.79000 19.79
2 2016-01-11 17:10:00         60     30 19.89 46.69333 19.2 44.72250 19.79
3 2016-01-11 17:20:00         50     30 19.89 46.30000 19.2 44.62667 19.79
4 2016-01-11 17:30:00         50     40 19.89 46.06667 19.2 44.59000 19.79
5 2016-01-11 17:40:00         60     40 19.89 46.33333 19.2 44.53000 19.79
6 2016-01-11 17:50:00         50     40 19.89 46.02667 19.2 44.50000 19.79
      RH_3       T4     RH_4       T5  RH_5       T6     RH_6       T7     RH_7
1 44.73000 19.00000 45.56667 17.16667 55.20 7.026667 84.25667 17.20000 41.62667
2 44.79000 19.00000 45.99250 17.16667 55.20 6.833333 84.06333 17.20000 41.56000
3 44.93333 18.92667 45.89000 17.16667 55.09 6.560000 83.15667 17.20000 41.43333
4 45.00000 18.89000 45.72333 17.16667 55.09 6.433333 83.42333 17.13333 41.29000
5 45.00000 18.89000 45.53000 17.20000 55.09 6.366667 84.89333 17.20000 41.23000
6 44.93333 18.89000 45.73000 17.13333 55.03 6.300000 85.76667 17.13333 41.26000
    T8      RH_8       T9  RH_9    T_out Press_mm_hg RH_out Windspeed Visibility
1 18.2 48.90000 17.03333 45.53 6.600000       733.5     92  7.000000   63.00000
2 18.2 48.86333 17.06667 45.56 6.483333       733.6     92  6.666667   59.16667
3 18.2 48.73000 17.00000 45.50 6.366667       733.7     92  6.333333   55.33333
4 18.1 48.59000 17.00000 45.40 6.250000       733.8     92  6.000000   51.50000
5 18.1 48.59000 17.00000 45.40 6.133333       733.9     92  5.666667   47.66667
6 18.1 48.59000 17.00000 45.29 6.016667       734.0     92  5.333333   43.83333
  Tdewpoint      rv1      rv2
1       5.3 13.27543 13.27543
2       5.2 18.60619 18.60619
3       5.1 28.64267 28.64267
4       5.0 45.41039 45.41039
5       4.9 10.08410 10.08410
6       4.8 44.91948 44.91948
```

Figure 1: Dataset columns

The dataset consists of 19,735 samples, containing 29 columns, which provide a comprehensive view of energy consumption and environmental parameters for the chosen household. All variables are numeric, with 'Appliances' as the target variable and the remaining columns as probable predictor variables.

Preprocessing the predictors

It is imperative that the data is suited for modeling which is why we begin the process by thoroughly analyzing and understanding the nature of the dataset (EDA) so we can apply any pre-processing that would be required to ensure the models perform well and there's minimum chances of error.

Exploratory Data Analysis

1. Histogram before transformation:

   The histograms below depict the level of skewness in our dataset. These visualizations suggest that we will have to transform our variables to handle the level of skewness.

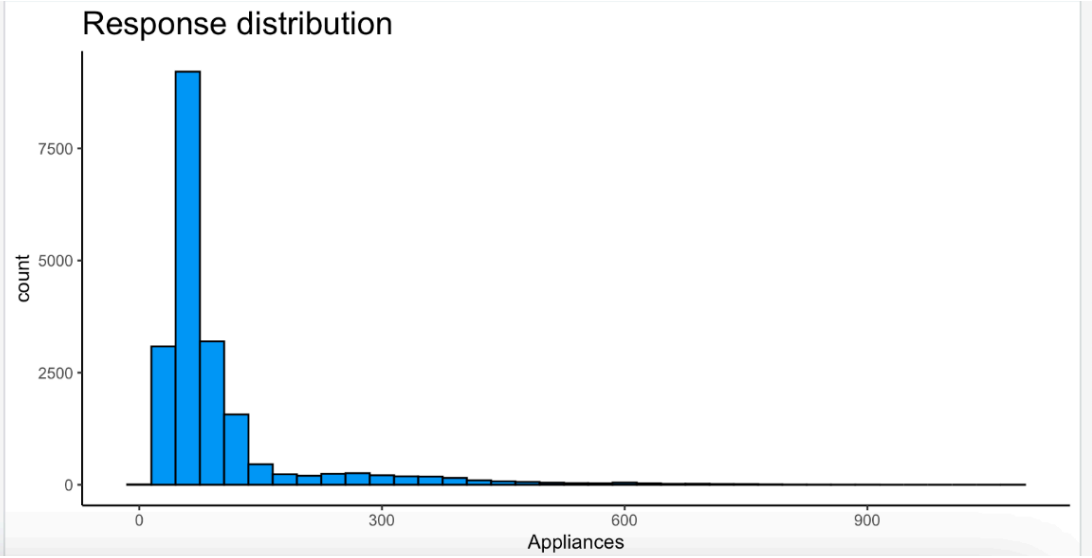
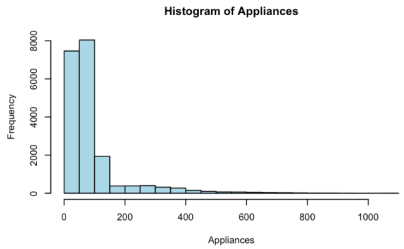
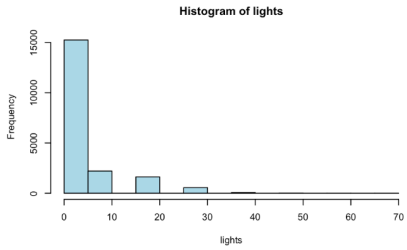
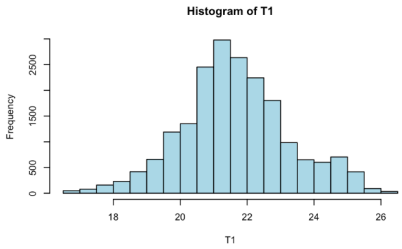
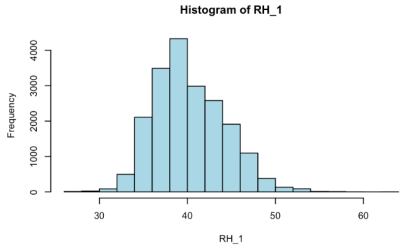


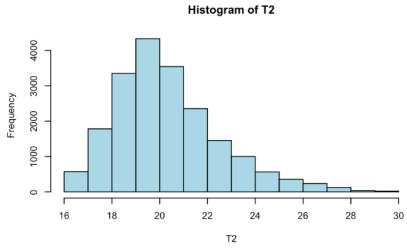
Figure 2: Histogram of response variable

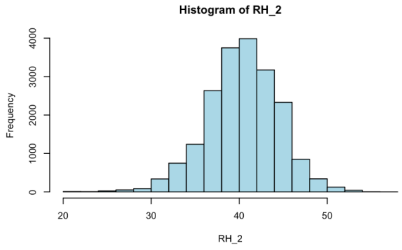

Histogram of Appliances
Frequency
0
2000
4000
6000
8000
0
200
400
600
800
1000
Appliances

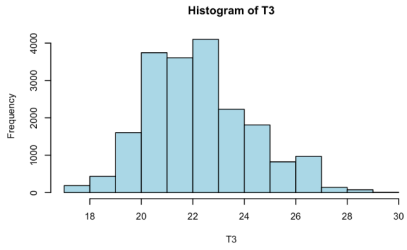

Histogram of lights
Frequency
0
5000
10000
15000
0
10
20
30
40
50
60
70
lights

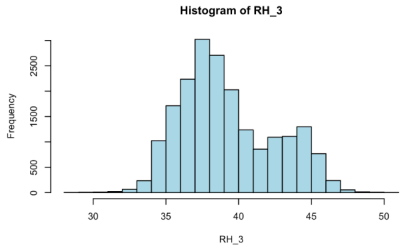

Histogram of T1
Frequency
0
500
1500
2500
18
20
22
24
26
T1

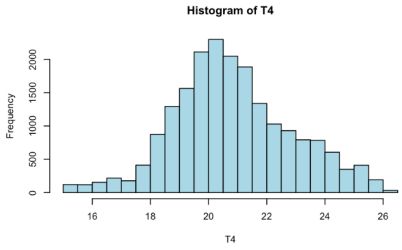

Histogram of RH_1
Frequency
0
1000
2000
3000
4000
30
40
50
60
RH_1

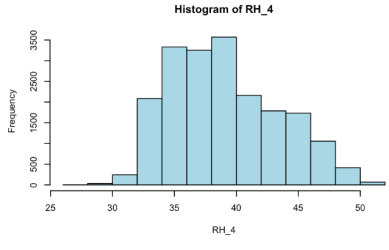

Histogram of T2
Frequency
0
1000
2000
3000
4000
16
18
20
22
24
26
28
30
T2

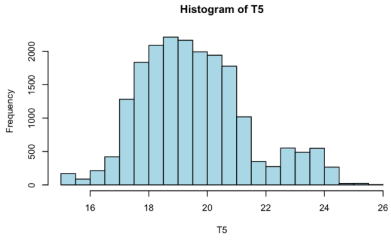

Histogram of RH_2
Frequency
0
1000
2000
3000
4000
20
30
40
50
RH_2

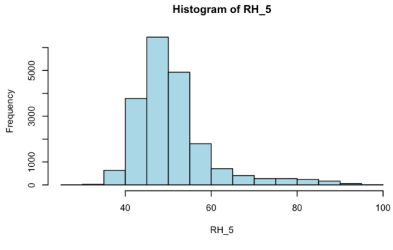

Histogram of T3
Frequency
0
1000
2000
3000
4000
18
20
22
24
26
28
30
T3

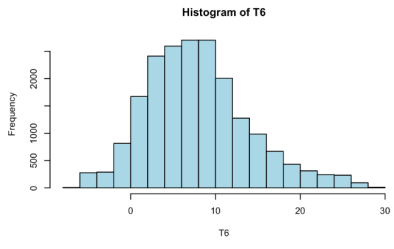

Histogram of RH_3
Frequency
0
500
1500
2500
30
35
40
45
50
RH_3

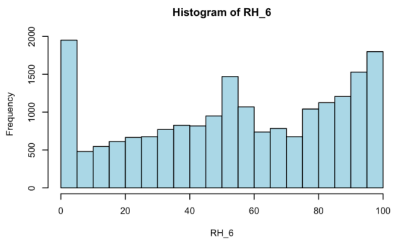

Histogram of T4
Frequency
0
500
1000
1500
2000
16
18
20
22
24
26
T4

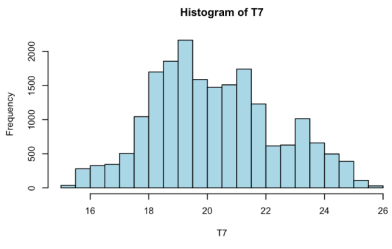

Histogram of RH_4
Frequency
0
500
1500
2500
3500
25
30
35
40
45
50
RH_4

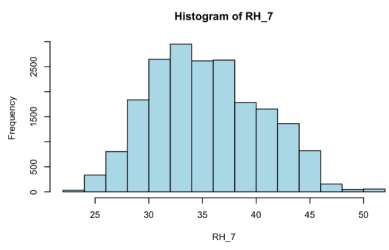

Histogram of T5
Frequency
0
500
1000
1500
2000
16
18
20
22
24
26
T5

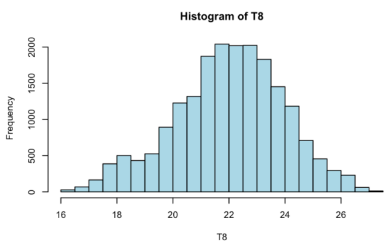

Histogram of RH_5
Frequency
0
1000
3000
5000
40
60
80
100
RH_5

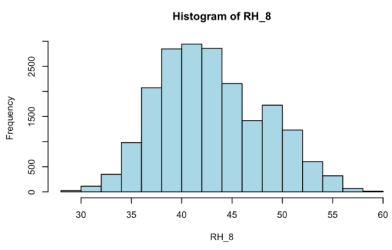

Histogram of T6
Frequency
0
500
1000
2000
0
10
20
30
T6


Histogram of RH_6
Frequency
0
500
1000
1500
2000
0
20
40
60
80
100
RH_6


Histogram of T7
Frequency
0
500
1000
1500
2000
16
18
20
22
24
26
T7


Histogram of RH_7
Frequency
0
500
1500
2500
25
30
35
40
45
50
RH_7


Histogram of T8
Frequency
0
500
1000
1500
2000
16
18
20
22
24
26
T8


Histogram of RH_8
Frequency
0
500
1500
2500
30
35
40
45
50
55
60
RH_8

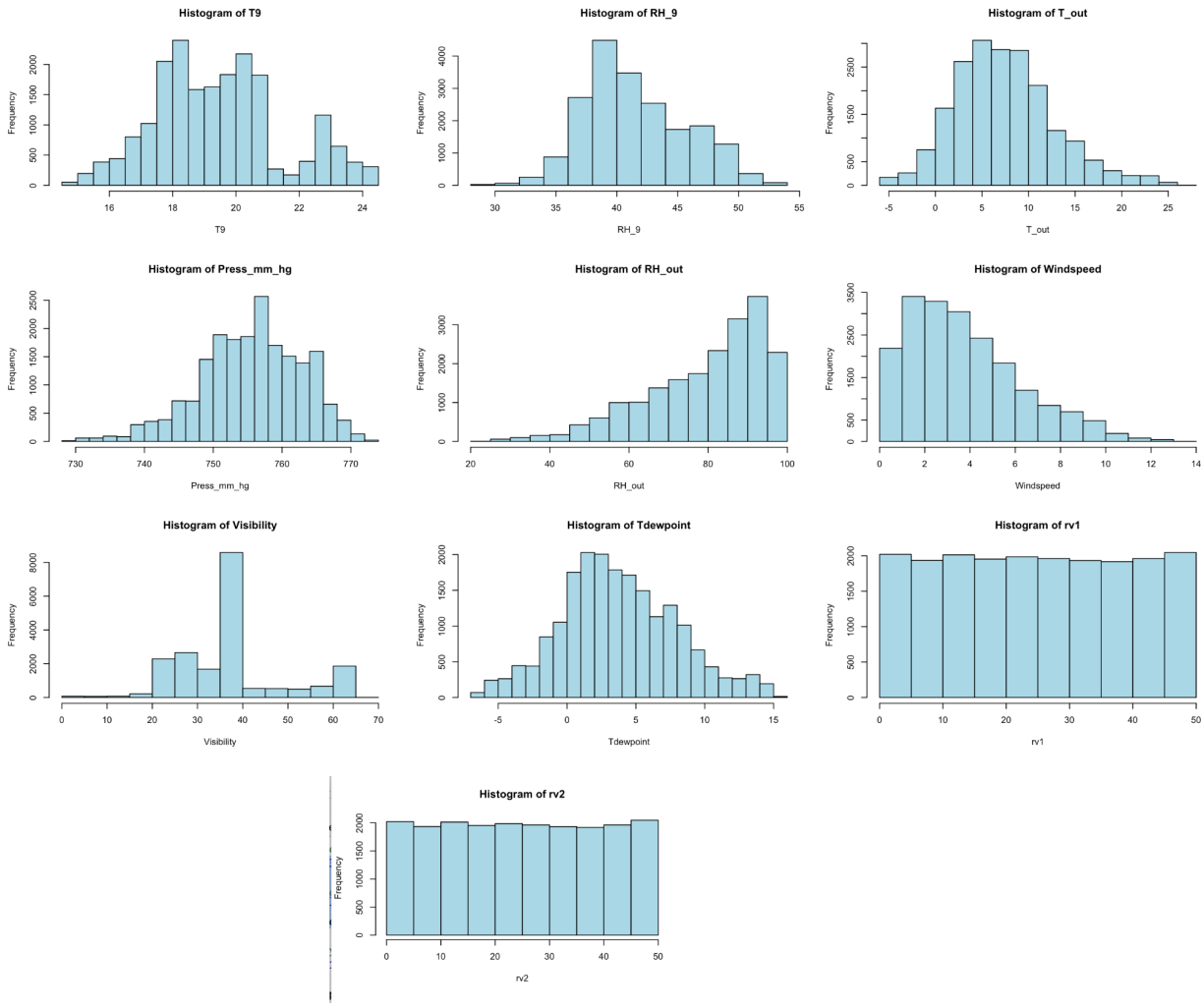


Figure 3: Histograms before transformation

The figure shows how skewed our variables were initially.

```
> print(skewness_summary)
              Predictor     Skewness
Appliances   Appliances  3.385852456
lights           lights  2.194821357
T1                   T1  0.120898737
RH_1               RH_1  0.465702955
T2                   T2  0.889522614
RH_2               RH_2 -0.268206397
T3                   T3  0.450708813
RH_3               RH_3  0.467518002
T4                   T4  0.170358157
RH_4               RH_4  0.444546754
T5                   T5  0.558134844
RH_5               RH_5  1.866536246
T6                   T6  0.597380519
RH_6               RH_6 -0.241924420
T7                   T7  0.254683697
RH_7               RH_7  0.242104021
T8                   T8 -0.256112198
RH_8               RH_8  0.307988791
T9                   T9  0.382653063
RH_9               RH_9  0.368880961
T_out             T_out  0.534191416
Press_mm_hg Press_mm_hg -0.420377607
RH_out           RH_out -0.922856943
Windspeed     Windspeed  0.859851376
Visibility   Visibility  0.441487342
Tdewpoint     Tdewpoint  0.239337690
rv1                 rv1  0.004943844
rv2                 rv2  0.004943844
>
```

Figure 4: Histograms after transformation

2. Missing Values

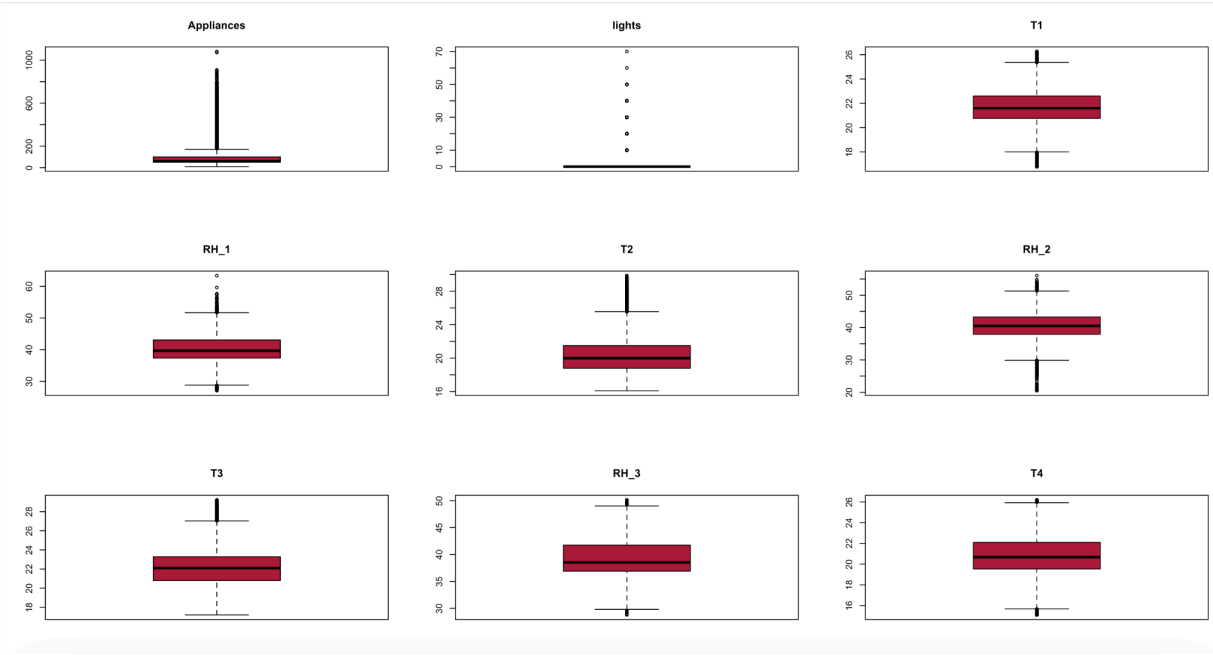
   Our dataset had no missing values as seen in the figure below.

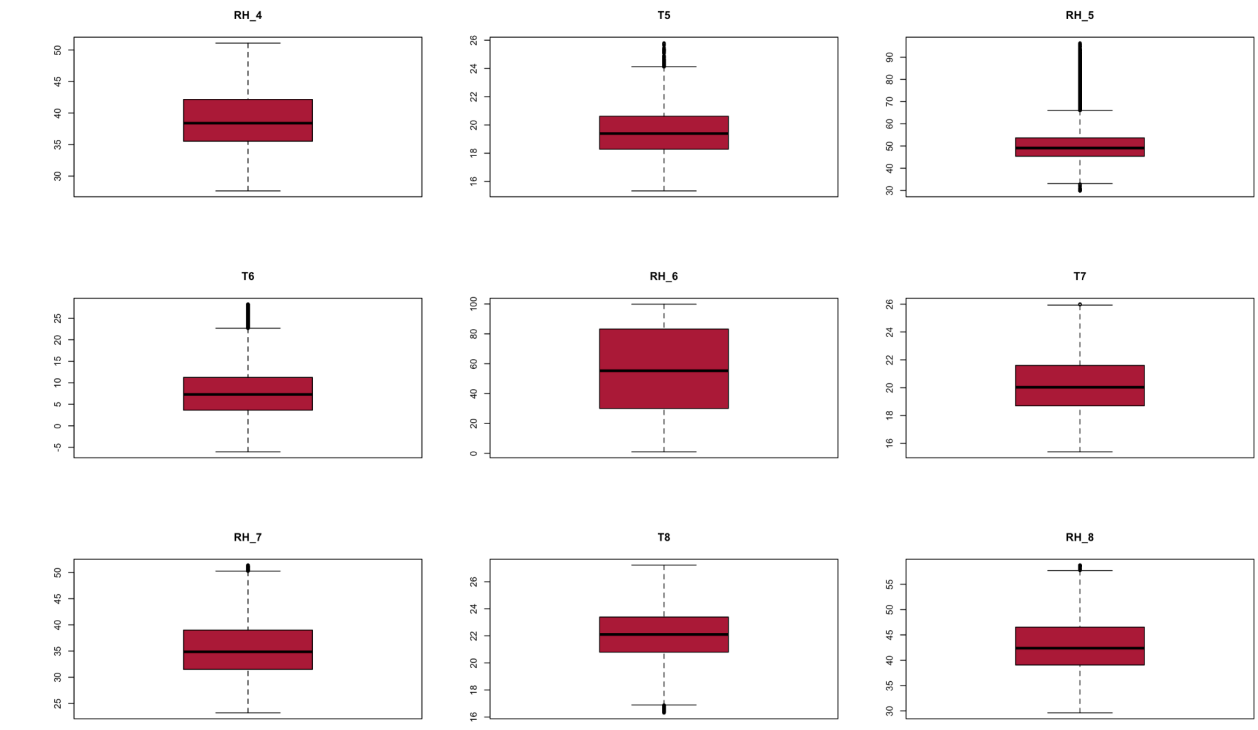
```
     date  Appliances      lights          T1        RH_1          T2        RH_2          T3
        0           0           0           0           0           0           0           0
     RH_3          T4        RH_4          T5        RH_5          T6        RH_6          T7
        0           0           0           0           0           0           0           0
     RH_7          T8        RH_8          T9        RH_9       T_out Press_mm_hg      RH_out
        0           0           0           0           0           0           0           0
Windspeed  Visibility   Tdewpoint         rv1         rv2
        0           0           0           0           0
```

Figure 5: Missing values

3. Outliers:

   There are some outliers in our dataset as we can see from the boxplots below.


Appliances
0
200
600
1000
lights
0
10
30
50
70
T1
18
20
22
24
26
RH_1
30
40
50
60
T2
16
20
24
28
RH_2
20
30
40
50
T3
18
20
22
24
26
28
RH_3
30
35
40
45
50
T4
16
18
20
22
24
26


RH_4
30
35
40
45
50
T5
16
18
20
22
24
26
RH_5
30
40
50
60
70
80
90
T6
-5
0
5
10
15
20
25
RH_6
0
20
40
60
80
100
T7
16
18
20
22
24
26
RH_7
25
30
35
40
45
50
T8
16
18
20
22
24
26
RH_8
30
35
40
45
50
55

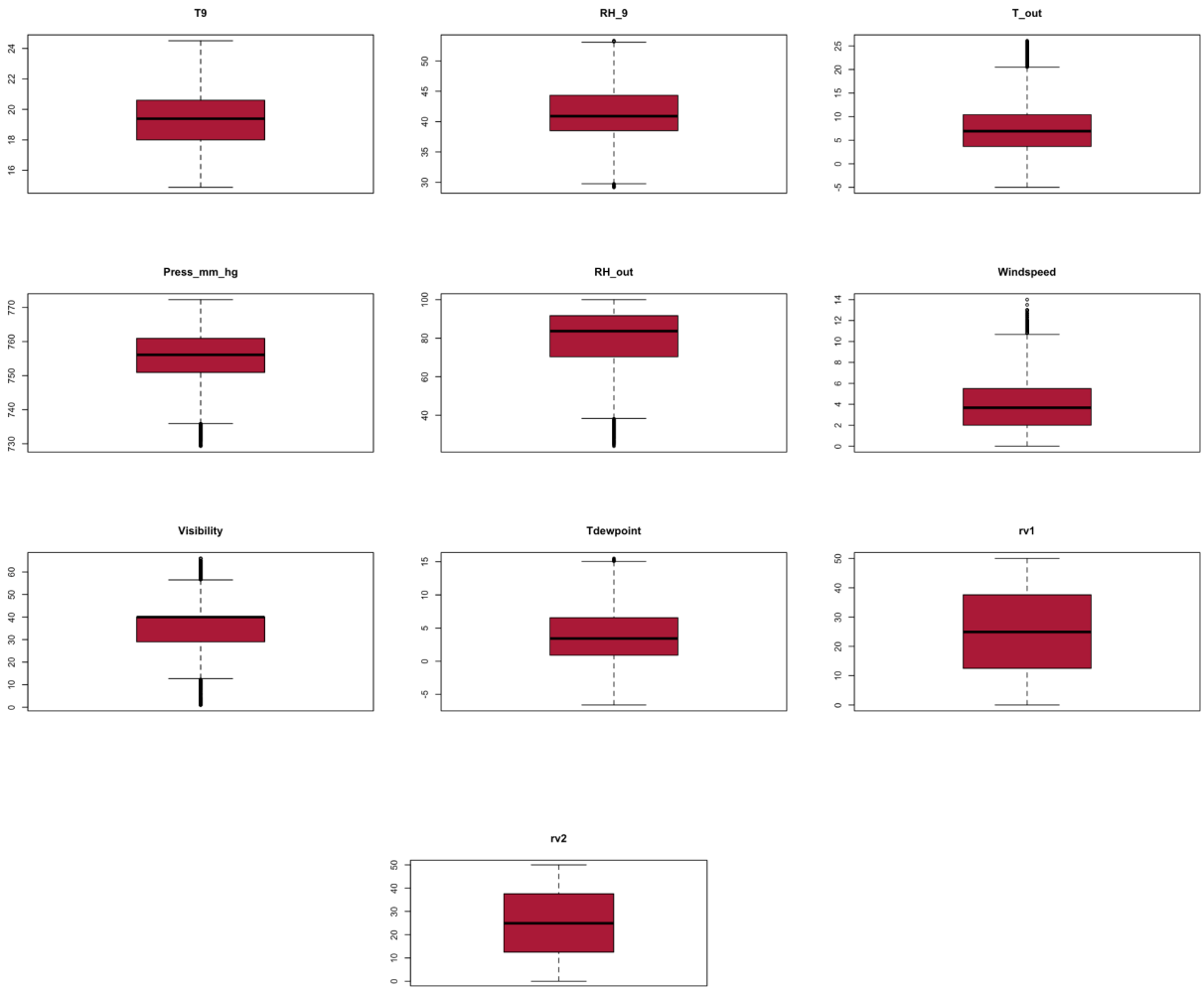


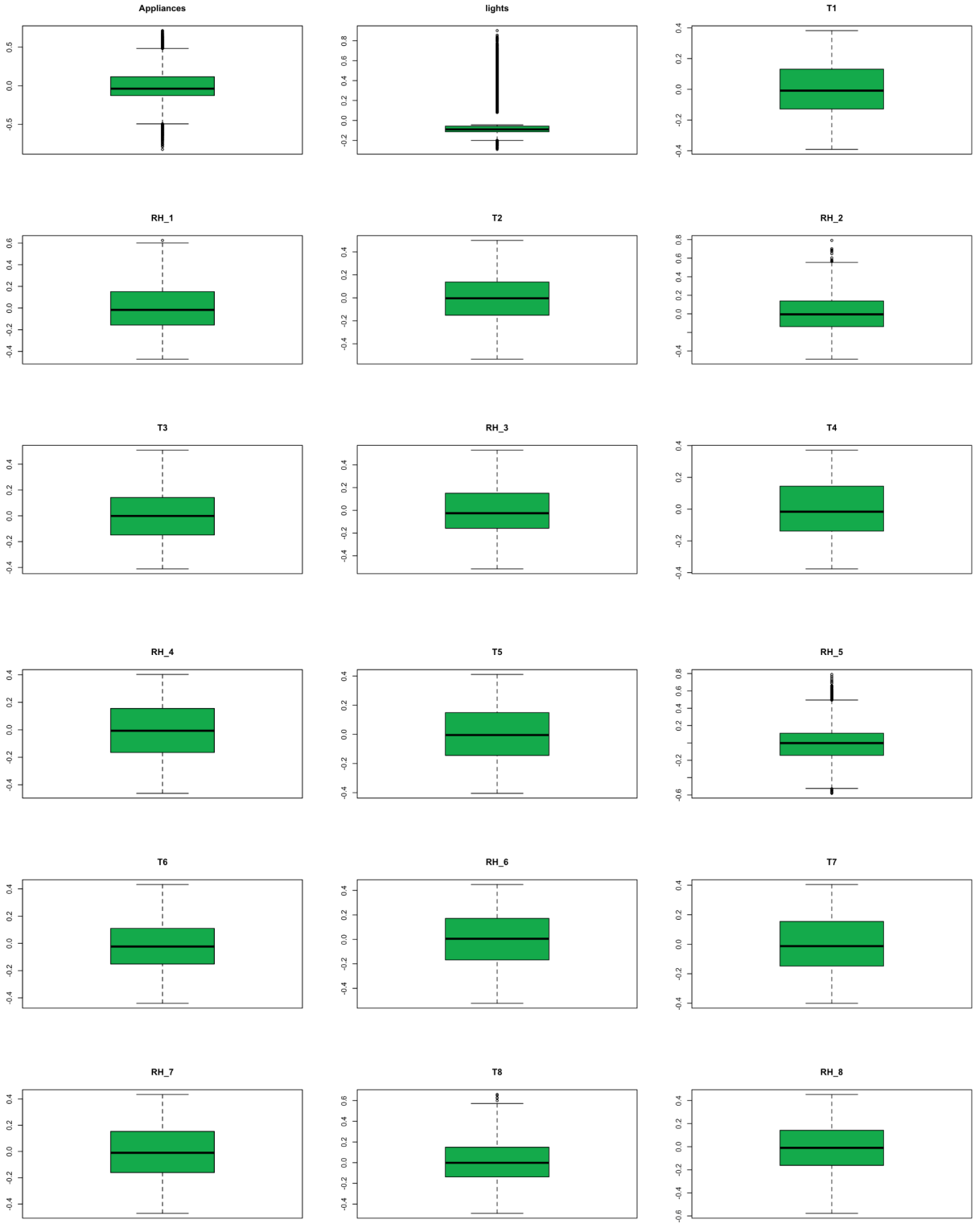
Figure 6: Box-plots before transformation

Data Preprocessing:

We used Centering, Scaling, spatial sign transformation and BoxCox transformations to address these issues. Here are the results visually how the data changed after transformation.

Boxplots after transformation:

To handle the outliers we performed spatial sign transformation and following the transformation, the outliers were removed as depicted in the plots below.


Appliances
lights
T1
RH_1
T2
RH_2
T3
RH_3
T4
RH_4
T5
RH_5
T6
RH_6
T7
RH_7
T8
RH_8

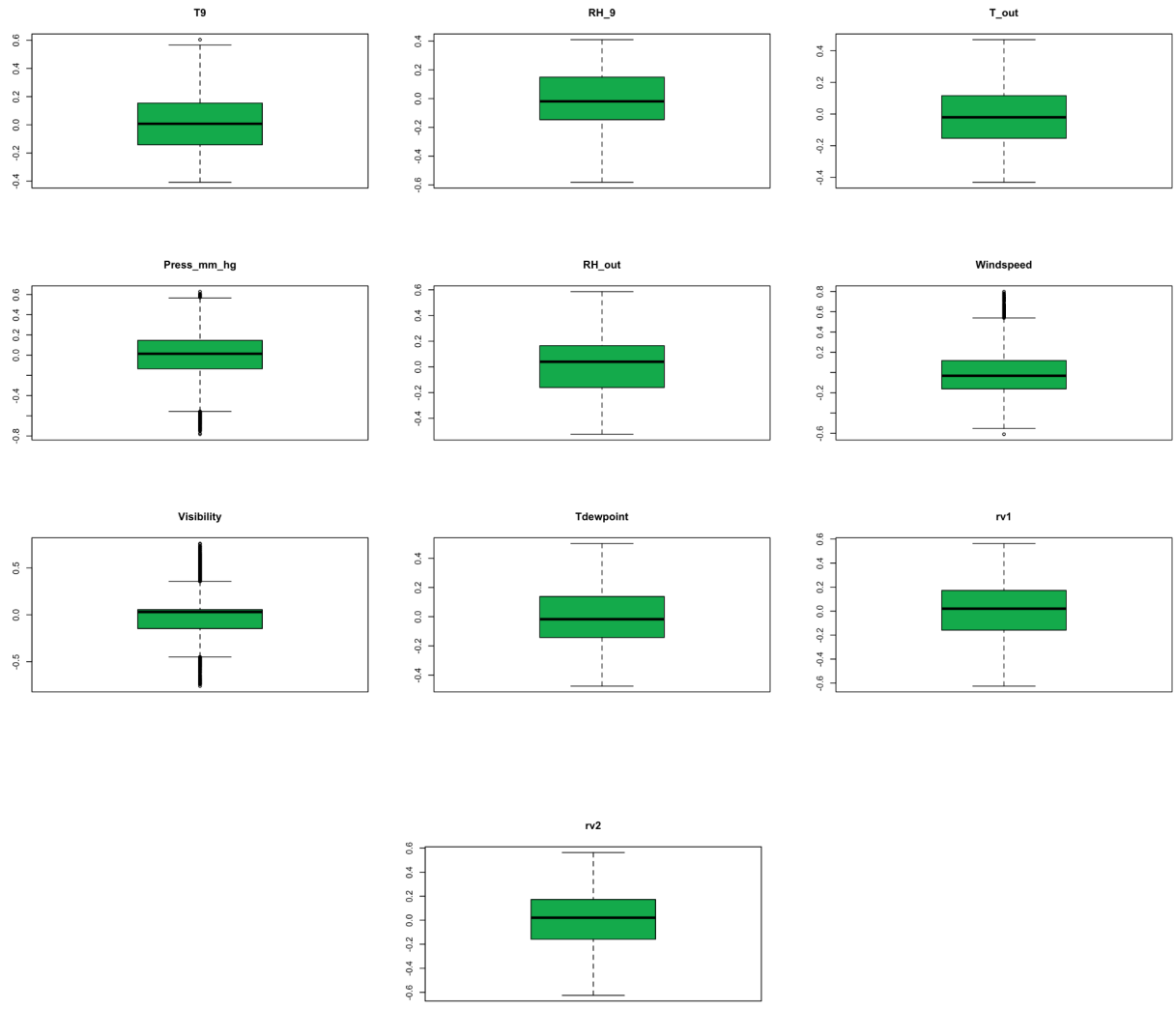
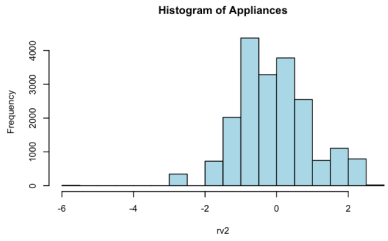
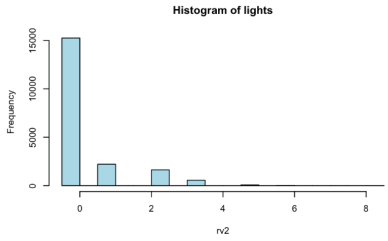


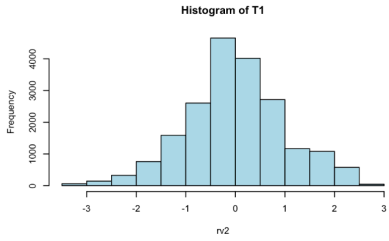
Figure 7: Boxplots after transformation

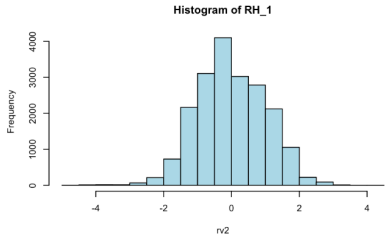
Histograms after transformation:

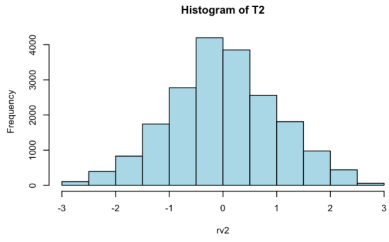
After undergoing the transformation, the varying degree of skewness was controlled as we can see in the histograms and the skewness summary shown below.

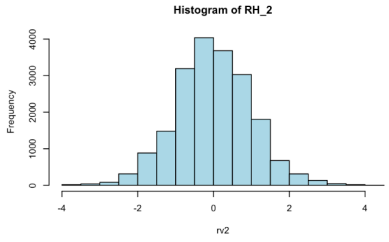

Histogram of Appliances
Frequency
rv2

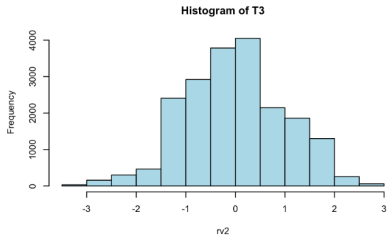

Histogram of lights
Frequency
rv2

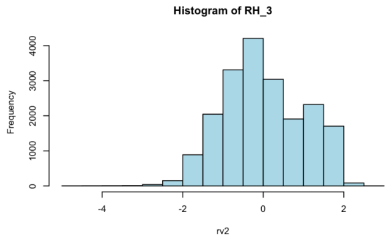

Histogram of T1
Frequency
rv2

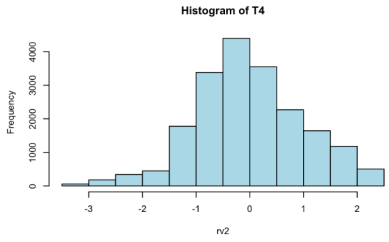

Histogram of RH_1
Frequency
rv2

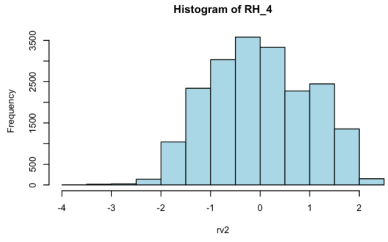

Histogram of T2
Frequency
rv2

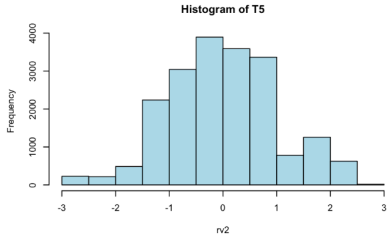

Histogram of RH_2
Frequency
rv2

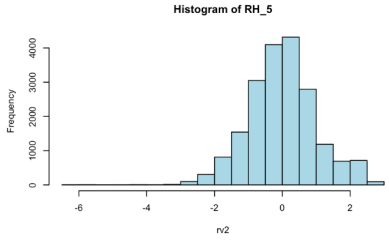

Histogram of T3
Frequency
rv2

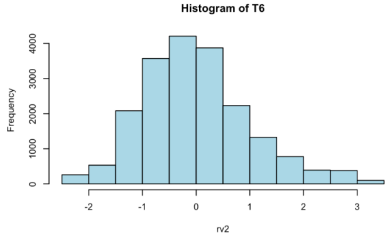

Histogram of RH_3
Frequency
rv2

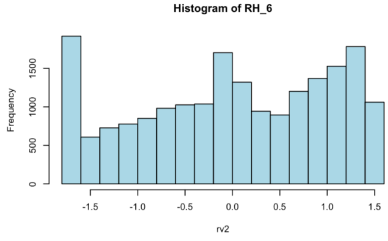

Histogram of T4
Frequency
rv2

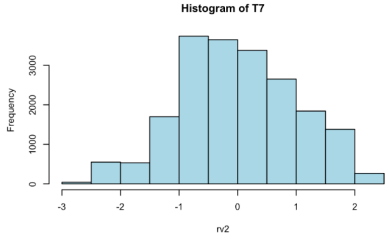

Histogram of RH_4
Frequency
rv2

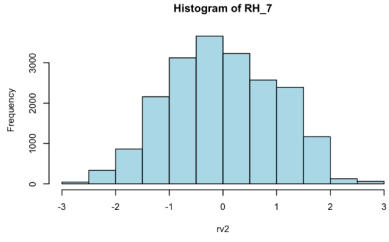

Histogram of T5
Frequency
rv2

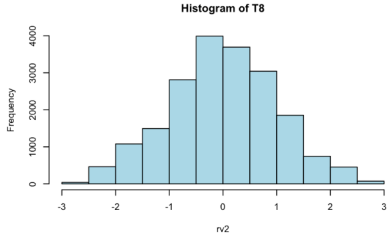

Histogram of RH_5
Frequency
rv2

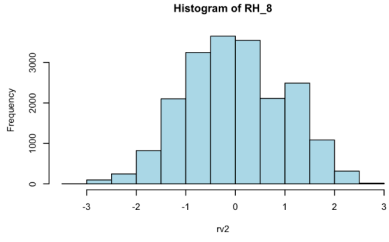

Histogram of T6
Frequency
rv2


Histogram of RH_6
Frequency
rv2


Histogram of T7
Frequency
rv2


Histogram of RH_7
Frequency
rv2


Histogram of T8
Frequency
rv2


Histogram of RH_8
Frequency
rv2

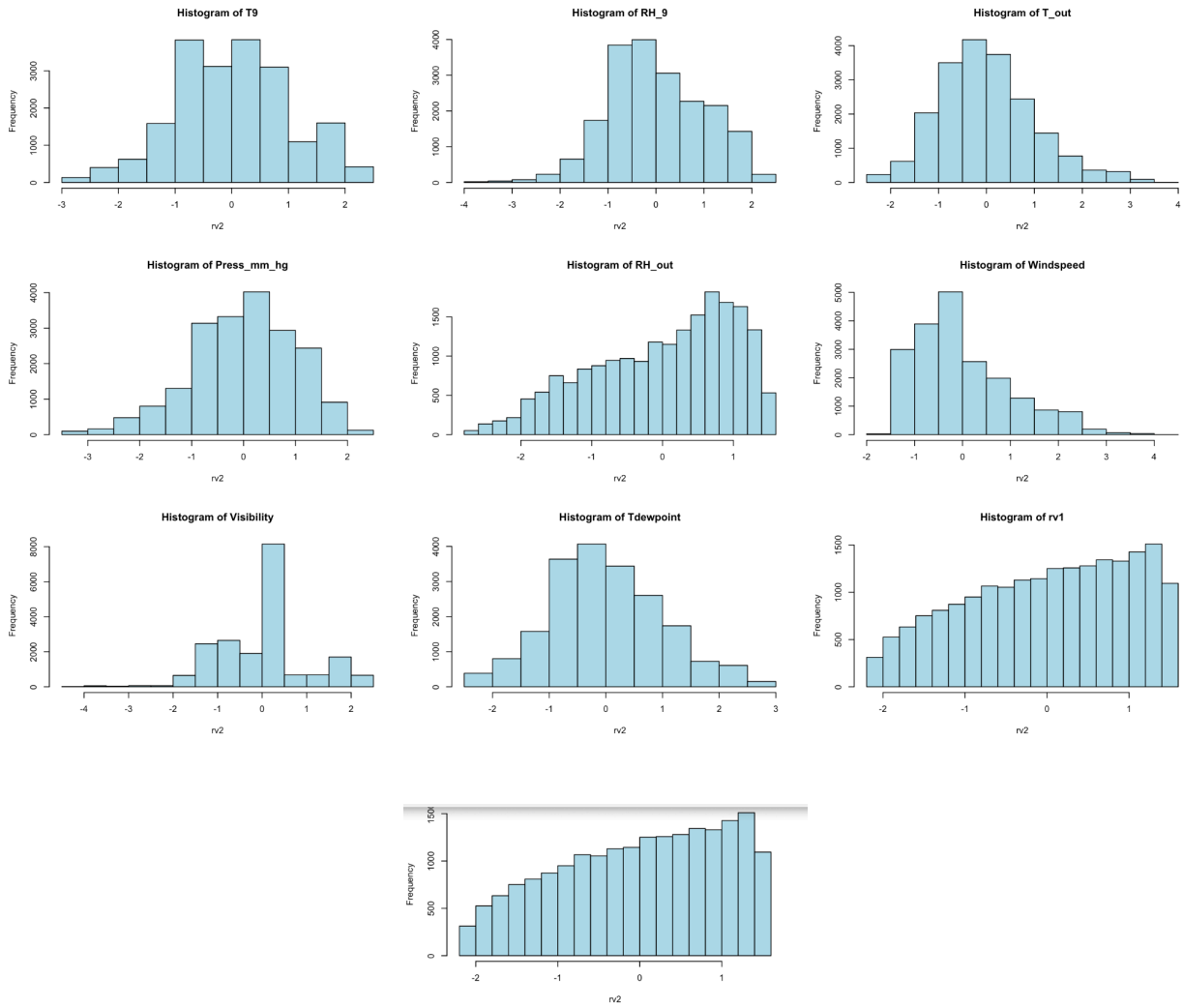


Figure 8: Histograms after transformation

Skewness after transformation:

```
> print(skewness_summ)
                Predictor     Skewness
Appliances    Appliances  0.025282351
lights            lights  2.194821357
T1                    T1 -0.000870929
RH_1                RH_1  0.003838621
T2                    T2  0.028761091
RH_2                RH_2  0.043452508
T3                    T3  0.012707287
RH_3                RH_3  0.033460336
T4                    T4 -0.007811260
RH_4                RH_4  0.030442115
T5                    T5  0.016013324
RH_5                RH_5 -0.035225350
T6                    T6  0.597380519
RH_6                RH_6 -0.241924420
T7                    T7  0.017387363
RH_7                RH_7 -0.000441908
T8                    T8 -0.018200554
RH_8                RH_8  0.043245077
T9                    T9  0.017691531
RH_9                RH_9  0.013902952
T_out              T_out  0.534191416
Press_mm_hg Press_mm_hg -0.392568539
RH_out            RH_out -0.531792274
Windspeed     Windspeed  0.859851376
Visibility   Visibility  0.086623440
Tdewpoint     Tdewpoint  0.239337690
rv1                 rv1 -0.294983560
rv2                 rv2 -0.294983560
```

Figure 9: Skewness values

Principal Component Analysis:

We performed PCA to reduce the dimensionality of the data. However, It reduced the size to 13 columns indicating information loss. We avoided using the PCA transformed dataset to prevent it from affecting the model's performance and slowing down predictions.

Here's the dataset after retaining the 13 columns from PCA with a cut-off of 95%.

```
> head(data_pca)
          PC1      PC2       PC3        PC4       PC5        PC6         PC7         PC8       PC9
[1,] 2.293483 3.349918 1.3671334  0.2196486 1.0516631 -1.1350059 -0.51561353 -0.02890193 0.9490499
[2,] 2.368183 3.383802 1.2203031 -0.1042481 1.0509112 -1.0077929 -0.42689483  0.02773568 0.9923292
[3,] 2.432885 3.363727 0.9785900 -0.6822392 1.0362416 -0.8562634 -0.32390122  0.06913188 1.0427960
[4,] 2.214713 2.990802 0.5754074 -1.3788200 0.8924482 -0.6173445 -0.19137615  0.09009259 0.9801571
[5,] 2.255870 3.043035 1.2132723  0.4848588 0.9045846 -0.5074631 -0.04664128  0.13493457 1.0467998
[6,] 2.273130 2.988376 0.5018933 -1.3432559 0.8563498 -0.3346676  0.04447549  0.16778487 1.0789614
           PC10       PC11      PC12         PC13
[1,] -0.08130520 -0.2473050 0.7079391  0.010534032
[2,] -0.04648723 -0.2865835 0.6685639 -0.007895209
[3,] -0.04292158 -0.3071059 0.6243516 -0.036652329
[4,] -0.03461100 -0.2809094 0.5469863 -0.063780437
[5,] -0.04393986 -0.2944100 0.5522236 -0.052950551
[6,] -0.03252994 -0.2944959 0.5032433 -0.065781999
> dim(data_pca)
[1] 19735    13
```

Figure 10: PC's after PCA

Near Zero variance:

Our dataset did not have any near-zero variance as can be seen in the figure below.

```
> print(near_zero_var)
             freqRatio percentUnique zeroVar   nzv
Appliances           1           100   FALSE FALSE
lights               1           100   FALSE FALSE
T1                   1           100   FALSE FALSE
RH_1                 1           100   FALSE FALSE
T2                   1           100   FALSE FALSE
RH_2                 1           100   FALSE FALSE
T3                   1           100   FALSE FALSE
RH_3                 1           100   FALSE FALSE
T4                   1           100   FALSE FALSE
RH_4                 1           100   FALSE FALSE
T5                   1           100   FALSE FALSE
RH_5                 1           100   FALSE FALSE
T6                   1           100   FALSE FALSE
RH_6                 1           100   FALSE FALSE
T7                   1           100   FALSE FALSE
RH_7                 1           100   FALSE FALSE
T8                   1           100   FALSE FALSE
RH_8                 1           100   FALSE FALSE
T9                   1           100   FALSE FALSE
RH_9                 1           100   FALSE FALSE
T_out                1           100   FALSE FALSE
Press_mm_hg          1           100   FALSE FALSE
RH_out               1           100   FALSE FALSE
Windspeed            1           100   FALSE FALSE
Visibility           1           100   FALSE FALSE
Tdewpoint            1           100   FALSE FALSE
rv1                  1           100   FALSE FALSE
rv2                  1           100   FALSE FALSE
```

Figure 11: Near- zero variance of variables

Data Splitting

The dataset was effectively split into training and testing sets using a random seed for reproducibility in a 80:20 ratio. The training set, containing 80% of the data, was derived using the 'createDataPartition' function, leaving the remaining 20% for the testing set.

Model Building

LINEAR REGRESSION MODELS

- **Ridge Model**

  The summary results and performance measure for Ridge model can be seen below.

```
Ridge Regression

15791 samples
   26 predictor

Pre-processing: centered (26), scaled (26)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12631, 12634, 12633, 12632, 12634
Resampling results across tuning parameters:

  lambda  RMSE       Rsquared   MAE
  0.1     0.1772957  0.2237362  0.1300600
  0.2     0.1785802  0.2121236  0.1312290
  0.3     0.1794798  0.2037105  0.1321036
  0.4     0.1802254  0.1968254  0.1328598
  0.5     0.1808997  0.1908836  0.1335576
  0.6     0.1815397  0.1856215  0.1342268
  0.7     0.1821642  0.1808924  0.1348793
  0.8     0.1827836  0.1766015  0.1355364
  0.9     0.1834039  0.1726808  0.1362055
  1.0     0.1840284  0.1690786  0.1368783

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.1.
```

Figure 12: Test values for Ridge model

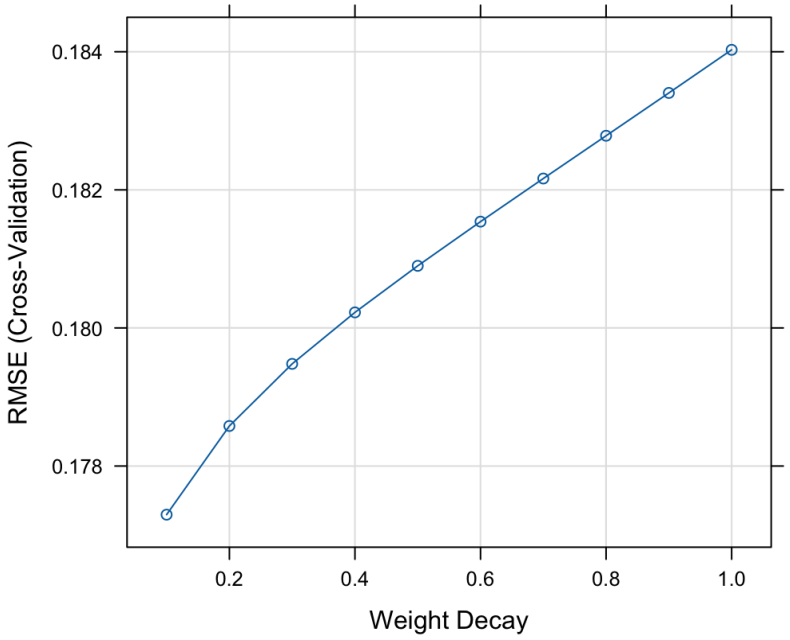


Figure 13: Tuning parameters plot for Ridge Model

RMSE	Rsquared	MAE
0.1572870	0.1458906	0.1291130

Figure 14: Test values for Ridge model

- **Lasso Model**

The summary results and performance measure for LASSO model can be seen below.

```
The lasso

15791 samples
   26 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12631, 12632, 12634, 12633, 12634
Resampling results across tuning parameters:

  fraction  RMSE       Rsquared   MAE
  0.1       0.1890665  0.1449402  0.1429420
  0.5       0.1769223  0.2295801  0.1298421
  0.9       0.1751209  0.2418236  0.1286441

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.9.
```

Figure 15: Test values for Lasso model

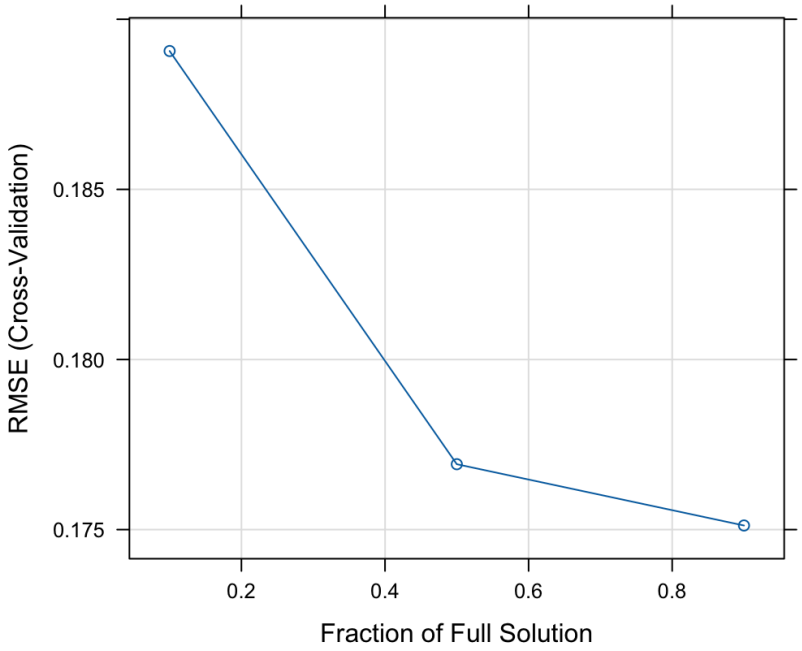


Figure 16: Tuning parameters plot for Lasso Model

RMSE	Rsquared	MAE
0.1655112	0.1024274	0.1348768

Figure 17: Test values for Lasso

- **Elastic Net Model**

The summary results and performance measure for Elastic Net model can be seen below.

```
Elasticnet

15791 samples
   26 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12634, 12633, 12632, 12633, 12632
Resampling results across tuning parameters:

  lambda  fraction  RMSE       Rsquared    MAE
  0e+00   0.050     0.1936332  0.10340290  0.1478820
  0e+00   0.525     0.1766506  0.23150652  0.1295936
  0e+00   1.000     0.1751387  0.24186807  0.1287108
  1e-04   0.050     0.1936556  0.10315219  0.1479072
  1e-04   0.525     0.1766728  0.23136332  0.1296130
  1e-04   1.000     0.1751383  0.24187017  0.1287068
  1e-01   0.050     0.1964771  0.08857318  0.1514681
  1e-01   0.525     0.1811447  0.19700208  0.1338585
  1e-01   1.000     0.1773418  0.22375364  0.1300751

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 1 and lambda = 1e-04.
```

Figure 18: Test values for Elastic net model

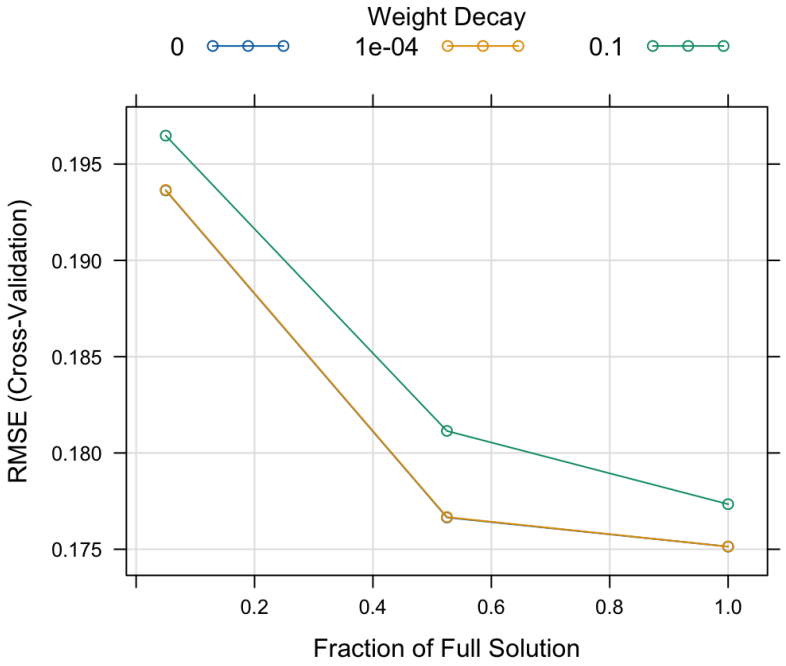


Figure 19: Tuning parameters plot for Elastic Net

RMSE	Rsquared	MAE
0.1661698	0.1001612	0.1353435

Figure 20: Test values for Elastic net

- Penalized model

The summary results and performance measure for the Penalized model can be seen below.

```
glmnet

15791 samples
   26 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12633, 12634, 12632, 12633, 12632
Resampling results across tuning parameters:

  alpha  lambda        RMSE       Rsquared   MAE
  0.10   0.0001002062  0.1751149  0.2417834  0.1286746
  0.10   0.0010020617  0.1752196  0.2409921  0.1286359
  0.10   0.0100206168  0.1770942  0.2286909  0.1300151
  0.55   0.0001002062  0.1751208  0.2417391  0.1286683
  0.55   0.0010020617  0.1754012  0.2396666  0.1286788
  0.55   0.0100206168  0.1807631  0.2032931  0.1336127
  1.00   0.0001002062  0.1751286  0.2416754  0.1286595
  1.00   0.0010020617  0.1755940  0.2383814  0.1287922
  1.00   0.0100206168  0.1849264  0.1683796  0.1382515

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.1 and lambda = 0.0001002062.
```

Figure 21: Test values for Penalized model

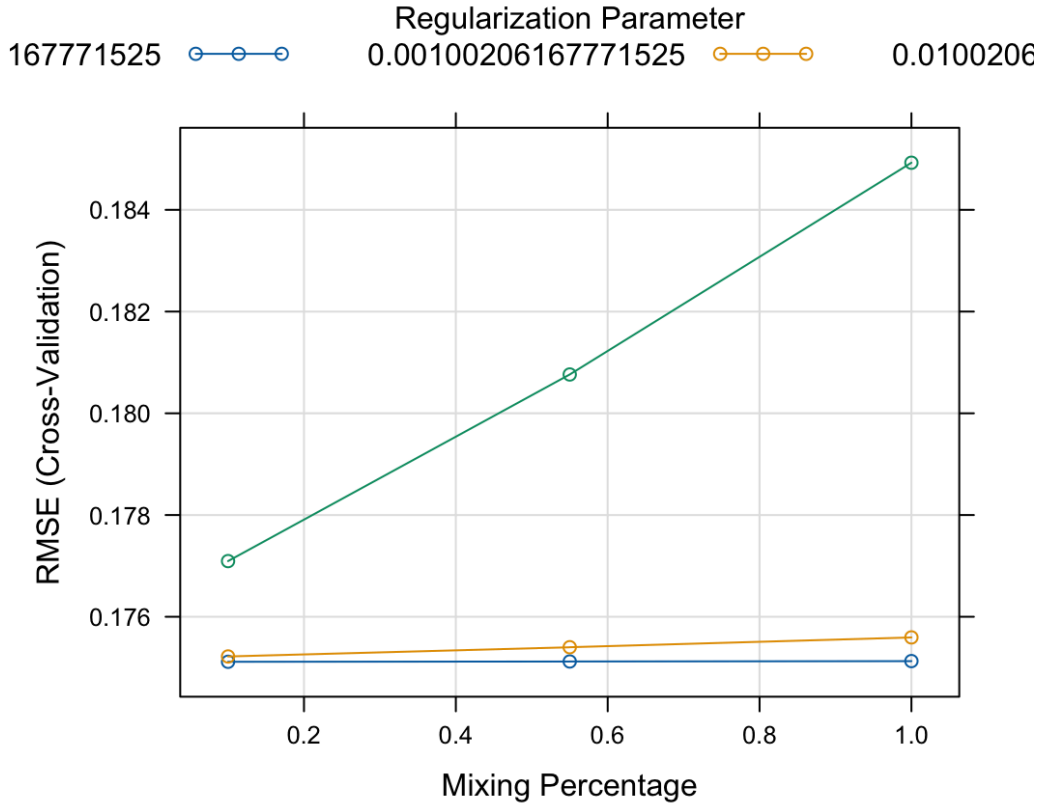


Figure 22: Tuning parameters plot for Penalized Model

RMSE	Rsquared	MAE
0.1659813	0.1007334	0.1352123

Figure 23: Test values for Penalized modes

- **OLS model**

  The summary results and performance measure for OLS model can be seen below.

```
Linear Regression

15791 samples
   26 predictor

Pre-processing: centered (26), scaled (26)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12634, 12633, 12634, 12631, 12632
Resampling results:

  RMSE       Rsquared   MAE
  0.1751837  0.2414558  0.128766

Tuning parameter 'intercept' was held constant at a value of TRUE
```

Figure : OLS values for train

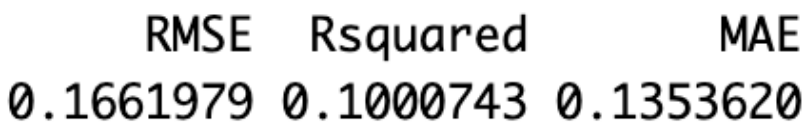


Figure : OLS values for test

NON-LINEAR MODELS

- **Neural Networks**

The summary results and performance measure for Neural Network model can be seen below.

```
Neural Network

15791 samples
   26 predictor

Pre-processing: centered (26), scaled (26), spatial sign transformation (26)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12632, 12634, 12632, 12633, 12633
Resampling results:

  RMSE       Rsquared   MAE
  0.1841265  0.2443934  0.1425017

Tuning parameter 'size' was held constant at a value of 3
Tuning parameter 'decay' was held constant at a value of 0.1
```

Figure 24: Test values for NNet model

Testing performance:

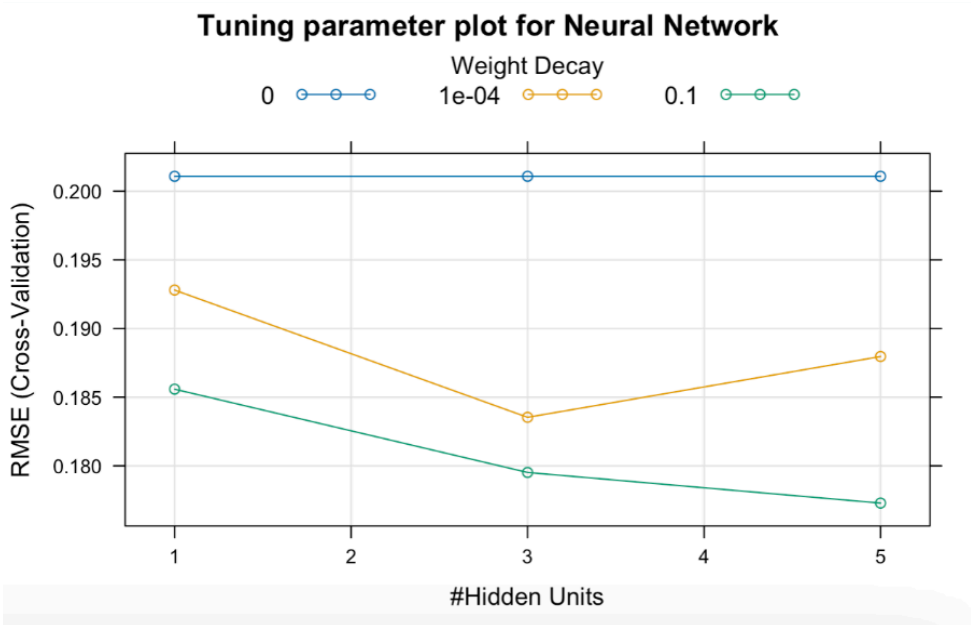


Figure 25: Tuning parameters plot for Neural Network

RMSE	Rsquared	MAE
0.18414838	0.05650354	0.14995284

Figure 26: Test values for NNet

- **Logistic Regression Model**

  The summary results and performance measure for Logistic Regression model can be seen below.

```
Generalized Linear Model

15791 samples
   26 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12633, 12632, 12633, 12633, 12633
Resampling results:

  RMSE       Rsquared   MAE
  0.1750688  0.2422752  0.1286807
```

Figure 27: Test values for Logistic model

Testing performance:

```
> lr_perf
     RMSE  Rsquared       MAE
0.1661979 0.1000743 0.1353620
```

Figure 28: Test values for MARS

- **PCR model**

```
Principal Component Analysis

15791 samples
   26 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12633, 12631, 12635, 12634, 12631
Resampling results across tuning parameters:

  ncomp  RMSE       Rsquared    MAE
  1      0.1975886  0.03481951  0.1523686
  2      0.1974536  0.03618013  0.1518582
  3      0.1973614  0.03697443  0.1517314

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 3.
```

Figure 29: Test values for PCR model

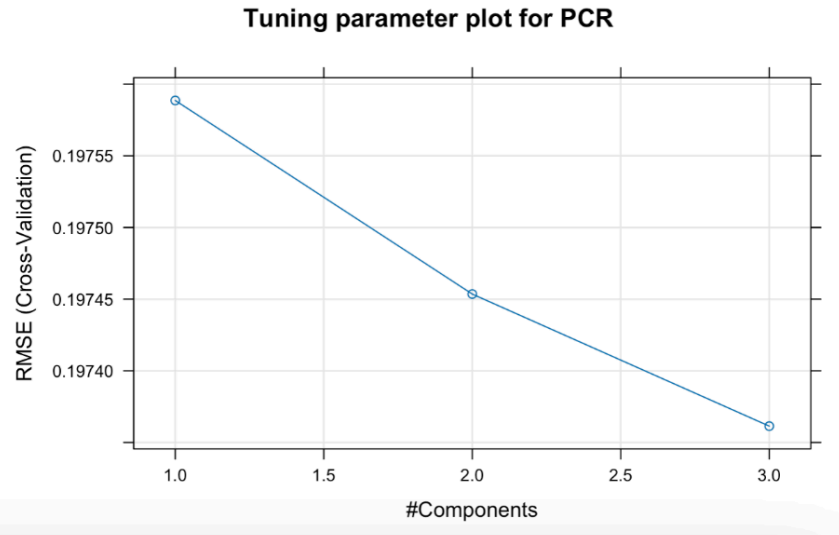


Figure 30: Tuning parameters plot for PCR

Testing performance:

RMSE	Rsquared	MAE
0.1387826	0.6585602	0.1162329

Figure 31: Test values for PCR

- **PLSDA model**

  The summary results and performance measure for PLSDA model can be seen below.

```
Partial Least Squares

15791 samples
   26 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12634, 12632, 12633, 12634, 12631
Resampling results across tuning parameters:

  ncomp  RMSE       Rsquared    MAE
  1      0.1929729  0.07905924  0.1465334
  2      0.1856388  0.14780687  0.1382420
  3      0.1811763  0.18842274  0.1342491

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 3.
```

Figure 32: Test values for PLSDA model

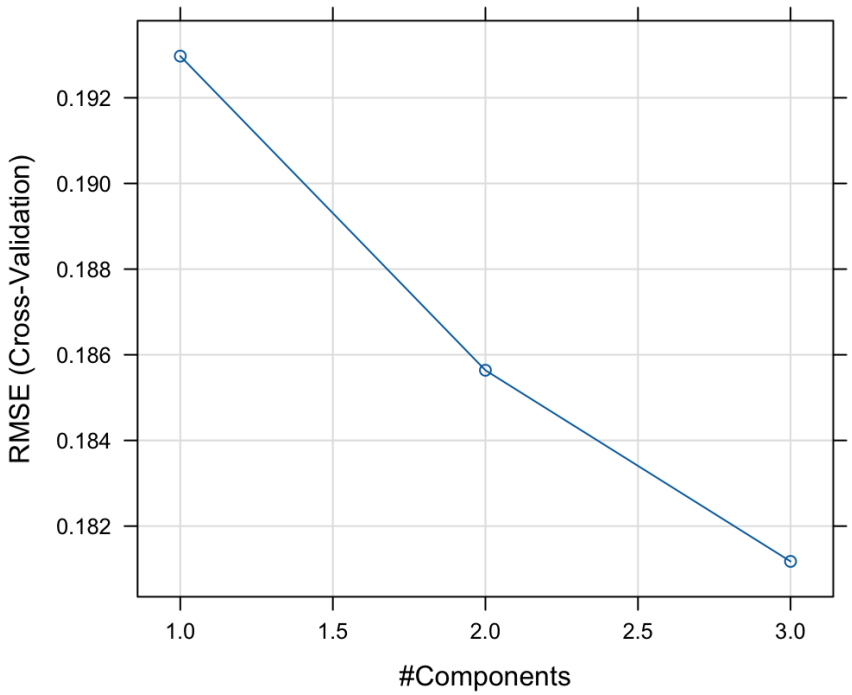


Figure 33: Tuning parameters plot for PLS

RMSE	Rsquared	MAE
0.1556668	0.1581440	0.1284719

Figure 34: Test values for PLSDA

- **K Nearest Neighbors (KNN)**

  The summary results and performance measure for KNN model can be seen below.

```
k-Nearest Neighbors

15791 samples
   26 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12633, 12633, 12633, 12634, 12631
Resampling results across tuning parameters:

  k  RMSE       Rsquared   MAE
  5  0.1386453  0.5282972  0.0986855
  7  0.1406898  0.5126975  0.1005451
  9  0.1427031  0.4979294  0.1021437

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 5.
```

Figure 35: Test values for KNN model

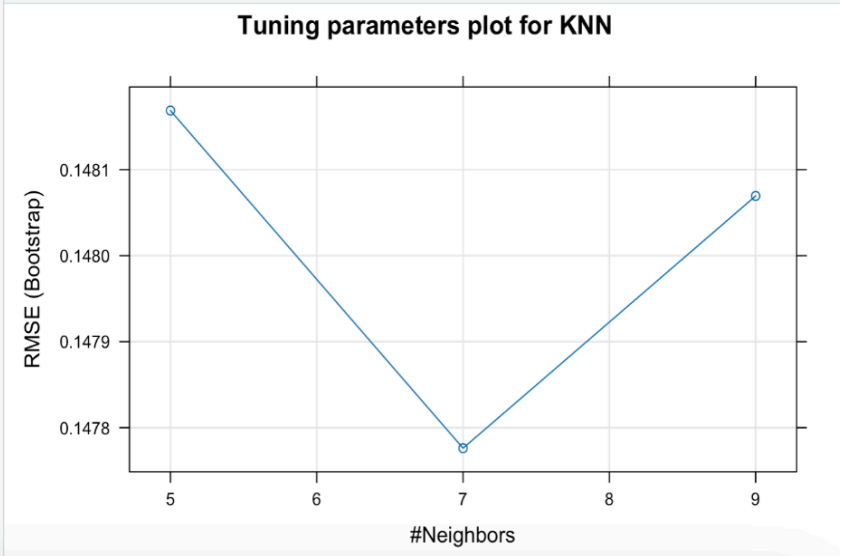


Figure 36: Tuning parameters plot for KNN

RMSE	Rsquared	MAE
0.20931442	0.03412591	0.16905966

Figure 37: Test values for MARS

- **Support Vector Machine (SVM)**

The summary results and performance measure for SVM model can be seen below.

```
Support Vector Machines with Radial Basis Function Kernel

15791 samples
   26 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12632, 12633, 12633, 12634, 12632
Resampling results across tuning parameters:

  C        RMSE       Rsquared   MAE
   0.0625  0.1712192  0.2867197  0.12296755
   0.1250  0.1676723  0.3142991  0.11997100
   0.2500  0.1641889  0.3416307  0.11705987
   0.5000  0.1608016  0.3678071  0.11424747
   1.0000  0.1574664  0.3929896  0.11138618
   2.0000  0.1542400  0.4171003  0.10860049
   4.0000  0.1507798  0.4424711  0.10565127
   8.0000  0.1474567  0.4663320  0.10280593
  16.0000  0.1439530  0.4906385  0.09981475
  32.0000  0.1403818  0.5147717  0.09684856
  64.0000  0.1368646  0.5375550  0.09375659

Tuning parameter 'sigma' was held constant at a value of 0.0120288
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.0120288 and C = 64.
```

Figure 38 : Test values for SVM model

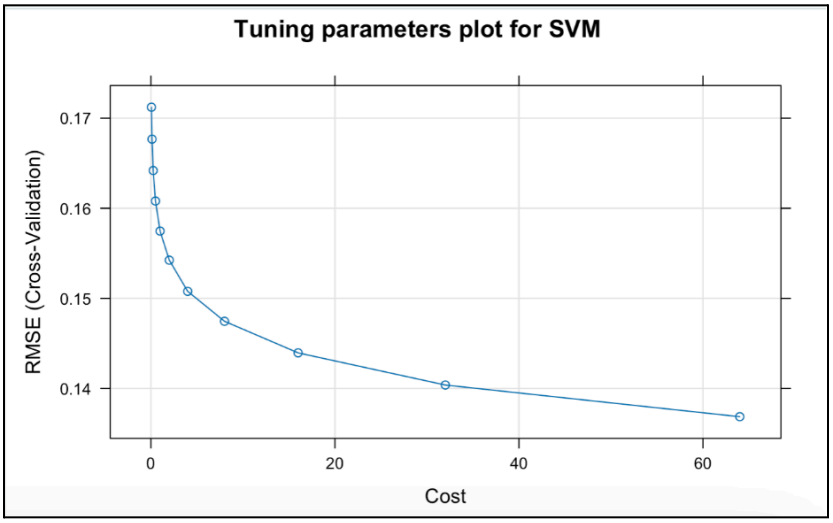


Fig 39: Plot for SVM

```
      RMSE    Rsquared         MAE
0.19662231 0.04586928 0.15927291
```

Figure 40: Test values for SVM

- **Multivariate Adaptive Regression Spline (MARS)**

  The summary results and performance measure for MARS model can be seen below.

```
Multivariate Adaptive Regression Spline

15791 samples
   26 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 15791, 15791, 15791, 15791, 15791, 15791, ...
Resampling results across tuning parameters:

  nprune  RMSE       Rsquared    MAE
   2      0.1980338  0.03343563  0.1520541
  10      0.1753592  0.24215602  0.1286324
  18      0.1724470  0.26710130  0.1263112

Tuning parameter 'degree' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 18 and degree = 1.
```

Figure 41 : Test values for MARS model

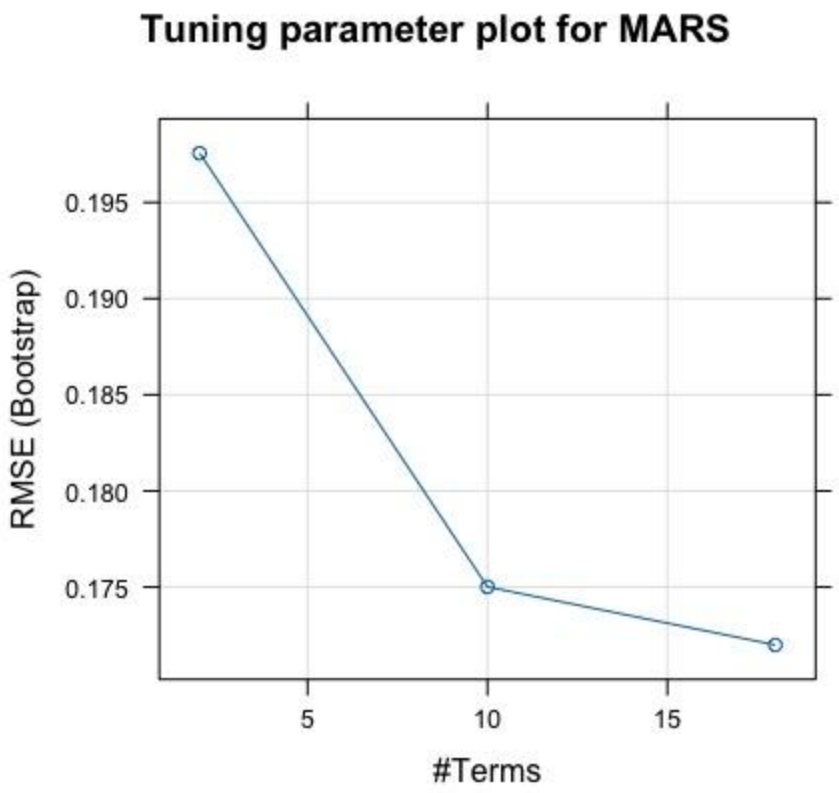


Figure 42: Tuning parameter plot for MARS

```
      RMSE   Rsquared        MAE
0.17096059 0.08043467 0.13995481
```

Figure 43: Test values for MARS

Results:

Linear models					
Model	**Best tuning parameter**	**Training**		**Test**	
		RMSE	**R-squared**	**RMSE**	**R-squared**
Penalized model	alpha = 0.1, lambda	0.175	0.241	0.165	0.1007

	= 0.001				
Ridge model	lambda = 0.1	0.177	0.223	0.157	0.145
Lasso model	fraction = 0.9	0.175	0.241	0.165	0.102
Elastic Net	lambda = 0.0004, fraction = 1	0.175	0.241	0.166	0.100
OLS		1.75	0.241	0.166	0.100
Non-Linear Models					
Model	**Best tuning parameter**	**Training**		**Test**	
		RMSE	**R-squared**	**RMSE**	**R-squared**
KNN	k = 5	0.138	0.528	0.209	0.034
SVM	Sigma= 0.012, C=64	0.136	0.537	0.196	0.046
Neural Network	size = 3, decay =0.1	0.184	0.244	0.184	0.056
MARS	degree = 1, nprune = 18	0.172	0.267	0.171	0.080
Logistic Regression		0.175	0.242	0.166	0.1
PCR	ncomp = 3	0.197	0.036	0.138	0.658
PLS	ncomp =3	0.181	0.188	0.556	0.158

After checking how well our models did using RMSE and R-squared, Principal Component Regression (PCR) and Logistic Regression turned out to be the top performers. PLS had the second-highest R-squared (0.158), and Logistic Regression came in third (0.1) after PCR. But, when we looked at the RMSE values, Logistic Regression had a lower value (0.556) compared to PLS (0.138). Even though PLS had a decent R-squared, the higher RMSE made it less reliable. So, we decided to stick with Logistic Regression as the second-best model after PCR. This highlights the importance of checking different metrics to get a clearer picture of how well our models are really doing.

Top 5 predictors:

Based on our best models, we found the following predictors to be the top predictors:

Logistic regression:

- Rh_8

- T8
- RH_1
- T3
- T9

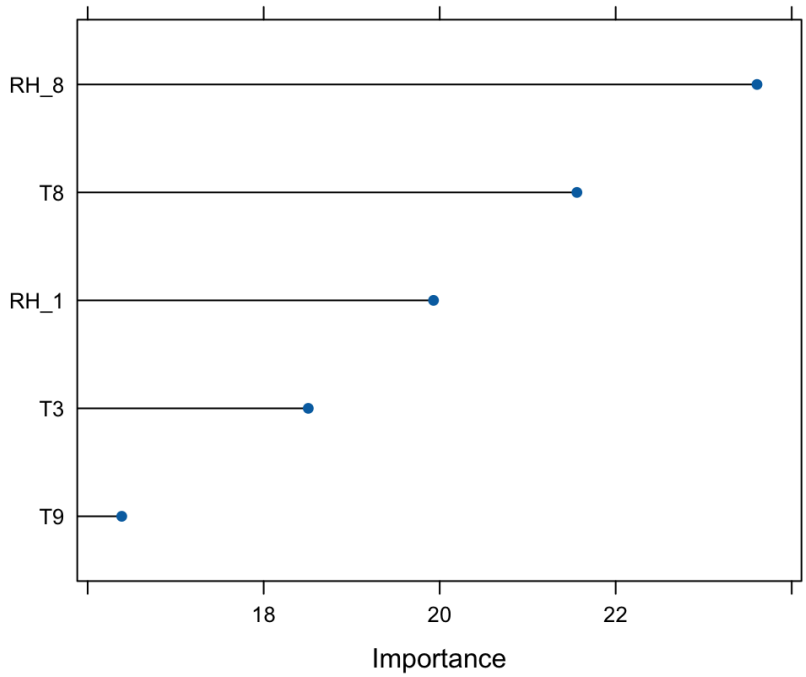


Figure 44: Variable Importance plot for LR model

PCR Model:

- Rh_8,
- T8
- RH_1
- T3
- T9

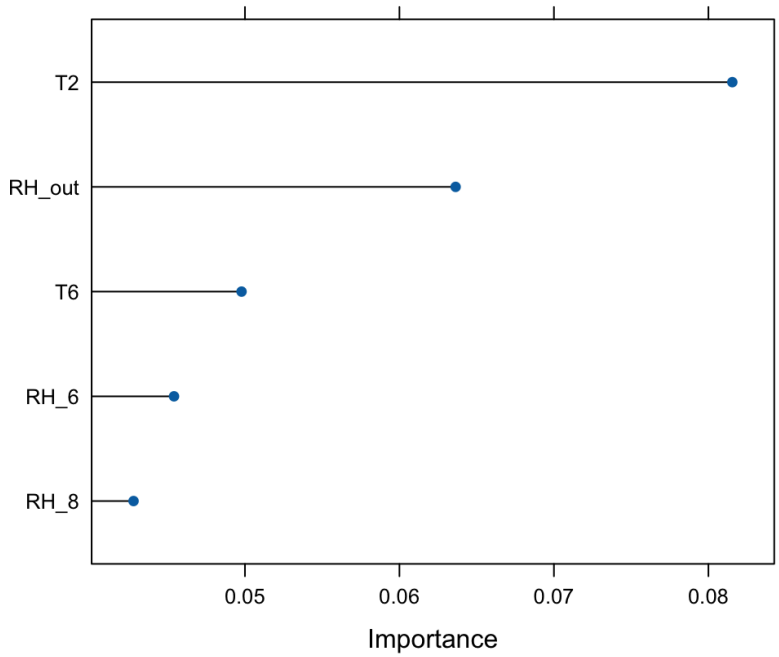


Figure 45: Variable Importance plot for PCR model

Conclusion:

In conclusion, our approach to predicting household appliance energy consumption involved thorough analysis, model selection, and comparison. Focused on a regression problem, we defined clear pre-processing steps and selected suitable models. Starting with Exploratory Data Analysis (EDA), we addressed missingness, degenerate predictors, skewness, and outliers using transformation techniques. Despite having 28 columns (excluding the target variable 'Appliances'), we opted not to proceed with a PCA reduced model to avoid information loss. After robust data cleaning and pre-processing, we split the data into an 80:20 ratio for fitting various regression models, both linear and non-linear. Our goal was to identify top-performing models based on standard metrics—RMSE and R-squared. Testing Ridge, Lasso, Penalized, Enet, KNN, SVM, Neural Network, OLS,MARS, Logistic Regression, PCR, and PLS, we found Principal Component Regression (RMSE = 0.138, R2 = 0.658) and Logistic Regression (RMSE = 0.166, R2 = 0.1) to be our two best-performing models.

References:

1. https://www.kaggle.com/datasets/loveall/appliances-energy-prediction/data

#APPENDIX

R-code:

```
library(ggplot2)
library(caret)
library(e1071)
library(MASS)
library(pamr)

#app <- read.csv('/Users/snehakarki/Downloads/KAG_energydata_complete.csv')
#app <- read.csv("/Users/karenfernandes/Documents/Predictive
modelling/KAG_energydata_complete.csv")
head(app)
summary(app)
str(app)
dim(app)

#Data preprocessing
ggplot(app, aes(Appliances)) +
  geom_histogram(binwidth = 30,color = "#000000", fill = "#0099F8") +
  ggtitle("Response distribution") +
  theme_classic() +
  theme(plot.title = element_text(size = 18))

#Checking missing values
# Check for missing values in the entire dataset
missing_values <- sum(is.na(app))
cat("Total missing values in the dataset:", missing_values, "\n")


# Check for missing values in each column
missing_values_per_column <- colSums(is.na(app))
missing_values_per_column


# Define the list of variables for which you want to create histograms
```

```
variables_to_plot <- c(names(app)[-1])


# Set up the layout for the plots
par(mfrow = c(4, 4), mar = c(4, 4, 2, 1))  # Adjust the rows and columns as needed


# Create histograms in a loop
for (variable in variables_to_plot) {
  hist(app[, variable], main = paste("Histogram of", variable), xlab = variable, col = "lightblue", border =
"black")
}
# Reset the layout to the default
par(mfrow = c(1, 1))

cont_vars <- app[c(names(app)[-1])]
numeric_vars <- app[, sapply(app, is.numeric)]
dim(numeric_vars)
# Create individual boxplots for each numeric predictor
par(mfrow = c(4, 4)) # Set the layout for multiple plots
for (var_name in names(cont_vars)) {
  boxplot(numeric_vars[[var_name]], main = var_name, col = "#ad1538", border = "black")
}
# Reset the plotting layout
par(mfrow = c(1, 1))



#Skewness values
skewness_values <- sapply(cont_vars, skewness)
skewness_summary <- data.frame(Predictor = names(skewness_values), Skewness =skewness_values)
print(skewness_summary)

#Transforming the data

trans <- preProcess(cont_vars, method = c("BoxCox", "center", "scale"))
app_transformed <- predict(trans, cont_vars)


#Checking skewness after transformation
skewness_trans<- sapply(app_transformed, skewness)
skewness_summ <- data.frame(Predictor = names(skewness_trans), Skewness =skewness_trans)
```

```
print(skewness_summ)

#Checking distribution after transformation
par(mfrow = c(4, 4))  # Adjust the rows and columns as needed

# Create histograms in a loop
for (i in names(app_transformed)) {
  hist(app_transformed[[i]], main = paste("Histogram of", i), xlab = variable, col = "lightblue", border =
"black")
}
# Reset the layout to the default
par(mfrow = c(1, 1))



#Spatial sign transformation to remove outliers
# Spatial sign transformation
spatial_sign <- function(x) {
  norms <- rowSums(x^2)^0.5
  return(x / norms)
}
transformed_data <- spatial_sign(app_transformed)

# Create individual boxplots for each numeric predictor
par(mfrow = c(4, 4)) # Set the layout for multiple plots
for (k in names(transformed_data)) {
  boxplot(transformed_data[[k]], main = k, col = "#15ad4f", border = "black")
}
# Reset the plotting layout
par(mfrow = c(1, 1))

#No attributes with Near zero variance
near_zero_var <- nearZeroVar(transformed_data, saveMetrics = TRUE)
print(near_zero_var)


preds <- transformed_data[, -c(1, 2)]
preds
dim(preds)

resp <- transformed_data$Appliances
resp
```

```
combined_data <- cbind(preds, resp)

#######PCA
pca_result <- prcomp(preds, scale. = TRUE)

# Summary of PCA
summary(pca_result)

# Scree plot to visualize the variance explained by each principal component
screeplot(pca_result, type = "line", main = "Scree Plot")

# Variance explained by each principal component
cumsum_var <- cumsum(pca_result$sdev^2) / sum(pca_result$sdev^2)
cumsum_var

#number of PCs based on the threshold
desired_variance <- 0.95
num_components <- which(cumsum_var >= desired_variance)[1]

# final number of principal components
pca_result_selected <- prcomp(preds, scale. = TRUE, rank. = num_components)

# data transformation - getting desired number of components
data_pca <- predict(pca_result_selected, newdata = preds)

# The transformed data contains the scores for each principal component
head(data_pca)
dim(data_pca)

#As we can see, we have reduced the number of columns/predictor variables
#from 26 to 13 while still capturing 95% of the data

library(caret)
library(e1071)
library(class)
library(nnet)
library(glmnet)

# Split the data
set.seed(123)
train_indices <- createDataPartition(resp, p = 0.8, list = FALSE)
```

```
train_data <- preds[train_indices, ]
test_data <- preds[-train_indices, ]
train_resp <- resp[train_indices]
test_resp <- resp[-train_indices]

###Enet model####
set.seed(476)
glmnTuned <- train(train_data,
                   train_resp,
                   method = "glmnet",
                   #preProc = c("center", "scale"),
                   #metric = "Accuracy",
                   tuneGrid = expand.grid(alpha = seq(0, 1, by = 0.1), lambda = seq(0.1, 1, by = 0.1)),
                   trControl = trainControl(method = "cv", number = 5))
glmpred <- predict(glmnTuned, newdata = test_data_bio)
#confusionMatrix(data = glmpred, reference = injury_test, mode = 'everything')
summary(glmnTuned)
plot(glmnTuned)

#Lasso regression model
lasso_model <- cv.glmnet(train_data, train_resp, alpha = 1)

#Ridge regression model
ridge_model <- cv.glmnet(train_data, train_resp, alpha = 0)


####GLM-Penalized model#####
###Penalised model
set.seed(476)
glmnTuned <- train(train_data,
                   train_resp,
                   method = "glmnet",
                   #preProc = c("center", "scale"),
                   #metric = "Accuracy",
                   #tuneGrid = expand.grid(alpha = seq(0, 1, by = 0.1), lambda = seq(0.1, 1, by = 0.1)),
                   trControl = trainControl(method = "cv", number = 5))
glmnTuned
glmpred <- predict(glmnTuned, newdata = test_data)

summary(glmnTuned)
plot(glmnTuned)
```

```
glm_perf<- postResample(glmpred, test_data[ ,1])
glm_perf
```

#########Non-Linear models #####

```
ctrl <- trainControl(summaryFunction = defaultSummary,
                     classProbs = TRUE, savePredictions = TRUE)
```

#PCR model

```
# Train the PCR model
pcr_model <- train(
  x = train_data,
  y = train_resp,
  method = "pcr",
  #tuneLength = 5,  # Number of principal components to consider (you can adjust this)
  trControl =  trainControl(method = "cv", number = 5)
)
pcr_model
pcr_model$results[3,]
plot(pcr_model, main = 'Tuning parameter plot for PCR')

pcr_pred <- predict(pcr_model, newdata = test_data)
pcr_perf<- postResample(pcr_pred, test_data[ ,1])
Pcr_perf
```

#PLS model

```
# Train the PLS model
pls_model <- train(
  x = train_data,
  y = train_resp,
  method = "pls",
  #tuneLength = 5,  # Number of components to consider (you can adjust this)
  trControl = trainControl(method = "cv", number = 5)
)

pls_model
pls_model$results[3,]
plot(pls_model)
```

 # Logistic Regression model

```
lr_model <- train(
  x = train_data,
  y = train_resp,
  method = "glm",
  #family = "binomial",  # Specify the family for logistic regression
  trControl = trainControl(method = "cv", number = 5)
)

# Display the model
lr_model

# Plot the model
plot(lr_model)
pcr_perf<- postResample(pcr_pred, test_data[ ,1])
pcr_perf

pls_perf<- postResample(pcr_pred, test_data[ ,1])
pls_perf

lr_perf<- postResample(lr_pred, test_data[ ,1])
lr_perf


####SVM####
#Train
svm_mod <- train(train_data,
                 factor(train_resp),
                 method = "svmRadial",
                 metric = "Accuracy",
                 #tuneGrid = svmGrid,
                 fit = FALSE,
                 trControl = ctrl)
svm_mod
plot(svm_mod, main = 'Tuning parameters plot for SVM')
svm_mod$results[11,]

#Test
svm_pred <- predict(svm_mod, test_data)
svm_pred
```

```
#####KNN#####
#Train
knn_mod <- train(train_data,
                 train_resp,
                 method = "knn",
                 metric = "RMSE",
                 trControl = ctrl)

knn_mod
plot(knn_mod,  main = 'Tuning parameters plot for KNN')
knn_mod$results[2,]


#####MARS model#####
mars_mod <- train(train_data,
                   train_resp,
                   method = "earth")
mars_mod$results[3,]
mars_mod
plot(mars_mod, main = "Tuning parameter plot for MARS")


####Neural Network model######
nn_model <- train(train_data, train_resp, method = "nnet",
                  trControl = trainControl(method = "cv", number = 5),
                  preProc = c("center", "scale"))
nn_model
nn_model$results[9,]
plot(nn_model, main = "Tuning parameter plot for Neural Network")


#Prediction#
svm_pred <- predict(svm_model, test_data)
knn_pred <- predict(knn_model, test_data)
marsPred <- predict(mars_model, newdata = test_data)
nn_pred <- predict(nnet_mod, test_data)
lasso_pred <- predict(lasso_model, s = lasso_model$lambda.min, newx = test_data)
ridge_pred <- predict(ridge_model, s = ridge_model$lambda.min, newx = test_data)
pcr_pred <- predict(pcr_model, newdata = test_data)
pls_pred <- predict(pls_model, newdata = test_data)
lr_pred <- predict(lr_model, newdata = test_data)
ols_pred <- predict(ols_model, newdata = test_data)
```

 # Evaluate the models using RMSE and R-squared

```
svm_perf<- postResample(svm_pred, test_data[,1])
svm_perf
knn_perf<- postResample(knn_mod, test_data[ ,1])
knn_perf
mars_perf<- postResample(mars_mod, test_data[ ,1])
mars_perf
nn_perf<- postResample(nn_pred, test_data[ ,1])
nn_perf
```