Capstone Project

Divyesh Peshavaria

Machine Learning Engineer Nanodegree

APPLIANCE ENERGY PREDICTION

1. Definition

a. Project Overview

This project aims to predict the energy consumption by home appliances. With the advent of smart homes and rising need for energy management, existing smart home systems can benefit from accurate prediction. If the energy usage can be predicted for every possible state of appliances, then device control can be optimized for energy savings as well.

This is a case of Regression analysis which is part of Supervised Learning problem. Appliance energy usage is the target variable while sensor data and weather data are the features.

Dataset source:
http://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

b. Problem Statement

Develop a Supervised learning model using Regression algorithms to predict the appliance energy usage using sensor readings and weather data as features.

c. Metrics

Since this is a regression problem, the metric used will be “*Coefficient of Determination*”, in other words denoted as R^2 (R squared) which gives a measure of the variance of target variable that can be explained using the given features.
It can be mathematically defined as:

$$R^2 \equiv 1 - \frac{SS_{\mathrm{res}}}{SS_{\mathrm{tot}}}$$

where,

SS$_{\mathbf{res}}$ = Residual sum of squares
SS$_{\mathbf{tot}}$ = Total sum of squares

For this project, I will use ‘***r2_score*()**’ method provided by the scikit-learn library to evaluate the performance of the model.

2. Analysis

a. Data Exploration

The dataset has 28 features and 1 target variable described as follows:

NAME	DESCRIPTION	UNIT
Features		
T1	Kitchen Temperature	℃
T2	Living Room Temperature	℃
T3	Laundry Room Temperature	℃
T4	Office Temperature	℃
T5	Bathroom Temperature	℃
T6	Temperature outside Building (North)	℃
T7	Ironing Room Temperature	℃
T8	Teenager Room Temperature	℃
T9	Parents Room Temperature	℃
T_out	Outside Temperature (Weather Station)	℃
T_dewpoint	Dewpoint Temperature (Weather Station)	℃
RH_1	Kitchen Humidity	%
RH_2	Living Room Humidity	%
RH_3	Laundry Room Humidity	%
RH_4	Office Humidity	%
RH_5	Bathroom Humidity	%
RH_6	Humidity outside Building (North)	%
RH_7	Ironing Room Humidity	%
RH_8	Teenager Room Humidity	%
RH_9	Parents Room Humidity	%
RH_out	Outside Humidity (Weather Station)	%
Pressure	Outside Pressure (Weather Station)	mm Hg
Wind speed	Outside Windspeed (Weather Station)	m/s
Visibility	Visibility (Weather Station)	km
Date	Timestamp of the reading	yyyy-mm-dd HH:MM:SS
rv1	Random Variable 1	-
rv2	Random Variable2	-
Lights	Energy used by lights	Wh
Target Variable		
Appliances	Total energy used by Appliances	Wh

Out of these features, I won’t be using the last 4 features as the problem is that of Regression and not Time series forecasting

("**Date**"). Also, the goal is to predict total energy consumption and not category-wise energy consumption ("**Lights**").
Therefore,

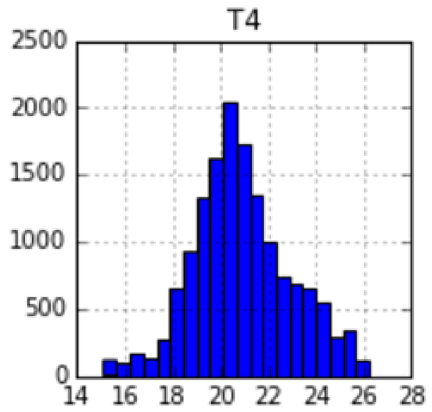
Number of features = 24
Number of target variables = 1
Number of instances in training data = 14,801
Number of instances in testing data = 4,934
Total number of instances = 19,735
Null values:- None

All features have numerical values. There are no categorical or ordinal features in this dataset.

b. Exploratory Visualization

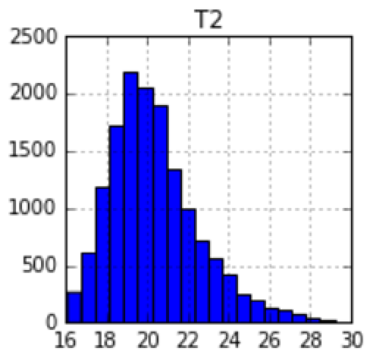
Most features have their values in normal distribution. For example:

i.

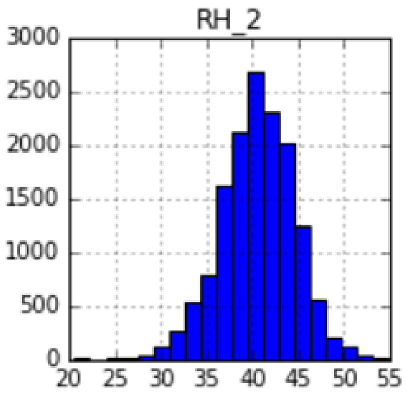


Out of which, some features are skewed left/right as shown below:

ii. Left

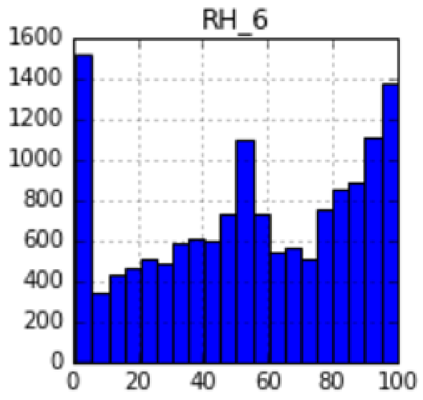


iii. Right

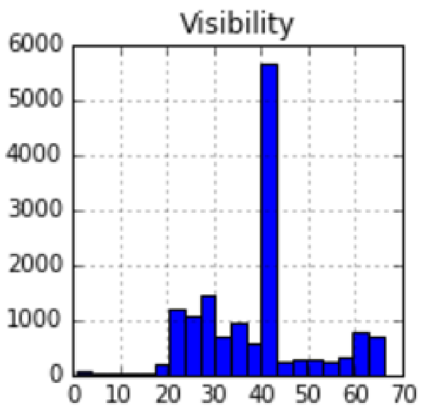


Some features don’t have normal distribution as shown below:

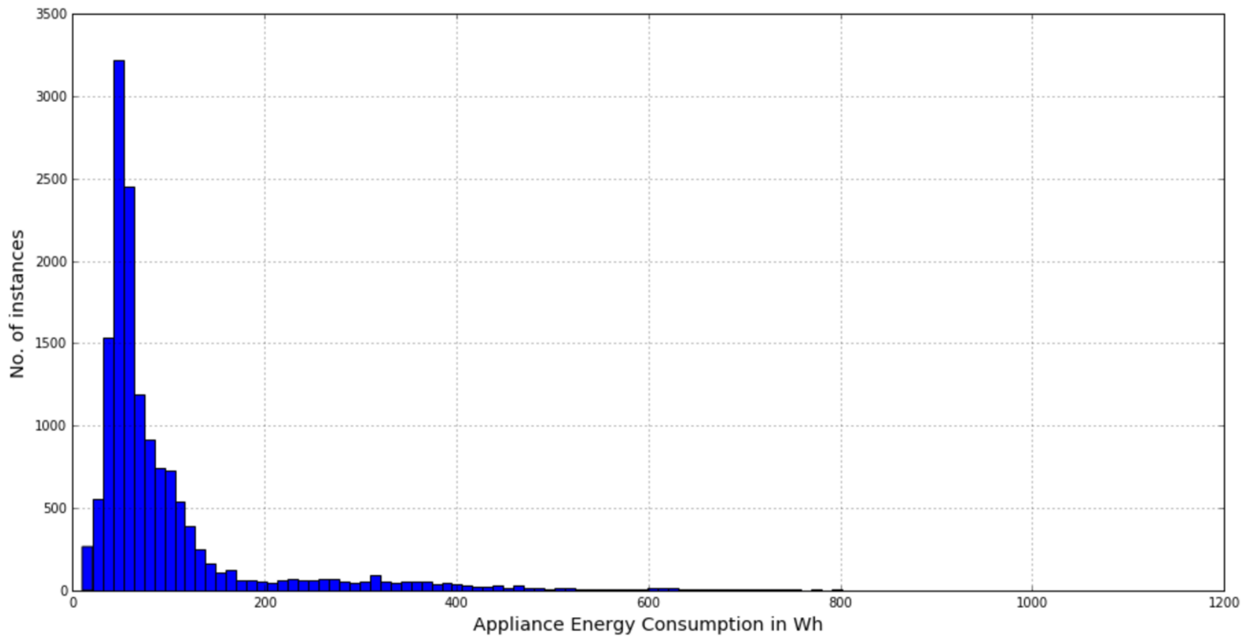
i.



ii.



Distribution of the target variable:



Observations:-

i. Most features are normally distributed.

ii. The target variable has a highly skewed distribution and it doesn't have linear relation with any other features.

iii. The feature ***T9*** is highly correlated with features ***T3***, ***T5*** and ***T7***.

iv. The feature ***T6*** is highly correlated with feature ***T_out***.

c. Algorithms and Techniques

I will try the following algorithms for Regression:

i. Linear Models
    1. Linear Regression
    2. Ridge Regression
    3. Lasso Regression

ii. Tree based models
    1. Random Forests
    2. Gradient Boosting Machines
    3. Extremely Randomized Trees

iii. Neural Networks
    1. Multi-layer Perceptron

d. Benchmark

The benchmark model is Linear Regression on unscaled data using all the features.

Observations:

i. Training score: 14.687%
ii. Test data score: 14.258%
iii. Time taken to fit: 0.043 seconds

3. Methodology

a. Data Preprocessing

Ranges of features irrespective of units

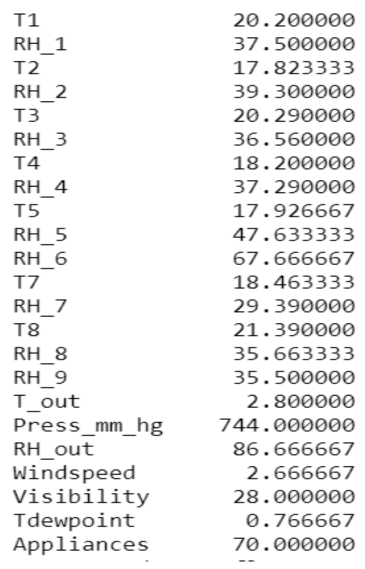
Temperature	-6 to 30
Humidity	1 to 100
Windspeed	0 to 14
Visibility	1 to 66
Pressure	729 to 772
Appliance Energy Usage	10 to 1080

Due to different ranges of features, it is possible that some features will dominate the Regression algorithm. To avoid this situation, all features need to be scaled.

Thus, the data was scaled to 0 mean and unit variance using the ***StandardScaler*** class in ***sklearn.preprocessing*** module.

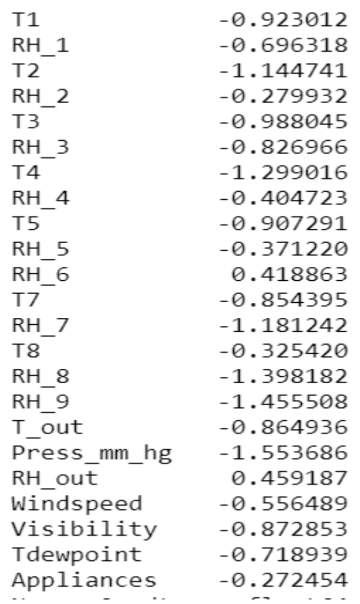
The first row of data is shown before and after scaling.

Before scaling:



```
T1               20.200000
RH_1             37.500000
T2               17.823333
RH_2             39.300000
T3               20.290000
RH_3             36.560000
T4               18.200000
RH_4             37.290000
T5               17.926667
RH_5             47.633333
RH_6             67.666667
T7               18.463333
RH_7             29.390000
T8               21.390000
RH_8             35.663333
RH_9             35.500000
T_out             2.800000
Press_mm_hg     744.000000
RH_out           86.666667
Windspeed         2.666667
Visibility       28.000000
Tdewpoint         0.766667
Appliances       70.000000
```

After scaling:



```
T1              -0.923012
RH_1            -0.696318
T2              -1.144741
RH_2            -0.279932
T3              -0.988045
RH_3            -0.826966
T4              -1.299016
RH_4            -0.404723
T5              -0.907291
RH_5            -0.371220
RH_6             0.418863
T7              -0.854395
RH_7            -1.181242
T8              -0.325420
RH_8            -1.398182
RH_9            -1.455508
T_out           -0.864936
Press_mm_hg     -1.553686
RH_out           0.459187
Windspeed       -0.556489
Visibility      -0.872853
Tdewpoint       -0.718939
Appliances      -0.272454
[illegible]
```

I also removed the columns ***T6*** and ***T9*** which had a significant correlation with columns ***T_out*** and (***T3, T5, T7***) respectively. As a result, there are 22 features in the training data.

b. Implementation

The model implementation is done in 3 steps:

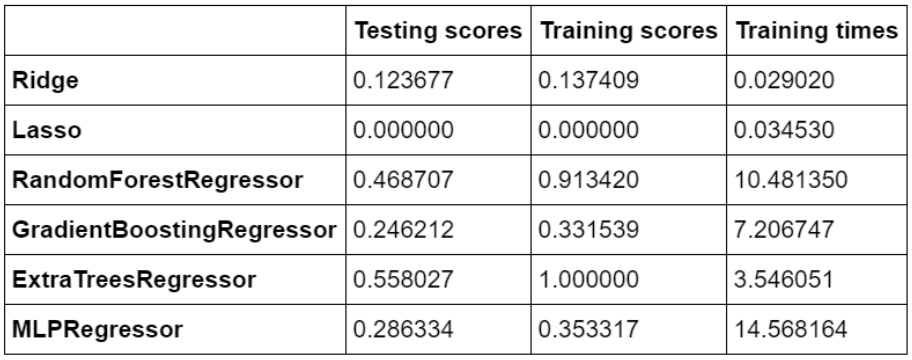
i. Create a ***pipeline()*** function to execute each Regressor and record the metrics.
ii. Pass each Regressor to above pipeline function from ***execute_pipeline()***.
iii. Consolidate the obtained metrics into a DataFrame in the function ***get_properties()*** and plot these metrics using a bar graph.

List of Algorithms tested:

i. sklearn.linear_model.Ridge
ii. sklearn.linear_model.Lasso
iii. sklearn.ensemble.RandomForestRegressor
iv. sklearn.ensemble.GradientBoostingRegressor
v. sklearn.ensemble.ExtraTreesRegressor
vi. sklearn.neural_network.MLPRegressor

Performance metric used:
R2 score (the ***r2_score()*** method mentioned in section 1.3) which is internally used by the ***score()*** method of all Regressors mentioned above.

Results:



	Testing scores	Training scores	Training times
Ridge	0.123677	0.137409	0.029020
Lasso	0.000000	0.000000	0.034530
RandomForestRegressor	0.468707	0.913420	10.481350
GradientBoostingRegressor	0.246212	0.331539	7.206747
ExtraTreesRegressor	0.558027	1.000000	3.546051
MLPRegressor	0.286334	0.353317	14.568164

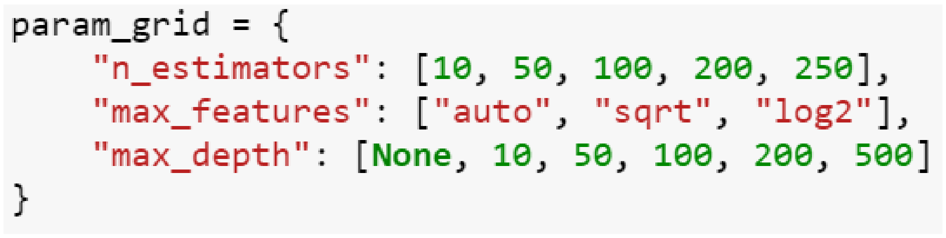
As observed from results, ExtraTreesRegressor performs better than other regressors.

c. Refinement

For refining the model, I tweaked the following properties of ExtraTreesRegressor:

i. n_estimators: The number of trees to be used.
ii. max_features: The number of features to be considered at each split.
iii. max_depth: The maximum depth of the tree.

My feature superset is as follows:



```
param_grid = {
    "n_estimators": [10, 50, 100, 200, 250],
    "max_features": ["auto", "sqrt", "log2"],
    "max_depth": [None, 10, 50, 100, 200, 500]
}
```

Here, the values of ***max_features*** parameter defines the function to be applied on total number of features to obtain the new number of features to be considered during splits.

For ***max_depth***, None means keep splitting until all leaves are pure or they have less samples than ***min_samples_split*** parameter whose default value is 2.

Before Tuning, the R2 score on test set was 0.558. After tuning, it rose to 0.610, a performance gain of **5.2%**.

4. Results

a. Model Evaluation and Validation

Features of the untuned model:

i. n_estimators = 10
ii. max_features = n_features = 22
iii. max_depth = None

Features of best model after hyper parameter tuning:

i. n_estimators = 250
ii. max_features = log2(n_features) = log2(22) ~ 4
iii. max_depth = None

Robustness check:

The best model is trained on reduced feature space having only 5 highest ranked features in terms of importance instead of 22 features.

It's R2 score on test data is **0.499**.
R2 score of untuned model = **0.558**.
Difference = **0.059** or **5.9%**.

Therefore, we can see that even though the feature space is reduced drastically (by more than 75%), the relative loss in performance on test data is less.

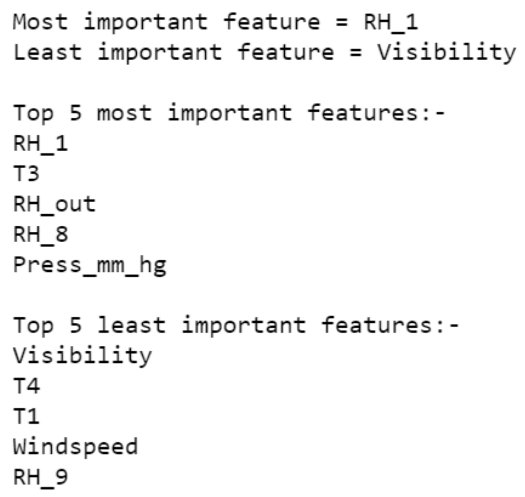
b. Justification

The final model's score on training dataset is **1.0**, and **0.61** on test dataset. The score on test dataset is more than **4 times** the score of benchmark model, which was **0.143**. Therefore, it can be argued that we have reached a satisfactory solution.

5. Conclusion

a. Free-form visualization

According the best fitted model, the feature importance is as follows:

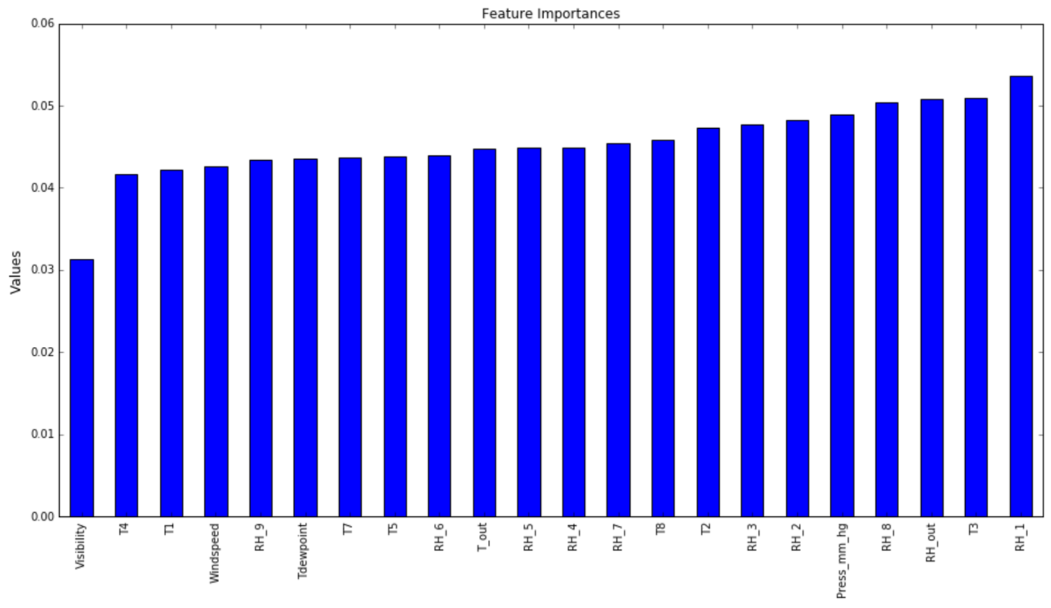


```
Most important feature = RH_1
Least important feature = Visibility

Top 5 most important features:-
RH_1
T3
RH_out
RH_8
Press_mm_hg

Top 5 least important features:-
Visibility
T4
T1
Windspeed
RH_9
```

Visual representation of all features:



It can be observed that on an average, humidity affects power consumption more than temperature. This is evident from the fact that more number of humidity readings are towards the higher end of the graph as compared to temperature readings.

Also, out of weather parameters, Humidity and Atmospheric pressure affect power consumption more significantly than others. This is in line with the general assumption that factors like Windspeed and Visibility shouldn’t affect the power consumption inside the home.

An important conclusion drawn from this visualization is that that although natural humidity cannot be controlled, controlling humidity inside the home can lead to energy savings.

b. Reflection

This project can be summarized as the sequence of following steps:
1. Searching for a problem by looking at datasets on UCI Machine Learning repository and Kaggle and deciding between Classification and Regression problems.
2. Visualizing various aspects of dataset.
3. Preprocessing the data and feature selection.
4. Deciding the algorithms to be used to solve the problem.
5. Creating a benchmark model.
6. Applying selected algorithms and visualizing the results.
7. Hyper parameter tuning for the best algorithm and reporting the test score of best model.
8. Discuss importance of selected features and check the robustness of model.

Out of this, I found steps 1, 2 and 5 very interesting. Deciding between Classification and Regression was an important hurdle. But, as I had approached a few classification problems before, I decided that it would be more exciting to solve a Regression based problem.

Therefore, visualizing a dataset from the point of view of solving a Regression problem where your output isn’t defined among a few classes was particularly challenging.

Also, in the case of Classification, a benchmark model can be created using the concept of chance i.e. Accuracy = 1/n_classes. In this project, I had initially decided to create two benchmark models, one that would always return the mean of the target variable and one which would return the median. But, after visualizing the data and concluding that there are no Linear relationships of any feature with the target variable, I realized that a Linear Regression model may serve as a better benchmark.

c. Improvement

A few of the ways the solution can be improved are:

i. Discarding seemingly irrelevant weather features like Windspeed and Visibility.
ii. Performing more aggressive feature engineering.
iii. Using Grid search instead randomized search to search the parameter space exhaustively and determine the best solution.
iv. As an add-on to previous step, more number of parameters can be added to the parameter space.