Machine Learning Engineer Nanodegree
Capstone Project
Appliances Energy Prediction
Mayur Sand

Project Overview

In this time of global uncertainty one thing is clear the world needs energy and in increasing quantities to support economic and social progress and build a better quality of life, in particular in developing countries. But even in today's time there are many places especially in developing world where there are outages .

These outages are primary because of excess load consumed by appliances at home . Heating and cooling appliances takes most power in house. In this project we will be analyzing the appliance usage in the house gathered via home sensors .All readings are taken at 10 mins intervals for 4.5 months . The goal is to predict energy consumption by appliances .

In the age of smart homes , ability to predict energy consumption can not only save money for end user but can also help in generating money for user by giving excess energy back to Grid (in case of solar panels usage). In this case regression analysis will be used to predict Appliance energy usage based on data collected from various sensors.

Related academic research and earlier work:

http://dx.doi.org/10.1016/j.enbuild.2017.01.083

https://github.com/LuisM78/Appliances-energy-prediction-data

Problem Statement

We should predict Appliance energy consumption for a house based on factors like temperature , humidity & pressure . In order to achieve this , we need to develop a supervised learning model using regression algorithms . Regression algorithims are used as data consist of continuous features and there are no identification of appliances in dataset

Metrics

The metric used are "Coefficient of Determination" also denoted as R2 score & RMSE (Root Mean Squred Error). They are use to measure quality and residual of regression problems.

Mathematically R2 score is defined as

$$\hat{R}^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

The numerator is MSE (average of the squares of the residuals) and the denominator is the variance in Y values. Higher the MSE, smaller the R2-Score & poorer is the model. R2 score shows the statistical robustness of the model.

Mathematically RMSE score is defined as

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

It represents the sample standard deviation of the differences between predicted values and observed values (called residuals) .RMSE indicates how accurate the predictions are to actual values.

Data Exploration

The dataset was collected by sensors placed inside the house and outside readings came from the nearby weather station. The main attributes are temperature, humidity and pressure readings. Each observation measures electricity in a 10-minute interval. The temperatures and humidity have been averaged for 10-minute intervals.

Independent variables : 28(11 temperature, 10 humidity, 1 pressure, 2 randoms)
Dependent variable : 1 (Appliances)

Attribute Information:

1. date : time year-month-day hour:minute:second
2. lights : energy use of light fixtures in the house in Wh
3. T1 : Temperature in kitchen area, in Celsius
4. T2 : Temperature in living room area, in Celsius
5. T3 : Temperature in laundry room area
6. T4 : Temperature in office room, in Celsius
7. T5 : Temperature in bathroom, in Celsius
8. T6 : Temperature outside the building (north side), in Celsius
9. T7 : Temperature in ironing room, in Celsius
10. T8 : Temperature in teenager room 2, in Celsius
11. T9 : Temperature in parents' room, in Celsius
12. T_out : Temperature outside (from Chievres weather station), in Celsius
13. Tdewpoint : (from Chievres weather station), Â°C
14. RH_1 : Humidity in kitchen area, in %
15. RH_2 : Humidity in living room area, in %
16. RH_3 : Humidity in laundry room area, in %
17. RH_4 : Humidity in office room, in %
18. RH_5 : Humidity in bathroom, in %
19. RH_6 : Humidity outside the building (north side), in %
20. RH_7 : Humidity in ironing room, in %
21. RH_8 : Humidity in teenager room 2, in %
22. RH_9 : Humidity in parents' room, in %
23. RH_out :Humidity outside (from Chievres weather station), in %
24. Pressure : (from Chievres weather station), in mm Hg
25. Wind speed: (from Chievres weather station), in m/s
26. Visibility :(from Chievres weather station), in km
27. Rv1 :Random variable 1, non-dimensional

28.Rv2 :Random variable 2, non-dimensional

29. Appliances : Total energy used by appliances, in Wh

Sample Data

```
data.head()
```

	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	...	T9	RH_9	T_out	Press_mm_hg	RH_out	Windsp
0	2016-01-11 17:00:00	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.000000	...	17.033333	45.53	6.600000	733.5	92.0	7.000
1	2016-01-11 17:10:00	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.000000	...	17.066667	45.56	6.483333	733.6	92.0	6.666
2	2016-01-11 17:20:00	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.926667	...	17.000000	45.50	6.366667	733.7	92.0	6.333
3	2016-01-11 17:30:00	50	40	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.890000	...	17.000000	45.40	6.250000	733.8	92.0	6.000
4	2016-01-11 17:40:00	60	40	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.890000	...	17.000000	45.40	6.133333	733.9	92.0	5.666

5 rows × 29 columns

Key Observations :

1. Date column is only used for understanding the consumption vs date time behavior and given this is not a time series problem it was removed . I added one more column temporarily (WEEKDAY)which focusses on if a day was weekday or weekend in order to check the difference in appliance consumption
2. Light column was also removed as the are the reading of submeter and we are not focusing on appliance specific reading
3. Number of Independent variables at this stage – 26
4. Number of Dependent variable at this stage – 1
5. Total number of rows - 19735
6. The data set will be split 75-25 % between train & test.
7. Total # of rows in training set - 14801
8. Total # of rows in test set - 4934
9. All the features have numerical values. There are no categorical or ordinal features.
10. Number of missing values & null values = 0

```
feature_vars.head(2)
```

	T1	T2	T3	T4	T5	T6	T7	T8	T9	RH_1	...	RH_8	RH_9	T_out	Tdewpoint	RH_out	Press_mm_hg	Windspeed
9544	22.6	19.5	21.50	22.89	19.166667	2.863333	21.0	22.89	19.89	34.70	...	38.5	37.26	2.233333	0.4	87.666667	764.200000	1.333333
19366	23.7	21.0	25.39	23.60	19.890000	5.500000	23.0	24.20	22.60	40.29	...	43.4	44.59	6.266667	5.3	93.666667	757.233333	1.000000

2 rows × 26 columns

Descriptive Statistics :

Temperature columns

	T1	T2	T3	T4	T5	T6	T7	T8	T9
count	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000
mean	21.685153	20.343487	22.268005	20.857724	19.589105	7.923834	20.264300	22.028348	19.484679
std	1.605537	2.199037	1.999986	2.040012	1.842916	6.083047	2.105079	1.951399	2.010610
min	16.790000	16.100000	17.200000	15.100000	15.335000	-6.065000	15.390000	16.306667	14.890000
25%	20.745000	18.790000	20.790000	19.533333	18.290000	3.663333	18.700000	20.790000	18.000000
50%	21.600000	20.000000	22.100000	20.666667	19.390000	7.300000	20.028571	22.111111	19.390000
75%	22.600000	21.533333	23.290000	22.100000	20.633333	11.293333	21.600000	23.390000	20.600000
max	26.260000	29.856667	29.200000	26.200000	25.745000	28.290000	25.963333	27.230000	24.500000

Humidity columns

	RH_1	RH_2	RH_3	RH_4	RH_5	RH_6	RH_7	RH_8	RH_9
count	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000
mean	40.271333	40.432370	39.252994	39.041339	50.939261	54.596942	35.401239	42.944133	41.567732
std	3.983201	4.081775	3.263513	4.355528	8.964531	31.163493	5.134281	5.240388	4.167305
min	27.233333	20.463333	28.766667	27.660000	29.815000	1.000000	23.200000	29.600000	29.166667
25%	37.363333	37.900000	36.900000	35.530000	45.400000	30.023333	31.500000	39.069091	38.500000
50%	39.656667	40.500000	38.560000	38.400000	49.090000	55.290000	34.900000	42.397143	40.900000
75%	43.090000	43.290000	41.790000	42.193333	53.694286	83.126667	39.000000	46.560000	44.363333
max	63.360000	56.026667	50.163333	51.090000	96.321667	99.900000	51.400000	58.780000	53.326667

Weather columns

	T_out	Tdewpoint	RH_out	Press_mm_hg	Windspeed	Visibility
count	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000
mean	7.422035	3.768053	79.744066	755.561311	4.057009	38.345054
std	5.304241	4.189370	14.952250	7.398129	2.449080	11.785900
min	-5.000000	-6.600000	24.500000	729.366667	0.000000	1.000000
25%	3.700000	0.933333	70.000000	750.983333	2.000000	29.000000
50%	6.933333	3.450000	83.833333	756.100000	3.666667	40.000000
75%	10.433333	6.566667	91.666667	760.966667	5.500000	40.000000
max	26.033333	15.500000	100.000000	772.300000	14.000000	66.000000

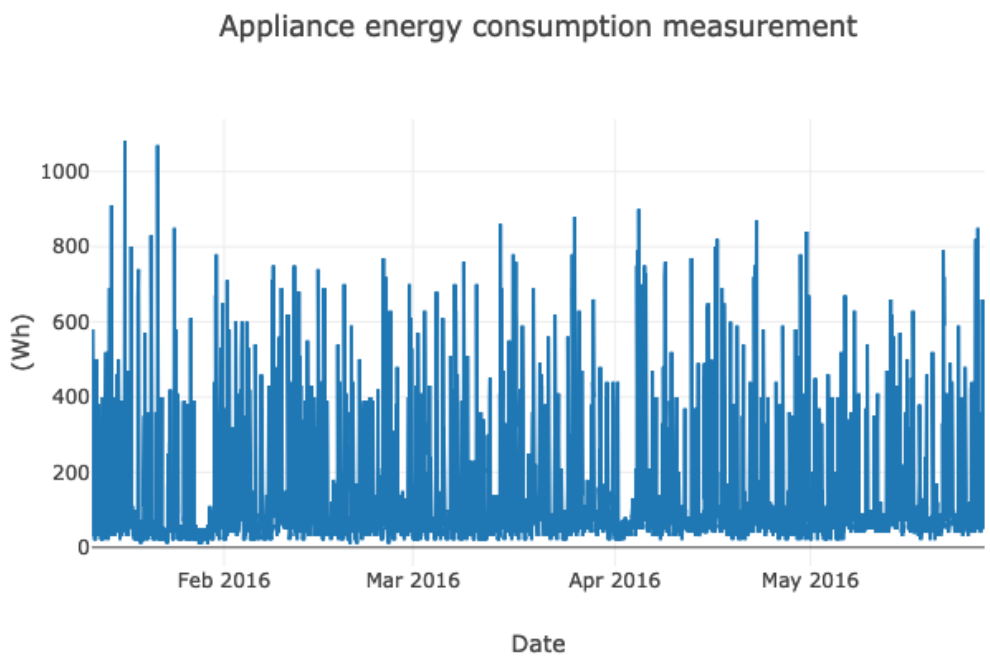
Appliance column (Dependent variable)

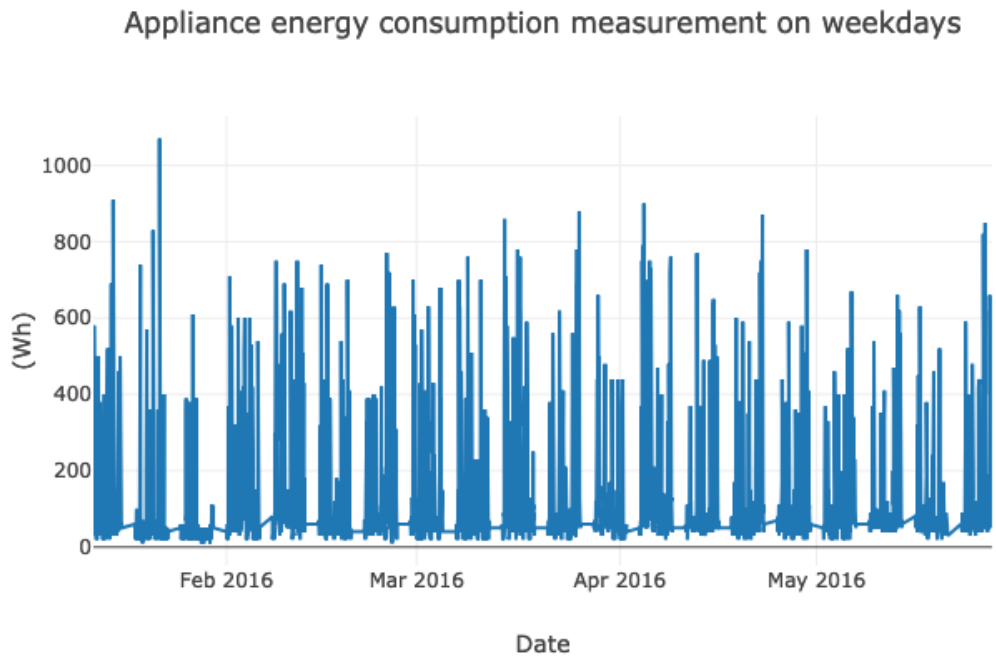
	Appliances
count	14801.000000
mean	97.835281
std	102.928289
min	10.000000
25%	50.000000
50%	60.000000
75%	100.000000
max	1080.000000

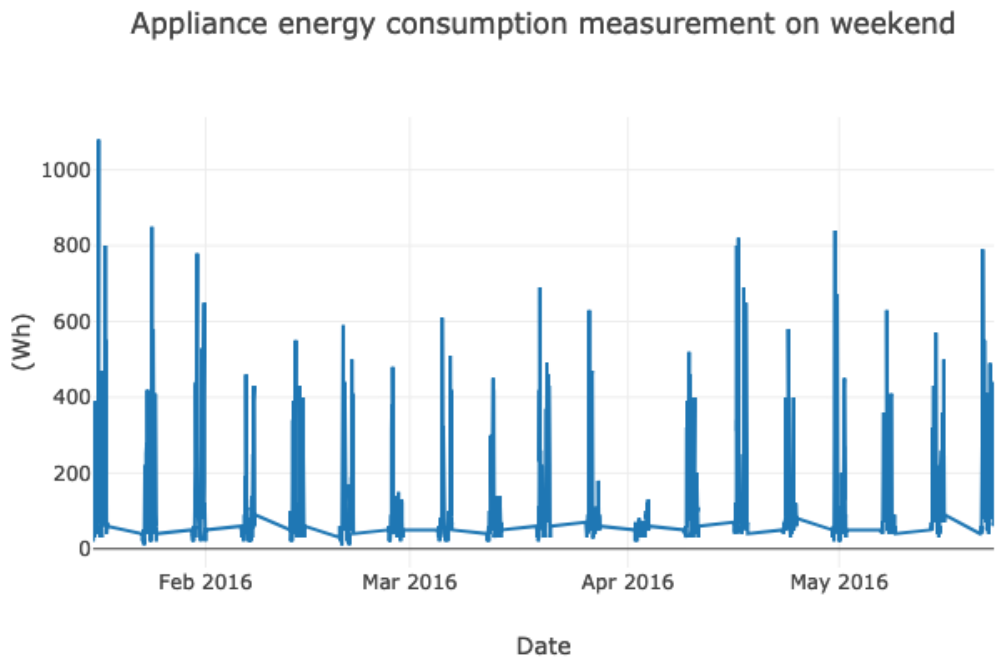
Feature ranges

1. Temperature : -6 to 30 deg
2. Humidity : 1 to 100 %
3. Windspeed : 0 to 14 m/s
4. Visibility : 1 to 66 km
5. Pressure : 729 to 772 mm Hg
6. Appliance Energy Usage : 10 to 1080 Wh

Data Visualization

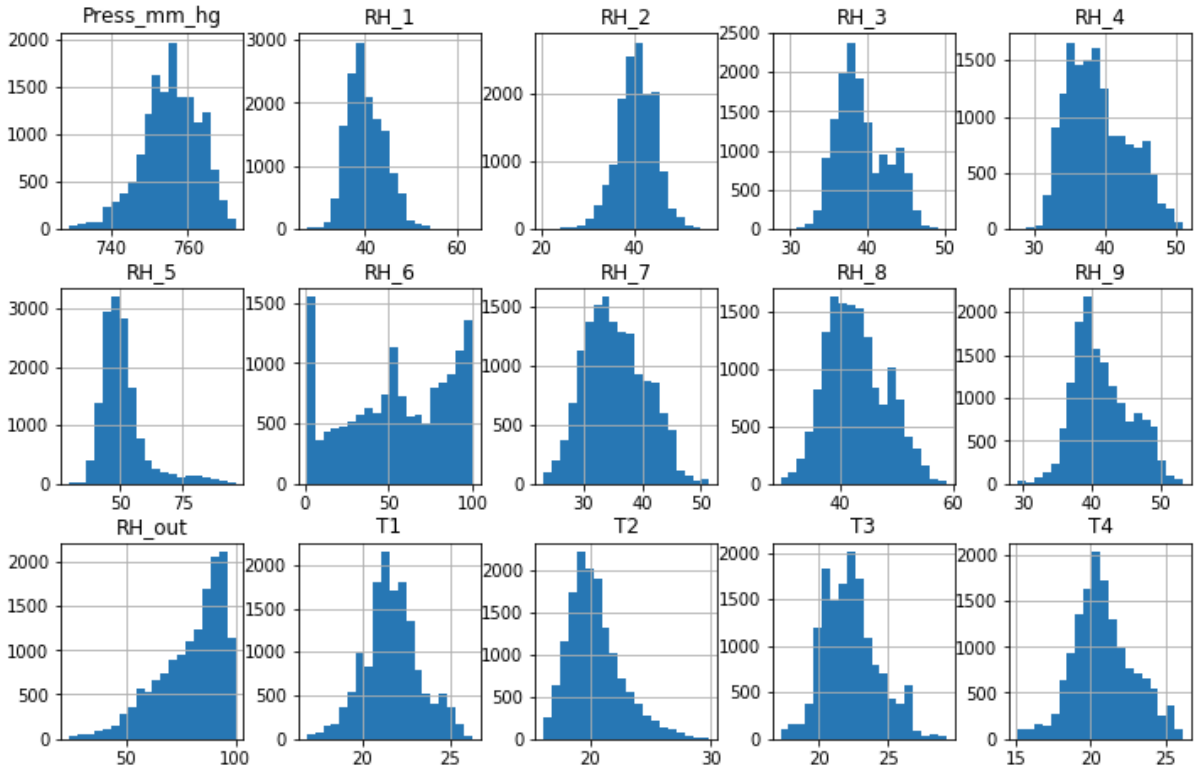


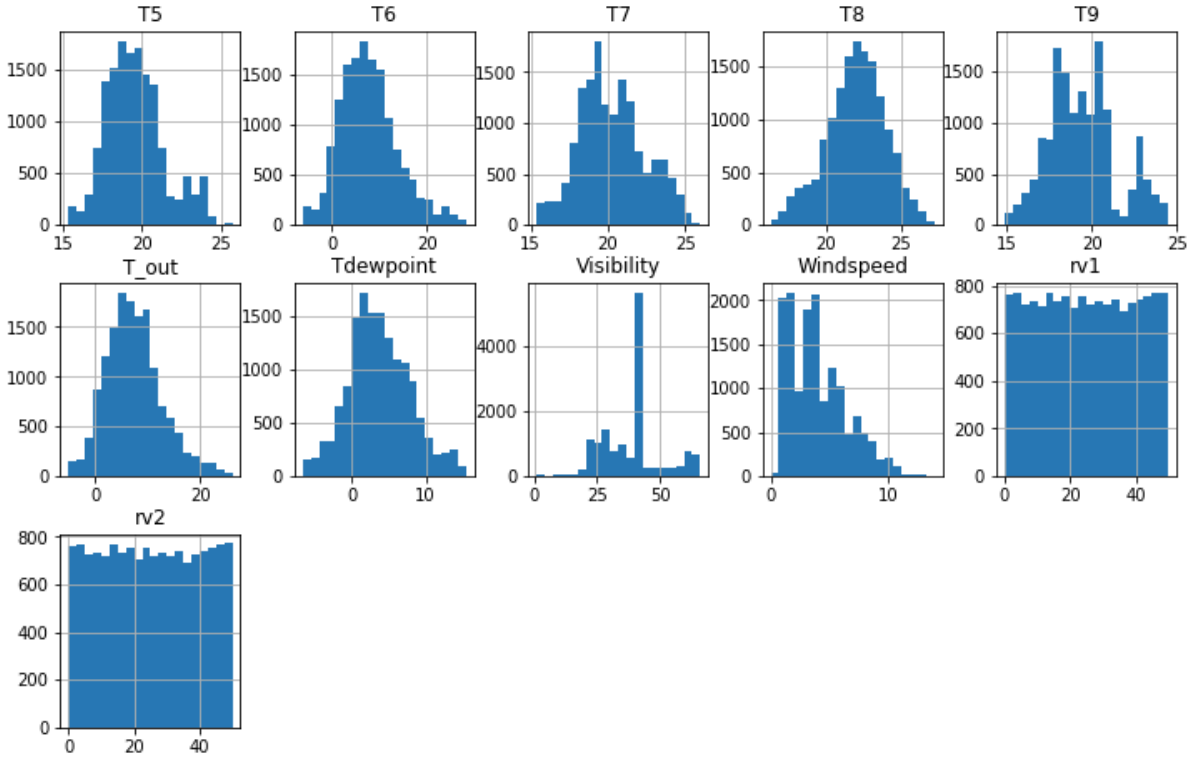




Independent Variable distribution

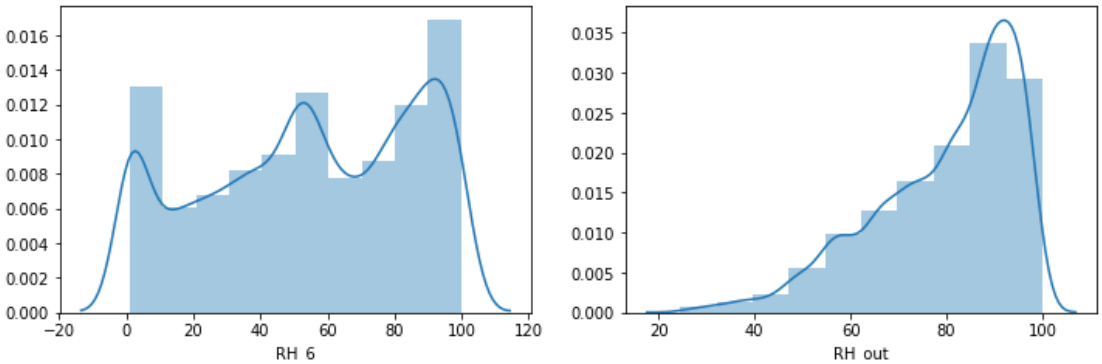
```
# Histogram of all the features to understand the distribution
feature_vars.hist(bins = 20 , figsize= (12,16)) ;
```

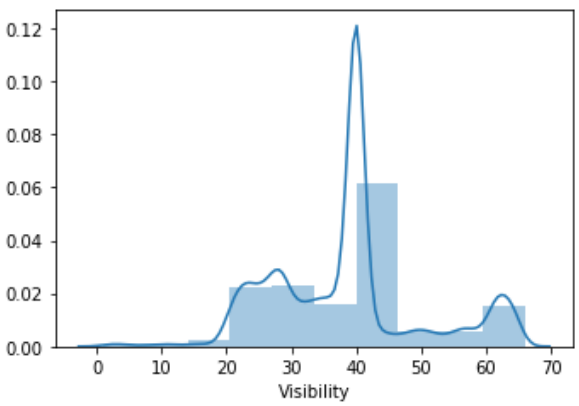


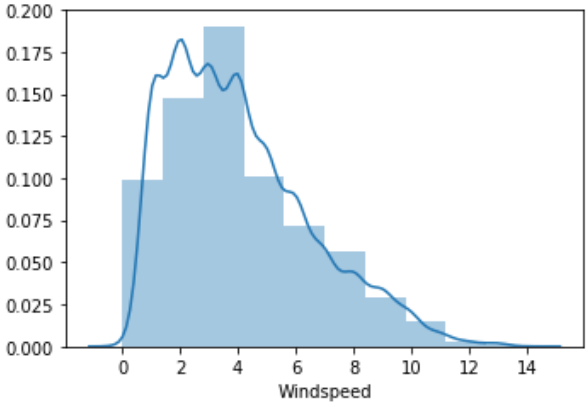


RH_6 , RH_out , Visibility , Windspeed irregular distribution

```
# focussed displots for RH_6 , RH_out , Visibility , Windspeed due to irregular distribution
f, ax = plt.subplots(2,2,figsize=(12,8))
vis1 = sns.distplot(feature_vars["RH_6"],bins=10, ax= ax[0][0])
vis2 = sns.distplot(feature_vars["RH_out"],bins=10, ax=ax[0][1])
vis3 = sns.distplot(feature_vars["Visibility"],bins=10, ax=ax[1][0])
vis4 = sns.distplot(feature_vars["Windspeed"],bins=10, ax=ax[1][1])
```

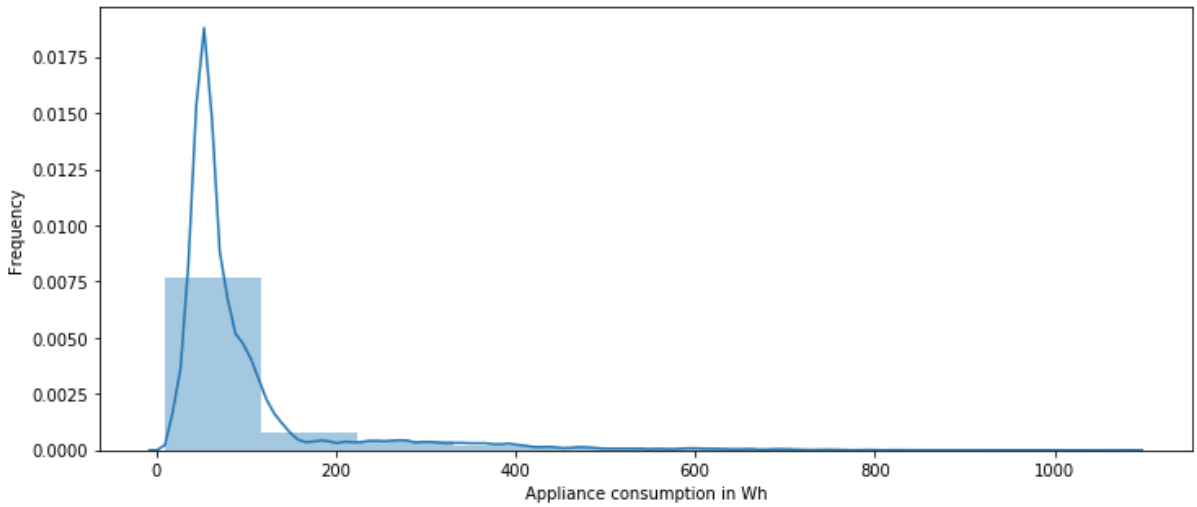






Dependent Variable distribution

```
# Distribution of values in Applainces column
f = plt.figure(figsize=(12,5))
plt.xlabel('Appliance consumption in Wh')
plt.ylabel('Frequency')
sns.distplot(target_vars , bins=10 ) ;
```



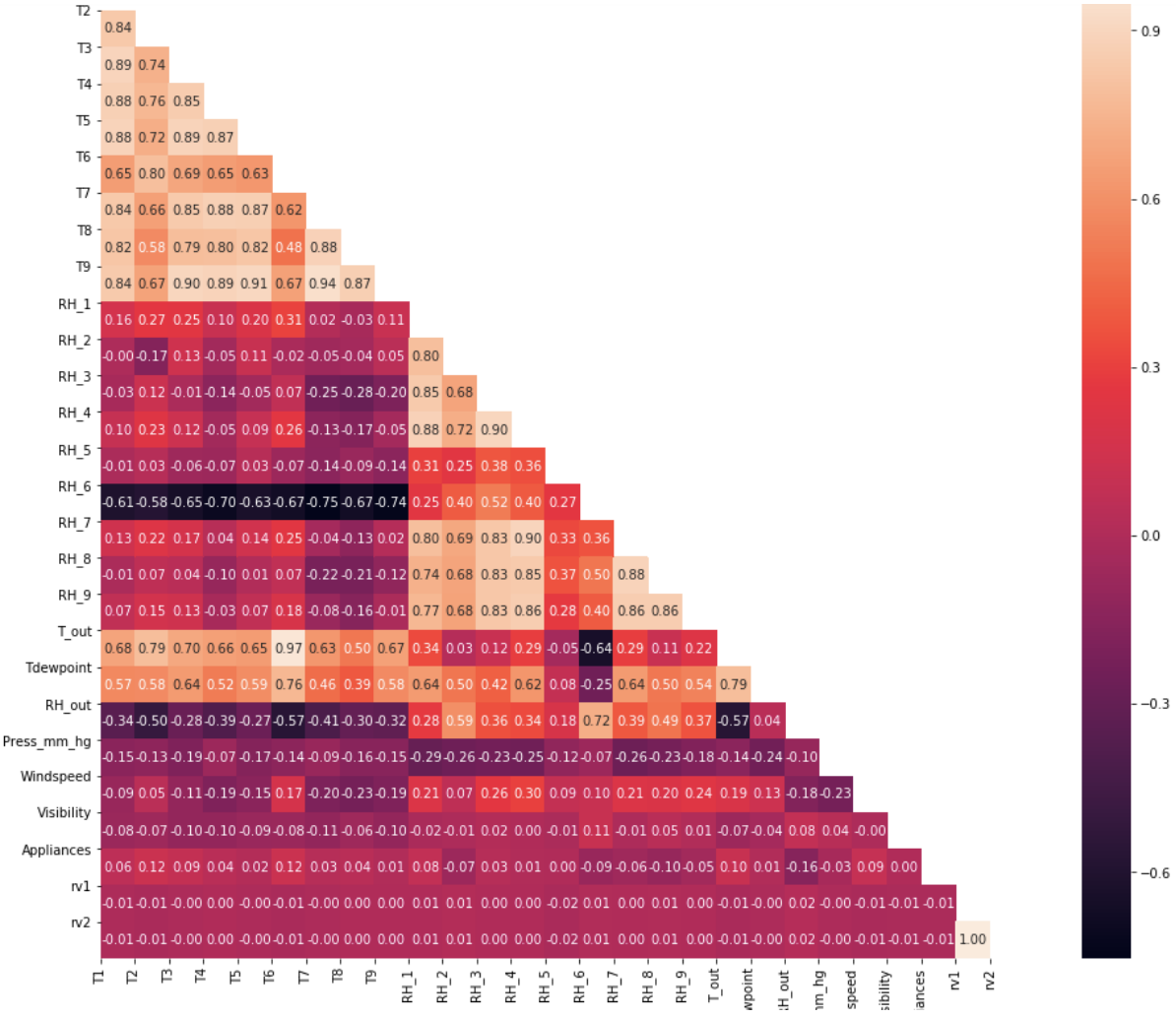
Observations based on distribution plot

1. All humidity values except RH_6 and RH_out follow a Normal distribution, i.e., all the readings from sensors inside the home are from a Normal distribution.
2. Similarly, all temperature readings follow a Normal distribution except for T9.
3. Out of the remaining columns, we can see that Visibility, Windspeed and Appliances are skewed.
4. The random variables rv1 and rv2 have more or less the same values for all the recordings.
5. The output variable Appliances has most values less than 200Wh, showing that high energy consumption cases are very low.

6. No column has a distribution like the target variable Appliances.

   Hence, there are no feature independent feature with a linear relationship with the target.

Co-relation plot



Observations based on correlation plot

1. Temperature - All the temperature variables from T1-T9 and T_out have positive correlation with the target Appliances . For the indoor temperatures, the correlations are high as expected, since the ventilation is driven by the HRV unit and minimizes air tempera-ture differences between rooms. Four columns have a high degree of correlation with T9

- T3,T5,T7,T8 also T6 & T_Out has high correlation (both temperatures from outside) . Hence T6 & T9 can be removed from training set as information provided by them can be provided by other fields.

2. Weather attributes - Visibility, Tdewpoint, Press_mm_hg have low correlation values
3. Humidity - There are no significantly high correlation cases (> 0.9) for humidity sensors.
4. Random variables have no role to play
5. The random variables rv1, rv2 and Visibility, Tdewpoint, Press_mm_hg have low correlation with the target variable.

Due to above conclusions , I have dropped rv1, rv2, Visibility, T6,T9.

Number of Input Variables - 21 (reduced from 26)

Modelling Techniques & Benchmarks

This is a Regression problem. Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (s) (predictor). The regression methods used are

1.Linear Models :

Linear Regression

In linear regression we wish to fit a function in this
Form $\hat{Y} = \beta0+\beta1X1+\beta2X2+\beta3X3$ where X is the vector of features and β0, β1 , β2, β3 are the coefficients we wish to learn. It updates β at every step by reducing the loss function as much as possible. As modification to Linear regression model, we can apply Regularization techniques to penalize the coefficient values of the features, since higher values generally tend towards overfitting and loss of generalization.

Ridge Regression

$$L = \sum(\hat{Y}_i - Y_i)^2 + \lambda \sum \beta^2$$

This loss function includes two elements. Sum of distances between each prediction and its ground truth. The second element sums over squared β values and multiplies it by another parameter λ. The reason for doing that is to “punish” the loss function for high values of the coefficients β. It enforces the βcoefficients to be lower, but it does not enforce them to be zero. That is, it will not get rid of irrelevant features but rather minimize their impact on the trained model.

Lasso Regression

$$L = \sum(\hat{Y}_i - Y_i)^2 + \lambda \sum |\beta|$$

The only difference from Ridge regression is that the regularization term is in absolute value. But this difference has a huge impact on the trade-off. Lasso method overcomes the disadvantage of Ridge regression by not only punishing high values of the coefficients β but actually setting them to zero if they are not relevant. Therefore, we might end up with fewer features included in the model than we started with, which is a huge advantage.

2.Support Vector Machine

Support vector regression

The Support Vector Regression (SVR) uses the same principles as the SVM for classification . In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem.

3.Nearest neighbour Regressor

KNeighborsRegressor

KNeighborsRegressor retrieve some k neighbors of query objects, and make predictions based on these neighbors . It computes the mean of the nearest neighbor labels.

4.Tree based Regression models

We divide the predictor space—that is, the set of possible values for X1, . . . , Xp—into J distinct and non-overlapping regions, R1, . . . , RJ . For every observation that falls into the region Rj , we make the same prediction, which is simply the mean of the response values for the training observations in Rj . Our goal is to find boxes R1, . . . , RJ that minimize the RSS given by

RSS = X J j=1 X i∈Rj (yi − yˆRj) 2 ,
where yˆRj is the mean response for the training observations within the jth box. Tree based models are less affected by outliers as compared to Linear models. Given there isn't a linear relation between any input and the target variable, so it is likely that Trees will work better than Linear models.

Ensemble methods

It combines several decision trees to produce better predictive performance than utilizing a single decision tree. The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner.

- **Bagging :** Bagging (Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree. Here idea is to create several subsets of data from training sample chosen randomly with replacement. Now, each collection of subset data is used to train their decision trees. Average of all the predictions from different trees are used which is more robust than a single decision tree.& Boosting

- **Boosting** : Boosting is another ensemble technique to create a collection of predictors. In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analyzing data for errors. In other words, we fit consecutive trees (random sample) and at every step, the goal is to solve for net error from the prior tree.

Random Forests

A Random Forest is an ensemble technique capable of performing both regression tasks with the use of multiple decision trees and a technique called bagging. and works well on high dimensional data

Gradient Boosting Machines

Gradient Boosting is an extension over boosting method. It uses gradient descent algorithm which can optimize any differentiable loss function. An ensemble of trees are built one by one and individual trees are summed sequentially. Next tree tries to recover the loss . Gradient Boosting= Gradient Descent + Boosting.

Extremely Randomized trees

The Extra-Trees algorithm builds an ensemble of unpruned decision or regression trees according to the classical top-down procedure. It splits nodes by choosing cut-points fully at random and that it uses the whole learning sample to grow the trees.

5.Neural Networks

A multilayer perceptron (MLP) is a deep, artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. MLPs with one hidden layer are capable of approximating any continuous function.

Benchmark

The benchmark is the R2 score of the Gradient Boosting technique used by the author in his original research paper. Following are the benchmark numbers

1. R2 score on training data: 57%
2. R2 score on test data: 97%
3. RMSE on training data = 17.56
4. RMSE on test data = 66.65

Data Preprocessing & Implementation

Data Scaling

The feature set has data in varying ranges . Temperature(-6 to 30) , Humidity (1-100) , Windspeed (0 to 14), Visibility (1 to 66) Pressure (729-772) and Application Energy Usage(10-1080). 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.

Implementation

Below mentioned scikit-learn & xgboost libaray's were used to test each regression model:

1. sklearn.linear_model.Ridge
2. sklearn.linear_model.Lasso
3. sklearn.ensemble.RandomForestRegressor
4. sklearn.ensemble.GradientBoostingRegressor
5. sklearn.ensemble.ExtraTreesRegressor
6. import xgboost
7. sklearn_neighbors
8. sklearn.svm.SVRPipeline
9. sklearn.neural_network. MLPRegressor

Pipeline :

1. Store all the algorithm's in a list and Iterate over the list
2. The regressor's random_state was initialized with a seed so that the results are the same every time other parameters were default.
3. the regressor was made to fit on the test & training data
4. The properties of the regressor , Name, timining & score for training and testing set were stored in a dictionary variable as key-value pairs.
5. The dictionary was appended to a global list of all dictionaries which is converted into dataframe

Result :

	Name	Test_R2_Score	Test_RMSE_Score	Train_R2_Score	Train_Time
0	Lasso:	0.000000	1.000000	0.000000	0.006955
1	Ridge:	0.121391	0.937341	0.137553	0.008538
2	KNeighborsRegressor:	0.485560	0.717245	0.681464	0.029766
3	SVR:	0.209934	0.888857	0.235724	9.764825
4	RandomForest	0.525681	0.688708	0.913691	2.994144
5	ExtraTreeRegressor :	0.577183	0.650244	1.000000	0.905229
6	GradientBoostingClassifier:	0.232897	0.875844	0.333526	1.999018
7	XGBRegressor:	0.226322	0.879590	0.322174	1.919970
8	MLPRegressor:	0.243178	0.869955	0.298567	1.427709

As observed from results, **ExtraTreesRegressor** performs better than all other regressors in terms of all metrics except for Training time

Feature Selection for Improvement

Extra Trees Regressor performed the best with default parameters. I used grid search cross validation using the GridSearchCV function of the sklearn.model_selection library. The parameters which were tuned :

1. n_estimators: The number of trees to be used
2. max_features: The number of features tob e considered at each split
3. max_depth : The maximum depth of the tree , If no param is provided then splitting will continue till all leaves are pure or contain less the min_samples_split specified

```
param_grid = [{
              'max_depth': [80, 150, 200,250],
              'n_estimators' : [100,150,200,250],
              'max_features': ["auto", "sqrt", "log2"]
            }]
```

The R2 score improved by 10% (0.57 to 0.63) post ususage parameters suggested by GridSearchCV

Challenges & Learning gained during project

1. Feature scaling is very important for regressions models , I initially tried without it and the results were not good . On Kaggle this is suggested by all users.
2. Using seed value helped in reproducing results for algortithms . Without this value the results were different each time.
3. It is very important to check the intercorrelation between all the variables in order to remove the redundant features with high correlation values.
4. While scaling data , it is useful to maintain separate copies of dataframe which can be created using index and column names of original dataframe
5. The pipeline of adding algorithms should be easy to manage
6. Seaborn and pyplot are good libraries to plot various properties of dataframe
7. For performing Exhaustive search or Random search in the hyperparameter space for tuning the model, always parallelize the process since there are a lot of models with different configurations to be fitted. (Set n_jobs parameter with the value -1 to utilize all CPUs)
8. One effective way to check the robustness of the model is to fit it on a reduced feature space in case of high dimensional data. Select the first ‘k’ (usually >= 3) key features for this task.

Results

Model Evaluation & Valuation

Features of the untuned model:

1. n_estimators : 10
2. max_features : auto
3. max_depth : None

Features of best model after hyper parameter tuning:

1. n_estimators : 200
2. max_features: 'sqrt'
3. max_depth: 80

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

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.

- R2 score on test data model with reduced features = 0.47.
- R2 score on test data for untuned model = 0.57.
- Difference = 0.10

- RMSE score on test data model with reduced features = 0.72
- RMSE score on test data for untuned model = 0.65
- Difference = 0.07

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.

Benchmark comparison

Parameters/Models	Final Model	Benchmark Model
Training R2 score	**1.0**	**0.97**
Testing R2 score	**0.63**	**0.57**
RMSE on test data	**0.60**	**0.66**

There has improvement of 10.53% in the R2 score for testing set , with more data and feature engineering this can be improved further.

Conclusions

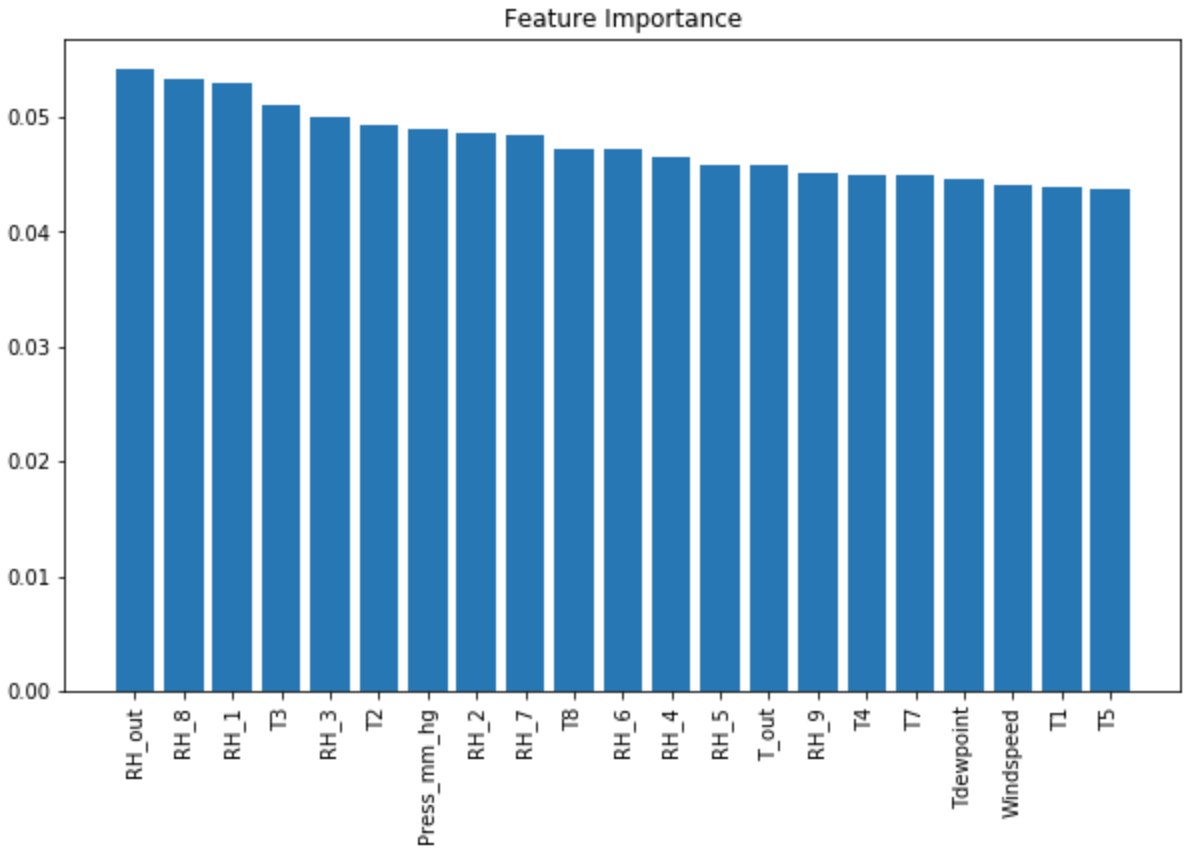
According to best fit model , the 5 most and least important features

```
# Get top 5 most important feature
names[0:5]
```

```
['RH_out', 'RH_8', 'RH_1', 'T3', 'RH_3']
```

```
# Get 5 least important feature
names[-5:]
```

```
['T7', 'Tdewpoint', 'Windspeed', 'T1', 'T5']
```



The top 3 important features are humidity attributes, which leads to the conclusion that humidity affects power consumption more than temperature. Windspeed is least important as the speed of wind doesn't affect power consumption inside the house. So controlling humidity inside the house may lead to energy savings.

Reflections

This project can be summarized as

1. Looking for Energy related dataset on UCI Machine Learning repository and Kaggle where in benchmark numbers are availiable
2. Deciding between Classification and Regression problems.
3. Visualized the data, did preprocessing by learning from other regression contests from Kaggle.
4. Preprocessing the data and feature selection.Look for correlation between features
5. Deciding the regression algorithms to be used to solve the problem.
6. Using GridSearchCV instead of RandomizedSearchCV to create benchmark model.
7. Applying selected algorithms and visualizing the results.
8. Hyper parameter tuning for the best algorithm and reporting the test score of best model.
9. Discuss importance of selected features and check the robustness of model..
10. Comparing my tuned model against the author's benchmark result .

Improvements

1. Perform aggressive feature engineering
2. Look for classification scenario in the dataset and explore the problems
3. Modifying parameters of the Grid Search parameter space.
    a. Add more parameters like min_samples_split, min_inpurity_decrease etc.
    b. max_depth can have more values

Refernces:

1. http://dx.doi.org/10.1016/j.enbuild.2017.01.083
2. https://github.com/LuisM78/Appliances-energy-prediction-data
3. http://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction
4. http://scikit-learn.org/stable/supervised_learning.html#supervised-learning