Capstone Project

Machine Learning Engineer Nanodegree

APPLIANCE ENERGY PREDICTION

1. Definition

a. Project Overview

Yet many places in the world are still affected by power outages. One significant cause of power outages is overloading. Overloading occurs when too much power is drawn from an electric circuit at once. This can be avoided if we know when excess electricity is going to be used. Heating and cooling appliances consume the most power in a household. The goal is to predict the electricity usage of heating and cooling appliances in a household based on internal and external temperatures and other weather conditions.

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

The problem at hand is to predict the electricity usage of heating and cooling appliances in a household based on internal and external temperatures and other weather conditions and 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 R2 (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:

$$SQ^2 = 1-\frac{SSres}{SStot}$$

where,
SSres = Residual sum of squares
SStot = Total sum of squares
For this project, I will use 'r2_score()' function of the metrics module of scikit-learn library While "Coefficient of Determination" provides relative a measure of the how well the model fits the data, the RMSE (Root Mean Squared Error) gives absolute measure of how well model fits the data i.e. how close are the predicted values to the actual values.

Mathematically, RMSE can be defined as:

RMSE = $\sqrt{\frac{!}{N} * \sum_{i=1}^{N}(\text{yi} - \text{yi}\text{^})2}$

Where,
N = number of observations
yi = Actual value of target variable
$\text{y}\text{^}$ = Predicted value of target variable
In this project, I will calculate RMSE by calculating square root of mean_squared_error() function provided in the metrics module of scikit-learn library.
Therefore, the metrics to be used are:

i. R2 score
ii. ii. RMSE

These two metrics are helpful for this problem because of the following reasons:

i. It is a Regression based problem.
ii. R2 score will show the statistical robustness of the model.
ii. RMSE will give an idea about how accurate the predictions are to actual values.

2. Analysis

a. Data Exploration

The dataset was collected by sensors placed inside the house and outside readings came from the nearby weather station. The main types of attributes are temperature readings, humidity and pressure. Each observation measures electricity in a 10-minute interval. The temperatures and humidity have been averaged for 10-minute intervals. Number of input attributes : 28 (11 temperature, 10 humidity, 1 pressure, 2 randoms,etc.)

Target variable: 1 (Appliances)
Attribute Information:

- date : time year-month-day hour:minute:second
- lights : energy use of light fixtures in the house in Wh
- T1 : Temperature in kitchen area, in Celsius
- T2 : Temperature in living room area, in Celsius
- T3 : Temperature in laundry room area
- T4 : Temperature in office room, in Celsius

- T5 : Temperature in bathroom, in Celsius
- T6 : Temperature outside the building (north side), in Celsius
- T7 : Temperature in ironing room, in Celsius
- T8 : Temperature in teenager room 2, in Celsius
- T9 : Temperature in parents' room, in Celsius
- T_out : Temperature outside (from Chievres weather station), in Celsius
- Tdewpoint : (from Chievres weather station), Â°C
- RH_1 : Humidity in kitchen area, in %
- RH_2 : Humidity in living room area, in %
- RH_3 : Humidity in laundry room area, in %
- RH_4 : Humidity in office room, in %
- RH_5 : Humidity in bathroom, in %
- RH_6 : Humidity outside the building (north side), in %
- RH_7 : Humidity in ironing room, in %
- RH_8 : Humidity in teenager room 2, in %
- RH_9 : Humidity in parents' room, in %
- RH_out : Humidity outside (from Chievres weather station), in %
- Pressure : (from Chievres weather station), in mm Hg
- Wind speed :(from Chievres weather station), in m/s
- Visibility :(from Chievres weather station), in km
- Rv1 : Random variable 1, non-dimensional
- Rv2 : Random variable 2, non-dimensional

Target Variable:

- Appliances: Total energy used by appliances, in Wh

I didn't use the following features as they were not relevant to the problem.

- Date : As the problem is regression not time-series, date of the record does not matter.
- Lights : The goal is to predict overall energy use and not category-wise.

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
Count of Null values = 0
All features have numerical values. There are no categorical or ordinal features in this dataset.

Descriptive statistics:

i. Range of 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	21.691343	20.344518	22.278802	20.860393	19.604773	7.923216	20.273236	22.028122	19.493479
std	1.615790	2.202481	2.012934	2.048076	1.849641	6.117495	2.118416	1.960985	2.022560
min	16.790000	16.100000	17.200000	15.100000	15.340000	-6.065000	15.390000	16.306667	14.890000
25%	20.760000	18.790000	20.790000	19.533333	18.290000	3.626667	18.700000	20.790000	18.000000
50%	21.600000	20.000000	22.100000	20.666667	19.390000	7.300000	20.075000	22.111111	19.390000
75%	22.633333	21.500000	23.340000	22.100000	20.653889	11.226667	21.600000	23.390000	20.600000
max	26.260000	29.856667	29.236000	26.200000	25.795000	28.290000	26.000000	27.230000	24.500000

	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	40.267556	40.434363	39.243995	39.043799	51.014065	54.615000	35.410874	42.948244	41.556594
std	3.974692	4.052420	3.245701	4.333479	9.107390	31.160835	5.097243	5.210450	4.161295
min	27.023333	20.596667	28.766667	27.660000	29.815000	1.000000	23.260000	29.600000	29.166667
25%	37.363333	37.900000	36.900000	35.560000	45.433333	29.996667	31.500000	39.096667	38.530000
50%	39.693333	40.500000	38.560000	38.433333	49.096000	55.267500	34.900000	42.390000	40.900000
75%	43.066667	43.273453	41.730000	42.200000	53.773333	83.226667	39.000000	46.500000	44.326667
max	63.360000	54.766667	50.163333	51.090000	96.321667	99.900000	51.327778	58.780000	53.326667

	T_out	Tdewpoint	RH_out	Press_mm_hg	Windspeed	Visibility
count	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000
mean	7.421836	3.782509	79.824197	755.480135	4.029001	38.290284
std	5.343737	4.194994	14.901776	7.389218	2.448171	11.789650
min	-5.000000	-6.600000	24.000000	729.300000	0.000000	1.000000
25%	3.666667	0.933333	70.500000	750.900000	2.000000	29.000000
50%	6.933333	3.483333	83.833333	756.000000	3.666667	40.000000
75%	10.433333	6.600000	91.666667	760.833333	5.500000	40.000000
max	26.100000	15.316667	100.000000	772.300000	14.000000	66.000000

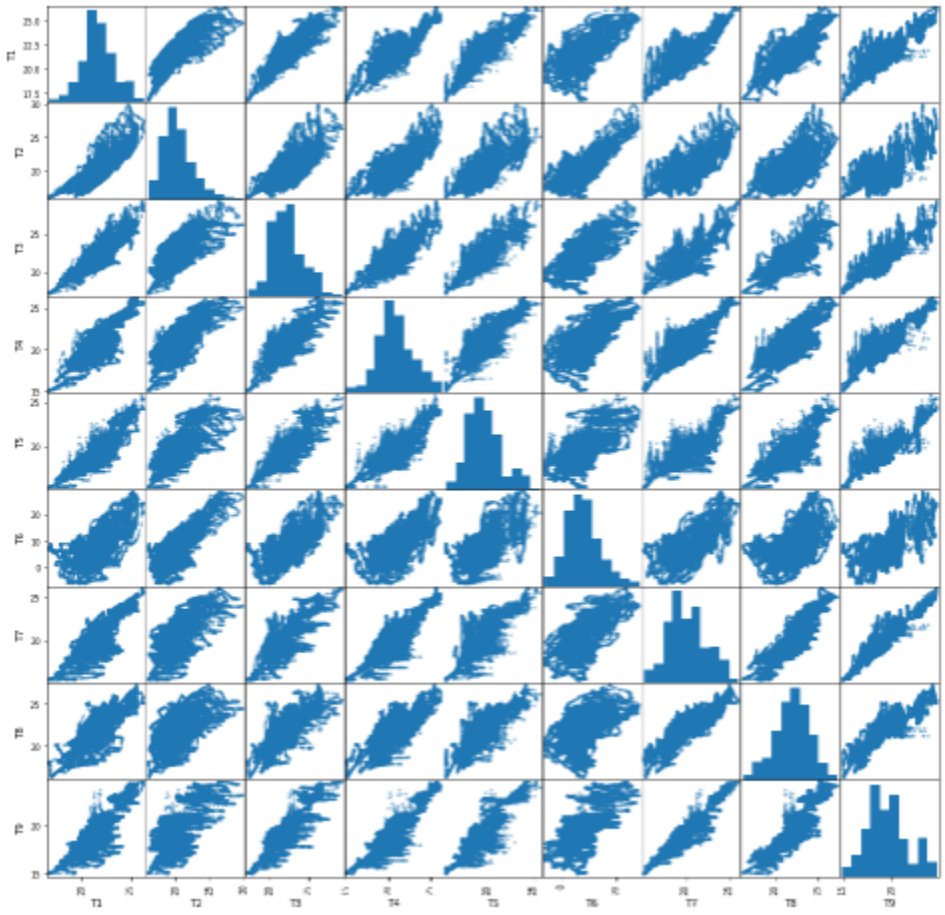
	Appliances
count	14801.000000
mean	97.875144
std	102.314986
min	10.000000
25%	50.000000
50%	60.000000
75%	100.000000
max	1080.000000

Observations

- Temperature ranges for all home sensors is between 14.89℃ to 29.86℃ except for T6 for which it is -6.06℃ to 28.29℃. The reason for such low readings is that the sensor is kept outside.
- Similarly, humudity ranges for all home sensors is between 20.60% to 63.36%. Except for RH_5 and RH_6, whose ranges are 29.82% to 96.32% and 1% to 99.9% respectively.
    - The reason behind this is that RH_5 is inside the bathroom,
    - And RH_6 is outside the building, explaining the high humidity values.
- One interesting observation can be seen in `Appliances` column that although the max consumption is 1080Wh, 75% of values are less than 100Wh. This shows that there are fewer cases when Appliance energy consumption is very high.

ii. Scatter plots:

- T1 to T9:

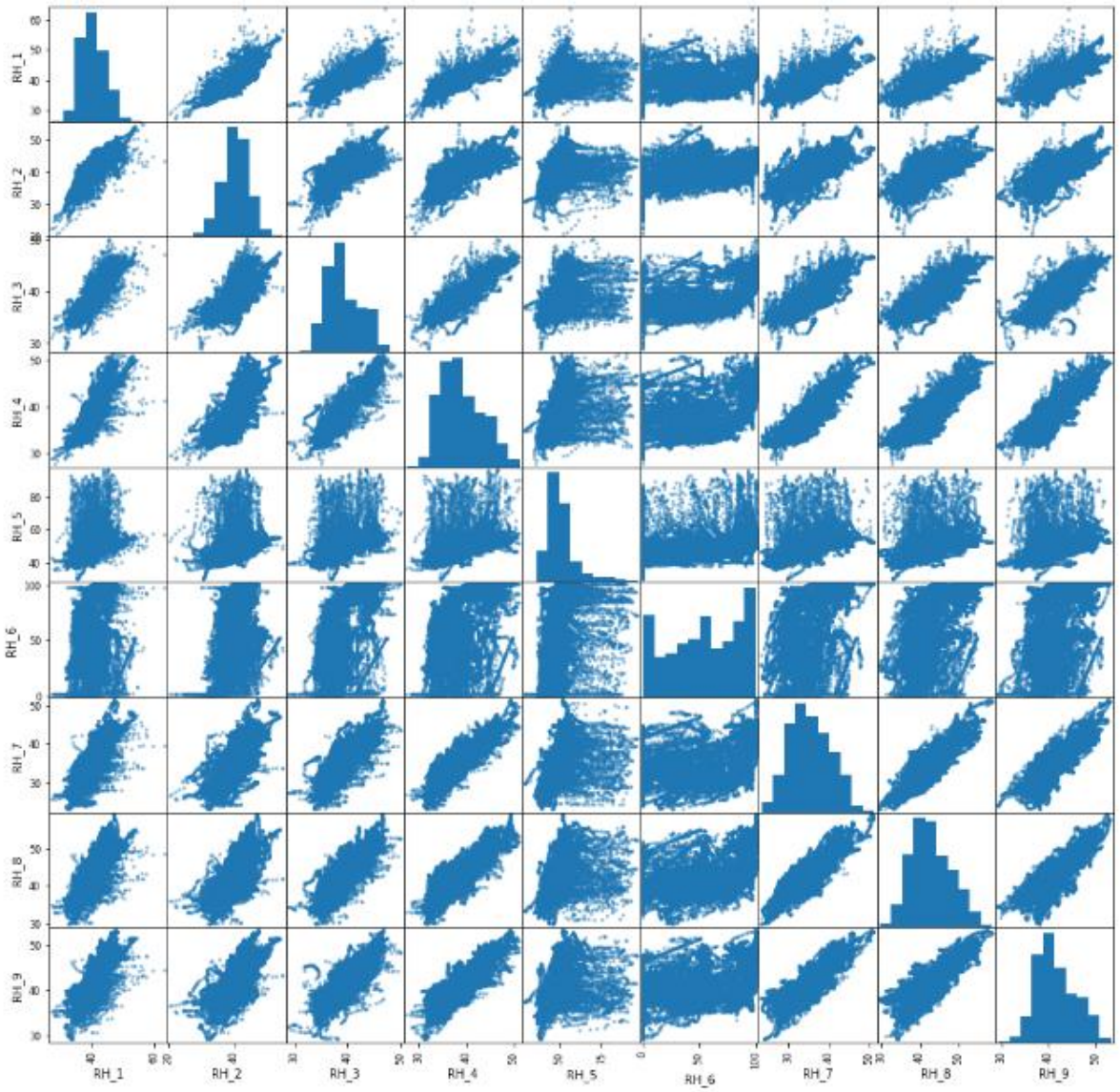


Some degree of correlation can be seen between T7 and T9. This can be confirmed by computing their Pearson coefficient which turns out as follows:

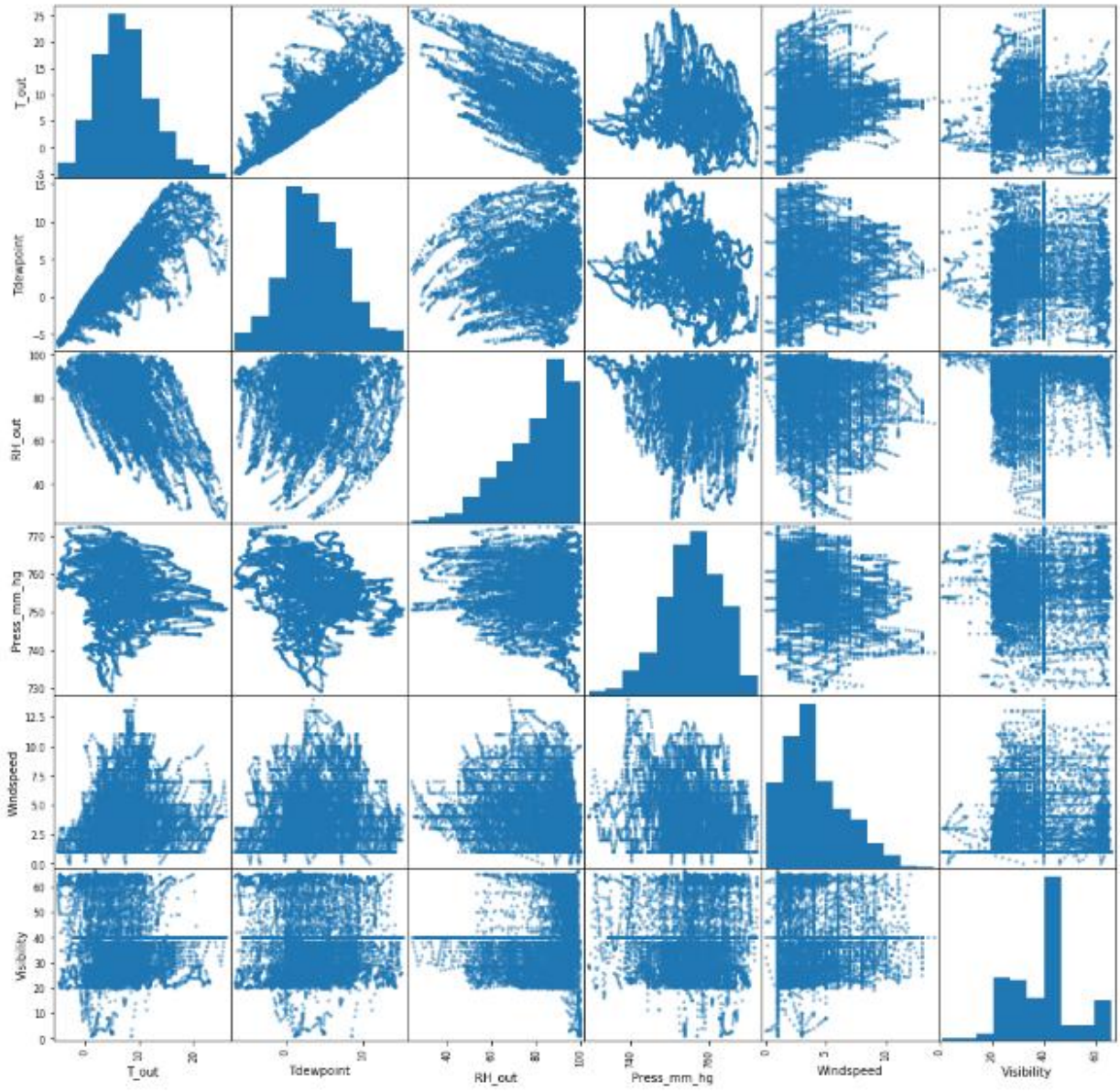
In [18]:
```
# Import pearson relation method from SciPy
from scipy.stats import pearsonr

# Calculate the coefficient and p-value
corr_coef, p_val = pearsonr(energy["T7"], energy["T9"])
print("Correlation coefficient : {}".format(corr_coef))
print("p-value : {}".format(p_val))
```

```
Correlation coefficient : 0.9460586115166221
p-value : 0.0
```

- RH_1 to RH_9:

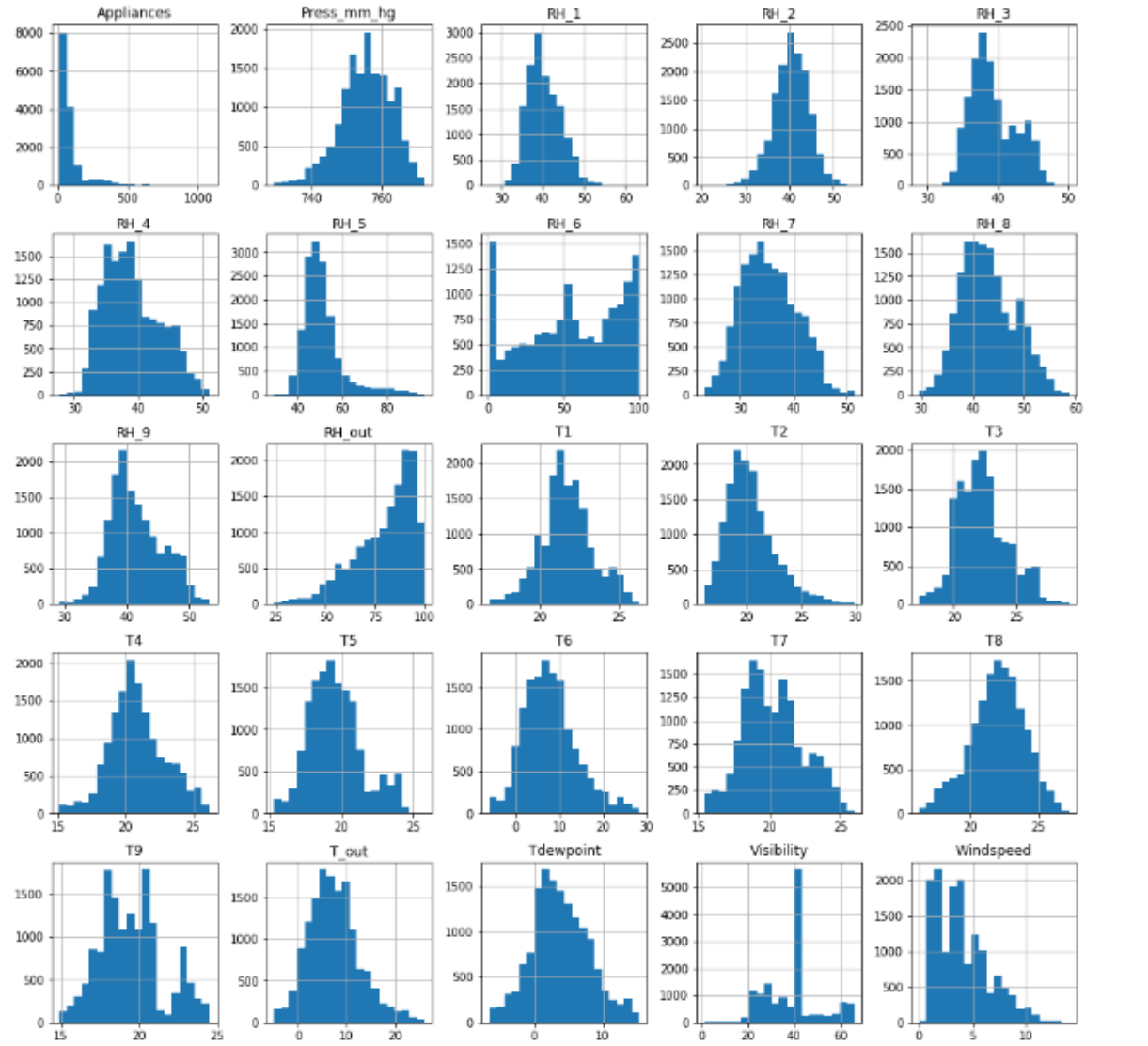


- c. Weather data:



No significant correlation exists among different humidity values and weather among weather parameters like Pressure, Windspeed, Temperature, etc. which can be confirmed from the the above last two plots.

iii. Distribution of all the columns:

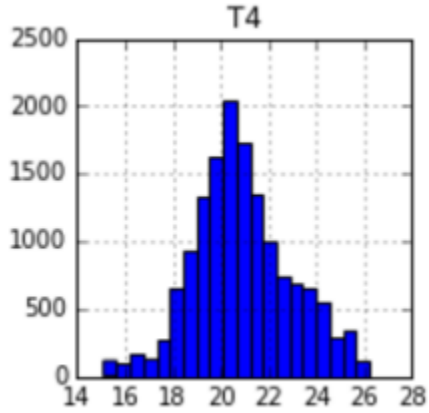


From this plot, it can be concluded that no columns have a distribution like the Appliances column, which is our target variable. Therefore, we can deny a linear relationship of any single feature independently with the target variable.

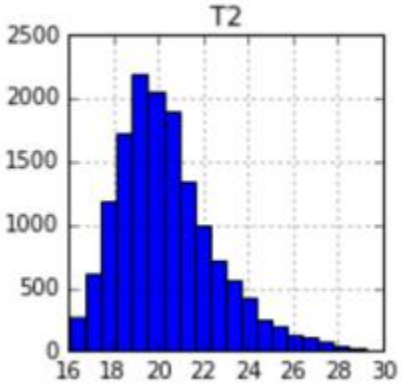
b. Exploratory Visualization:

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

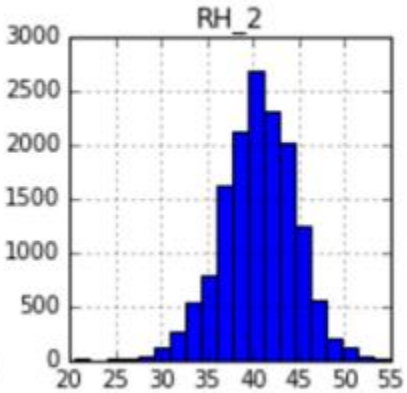
ii. Out of which, some features are skewed left/right as shown below:



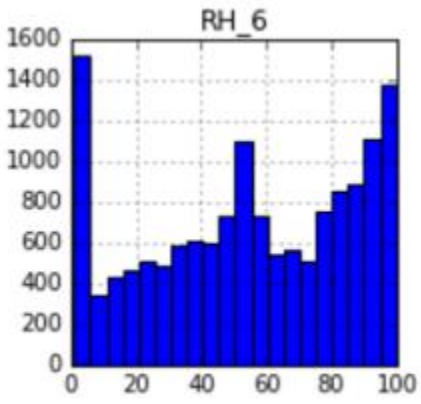
Left

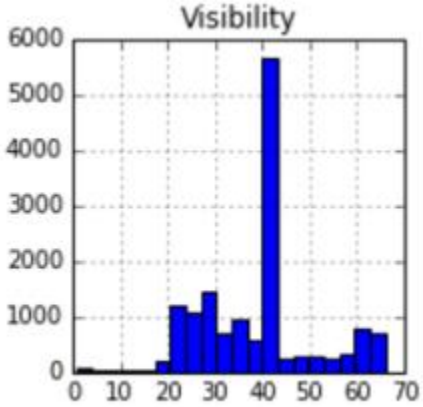


Right

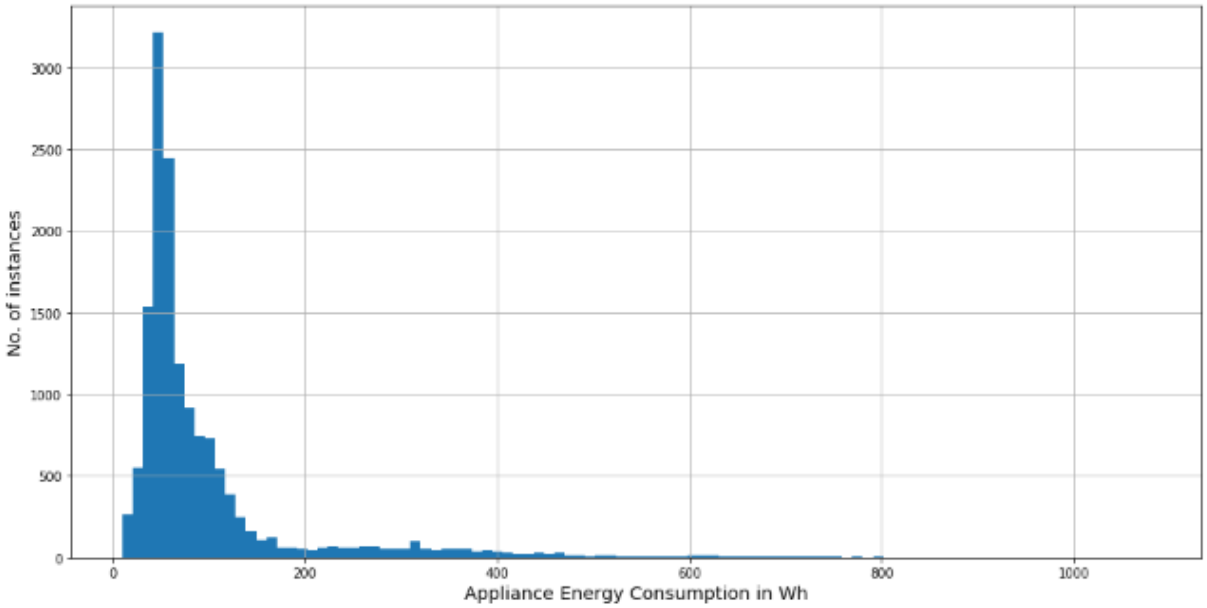


iii. Some features don't have normal distribution as shown below:





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:

The most basic Regression algorithm is Linear Regression. If a Linear model can explain the data well, there is no need for further complexity. As modification to original Least Squares Regression, we can apply Regularization techniques to penalize the coefficient values of the features, since higher values generally tend towards overfitting and loss of generalization. Regularization techniques enhance performance of Linear models greatly. Also, there very few practical cases when a Linear model can fit the data well without Regularization.

In case of Regularization, depending upon whether we add the absolute values of coefficients or their squares to our loss function, the problem of Linear Regression is transformed into Lasso or Ridge Regression respectively.

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

The next category of algorithms is of Tree based Regression models. An important advantage of Tree based models is that they are robust to outliers compared to Linear models. We haven't seen that a Linear relationship between any feature and the target variable, it is likely that Regression tress will turn out to be better than Linear models.

Given the substantial number of features, it is evident that a Decision Tree will overfit the data. Hence, I have skipped it and directly jumped towards ensemble methods listed below, which include building multiple regressors on copies of same training data and combining their output either through mean, median, mode (Bagging) or growing trees sequentially (i.e. each tree is built from data of the previous tree) and using weighted average of these weak learners (a learner which performs just a little better than chance (50%)) (Boosting).

Random Forests is one of the primary Bagging methods and works well on high dimensional data like ours. Extreme Trees Regression goes one step further by making splits Random. Gradient Boosting Machines is a type of Boosting method. It builds an additive model in a way that performance always increases.

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

Finally, one of the primary algorithms for non-linear hypothesis is a neural network. Neural networks work great when there is a complex nonlinear relationship between the inputs and the output. Although they generally have superior performance, one of their downside is that they take very long time to train. I will be using a Multi-Layer Perceptron as my choice of Neural network. The error function is squared

iii. Neural Networks
    1. Multi-layer Perceptron

d. Benchmark

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

i. R2 score on training data: 14.687%
ii. R2 score on test data: 14.258%
ii. RMSE on test data = 0.926 (For calculating RMSE, the data was scaled so that comparison with other models is easier)
iii. Time taken to fit: 0.032 seconds

3. Methodology

a. Data Preprocessing

The variables present in the data have varying ranges, with some having very low ranges like Windspeed (0 to 14), while others like Pressure have high range (729-772). So, the variables need to be scaled else some features may dominate the result. I scaled all the input variables to mean 0 and 1 variance using StandardScaler in scikit-learn's preprocessing module. Also I had removed certain variables based on importance and redundancy as explained above. As a result the data has 21 input attributes.

b. Implementation

The model implementation is done in 3 steps:

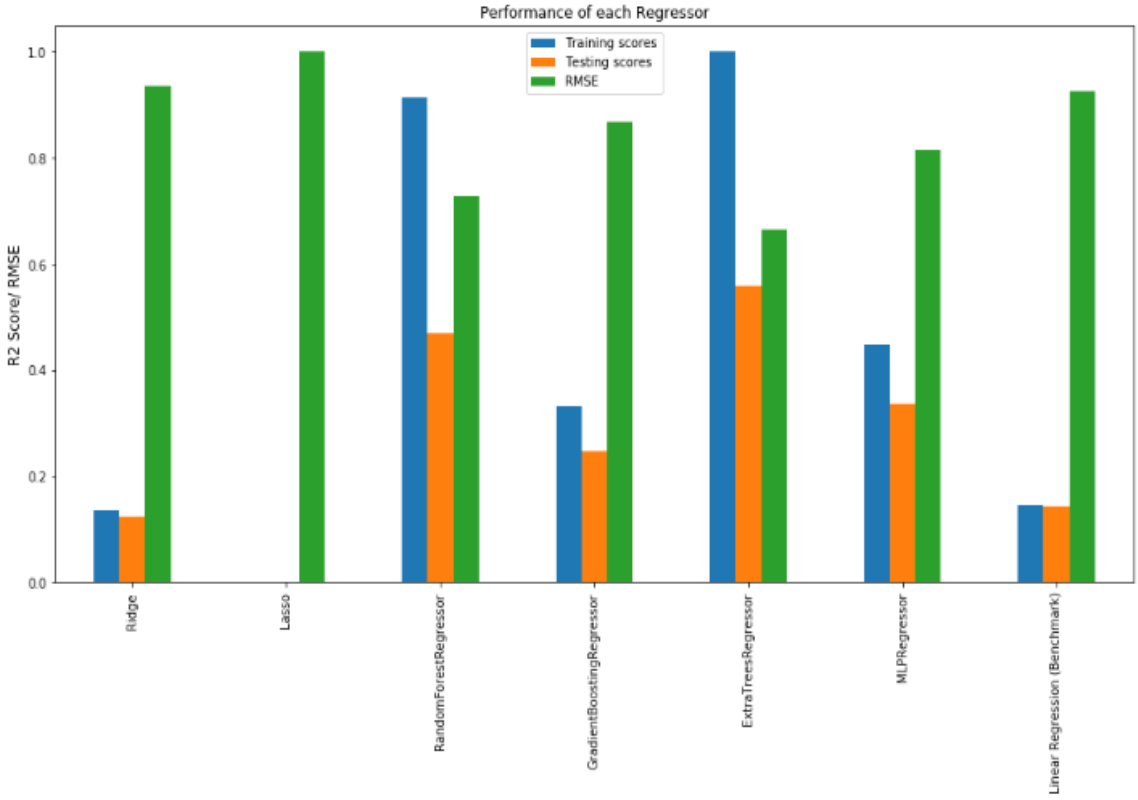
i. Create a pipeline() function to execute each Regressor and record the metrics.
ii. Pass each Regressor to above pipeline function from execute_pipeline().
iii. Consolidate the obtained metrics into a DataFrame in the function get_properties() and plot these metrics using a bar graph.

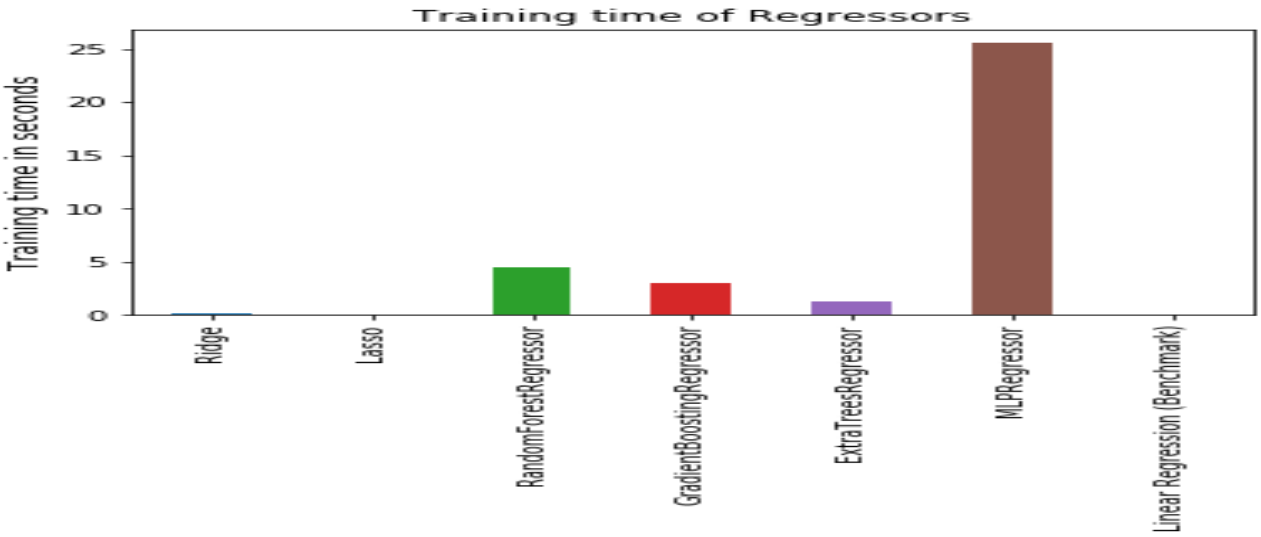
List of Algorithms tested:

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. RMSE will be calculated by taking square root of MSE value calculated using mean_square_error() function.

Results:





As observed from results, ExtraTreesRegressor performs better than all other regressors in terms of all metrics except for Training time where Linear models outperform it. Even then, it's training time is less than all the 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:

```
# Define the parameter subset
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%.`

A summary of challenges faced an overcame:

1. Always check for correlated features in a high dimensional dataset, and remove redundant features with high correlation.
2. Feature scaling is a must for Regression.
3. Use a seed generator for reproducible results.
4. If you want to maintain separate copies of DataFrames with scaled data, it is viable to create dummies using original DataFrame's index and columns and then filling it with scaled data.
5. The pipeline should be as modular as possible. For example, I can easily add an algorithm to the list of algorithms to be tested in the execute_pipeline() function without changing the model implementation function pipeline().
6. It is easier to plot various properties of the models if they are consolidated into a DataFrame rather than storing and manually plotting them individually.
7. Cross validation is very useful for finding out the best model.
8. 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)

9. 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.

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.

R2 score on test data = 0.499.
R2 score of untuned model = 0.558.
Difference = 0.059 or 5.9%.

RMSE on test data = 0.708
RMSE of untuned model = 0.665
Difference = 0.343

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

Parameters/Models	Final Model	Benchmark Model	Difference
Training R2 score	1.0	0.147	0.853
Testing R2 score	0.61	0.142	0.468
RMSE on test data	0.624	0.926	0.302

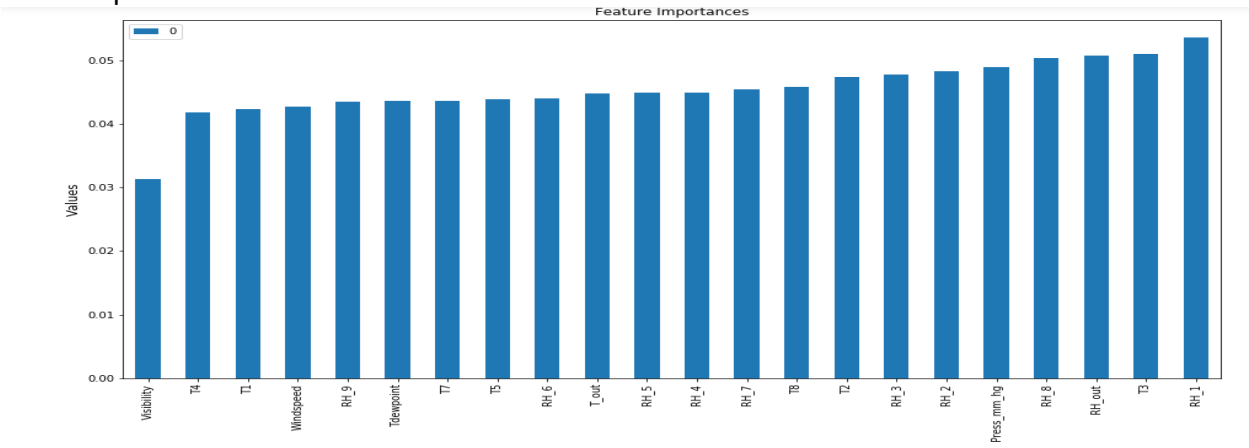
Based on the improvements recorded above, the final tuned model can be deemed as a satisfactory solution.

5. Conclusion

a. Free-form visualization

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

Top 5 most features :-
RH_1
T3
RH_out
RH_8
Press_mm_hg

Top 5 least 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 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.