Appliances Energy Prediction

Almabetter Capstone Project

Team: AETHER

Ankur Bhattacharjee

Bhabakrishna Talukdar

Mayank Tiwari

Md Suhel Ansari

Pratheek T M

Dated: 26/03/2022

Outline:

Every Household today has a range of Appliances for various daily needs. The amount of energy that a household consumes depends highly on these Appliances. So it’s vital to understand these patterns of energy consumption and come up with predictions for future scenarios.

To produce these predictions, we require a variety of data related to the region where the Household is situated. The examples for such data can be temperature in that region, humidity in that region, atmospheric pressure in that region, visibility, wind speed etcetera.

With these data have been provided for a given amount of time, to make such predictions for future scenarios, we have tried to capture some energy consumption patterns and tried to optimise our Machine Learning models for accurate predictions.

Problem Statement:

- Performing some Exploratory Analysis on our dataset.
- Preparing our dataset for Machine Learning models by removing most of the outliers and scaling our features and target variables.
- Selection of suitable models for further improvements on such models.
- Hyper-tuning selected models for improved predictions.
- Analysing the performance using some Evaluation metrics.
- Combining top performing models using Stacking to check for improvements in our predictions.

Introduction:

Appliances Energy Prediction of a household is an approach to predict the amount of energy that will be consumed by that household in the future, with respect to existing physical conditions inside the house and of the locality that house belongs to at a given interval of time.

The data given has information about the amount of energy consumed(in watt-hour) and the existing physical conditions such as temperature and humidity both inside and outside the house and pressure, windspeed, visibility and Tdewpoint in the area(from a local weather station), which are recorded at every 10 minutes for about 4.5 months.

As we can see the dependent variable ‘Appliance’(energy consumed in watt-hour) is continuous and so we aim to use Regression algorithms to predict the amount of energy consumed. But first, we attempt to clean the data and perform Exploratory Data Analysis on different variables to understand the relationship among independent variables and between dependent and independent variables. Then we remove outliers and we will scale the data to complete the data pre-processing for finally feeding it to the Regression Algorithms.

After this we will tune the Hyper-parameters using some cross validation techniques and we will attempt stacking to workout different combinations of algorithms aiming to improve the accuracy of predictions. At the end we will perform feature selection and conclude which one yields better results.

Data Description:

The data set is at 10 min for about 4.5 months. The house temperature and humidity conditions were monitored with a ZigBee wireless sensor network. Each wireless node transmitted the temperature and humidity conditions around 3.3 min. Then, the wireless data was averaged for 10 minutes periods. The energy data was logged every 10 minutes with m-bus energy meters.

Weather from the nearest airport weather station (Chievres Airport, Belgium) was downloaded from a public data set from Reliable Prognosis (rp5.ru) and merged together with the experimental data sets using the date and time column. Two random variables have been included in the data set for testing the regression models and to filter out non-predictive attributes(parameters).

- **date time: year-month-day hour : minute : second**
- **Appliances: energy use in Wh (Dependent variable)**
- **lights, energy use of light fixtures in the house in Wh (Drop this column)**
- **T1: Temperature in kitchen area in Celsius**
- **RH1: Humidity in kitchen area in %**
- **T2: Temperature in living room area in Celsius**
- **RH2: Humidity in living room area in %**
- **T3: Temperature in laundry room area in Celsius**
- **RH3: Humidity in laundry room area in %**
- **T4: Temperature in office room in Celsius**
- **RH4: Humidity in office room in %**
- **T5: Temperature in bathroom in Celsius**
- **RH5: Humidity in bathroom in %**
- **T6: Temperature outside the building (north side) in Celsius**
- **RH6: Humidity outside the building (north side) in %**
- **T7: Temperature in ironing room in Celsius**

- **RH7: Humidity in ironing room in %**
- **T8: Temperature in teenager room 2 in Celsius**
- **RH8: Humidity in teenager room 2 in %**
- **T9: Temperature in parents room in Celsius**
- **RH9: Humidity in parents room in %**
- **To: Temperature outside (from Chievres weather station) in Celsius**
- **Pressure (from Chievres weather station) in mm Hg**
- **RHout: 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**
- **rv2: Random variable 2**

Where indicated, hourly data (then interpolated) from the nearest airport weather station (Chievres Airport, Belgium) was downloaded from a public data set from Reliable Prognosis, rp5.ru. Permission was obtained from Reliable Prognosis for the distribution of the 4.5 months of weather data.

Steps Involved:

Step 1 - Data Exploration and Cleaning:

- Checking the initial information of the dataset, such as number of rows and columns, data-types of columns and checking for null-values.
- Renaming columns in accordance with the data description provided.
- Checking the description of numerical features, to have an idea about their distributions.
- Transforming 'date' column to 'date-time' format, and dividing them into columns 'Weekday', 'Month', 'Hour' and 'Day'.
- Dropping few columns such as, 'date', 'rv2' and 'light' since they're not essential.

Step 2 - Exploratory Data Analysis:

- Performing Univariate analysis, to check the distribution of each column.
- Performing Bivariate analysis, such as correlation heat-map, scatterplot etc.

Step 3 - Feature Engineering:

- Creating new columns using 'date' column.
- Removal of outliers.
- Scaling of the dataset.

Step 4 - Model Training and Model Selection:

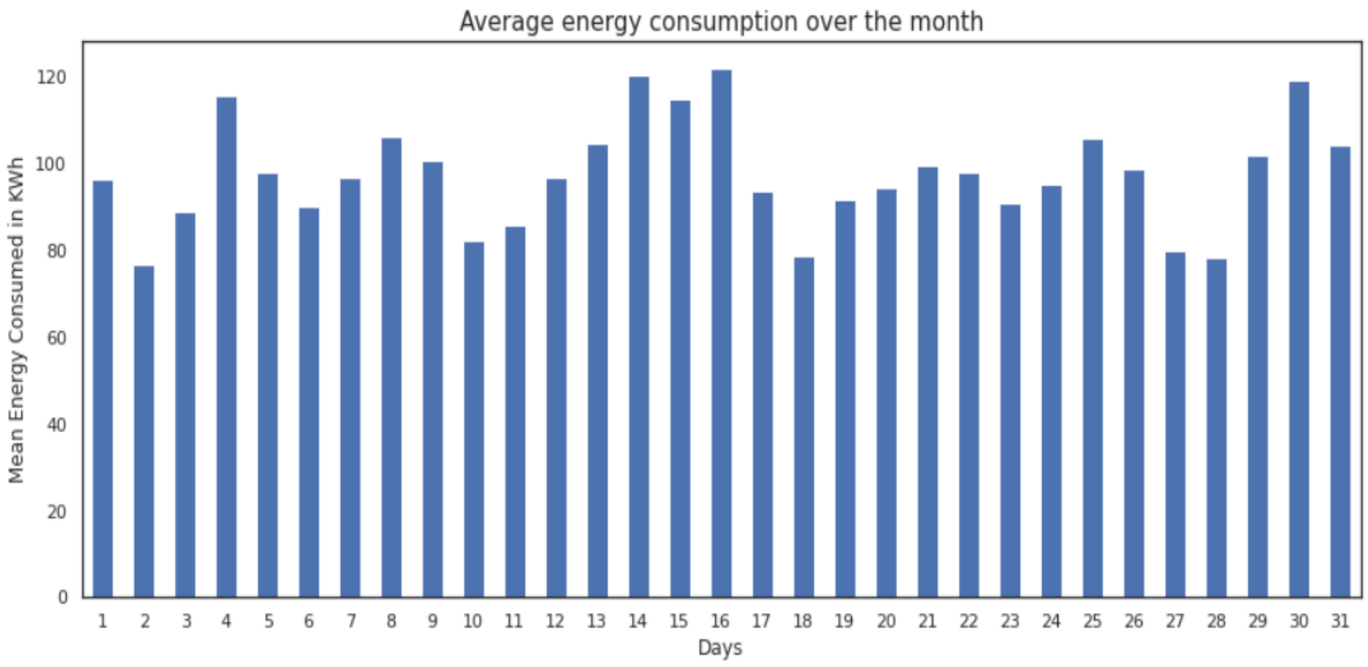
- Splitting the dataset into Training and Test sets.
- Training a set of regression models on the dataset and evaluating their results.
- Selection of top 3 models based on Evaluation Metrics, for further process.

Step 5 - Hyper-parameter Optimization:

- Tuning the top 3 models to find best estimators for each model.
- Comparing the results using Evaluation Metrics.
- Performing Stacking with these Tuned Models to check for improvements.

Exploratory Data Analysis

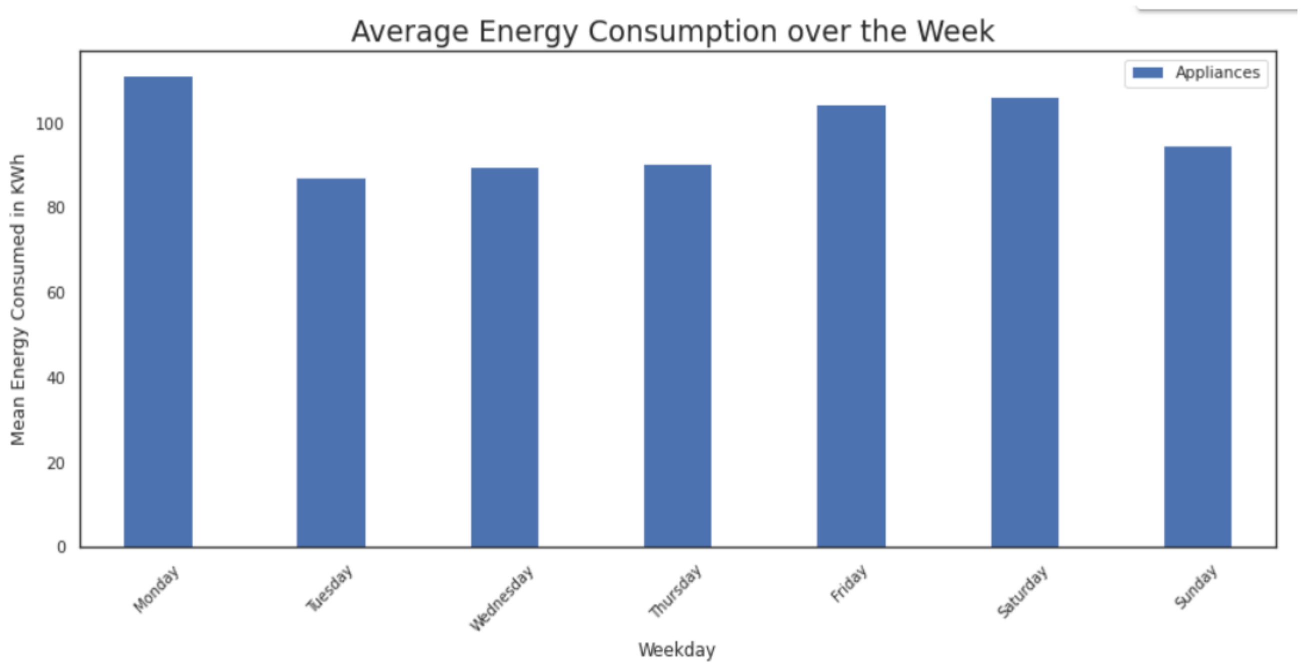
Average energy consumption over the month.



Observation:

- **The average Appliances energy consumption is not uniform in a month for 5 months of given Data.**

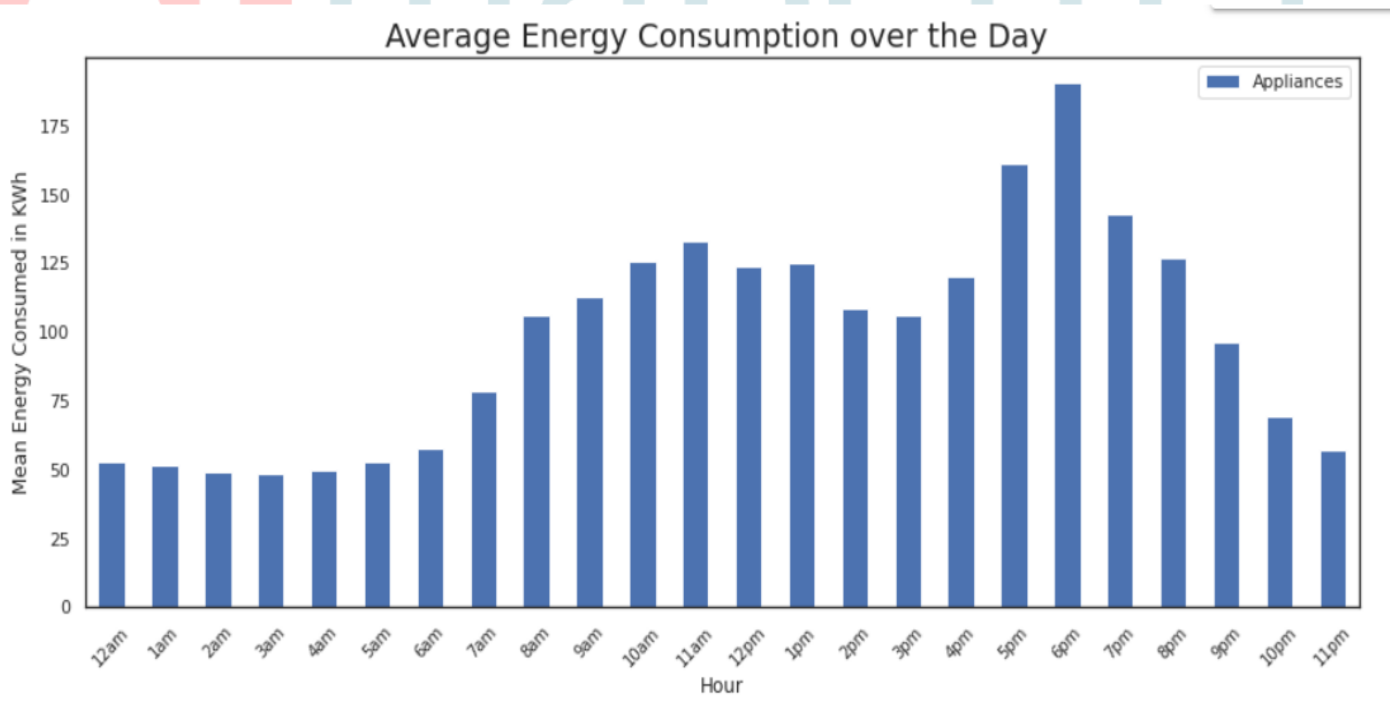
Average Energy Consumption over the Week.



Observation:

- **We can observe that the Energy Consumption is highest on Monday and compared to other days of the week higher on Fridays and Saturdays.**

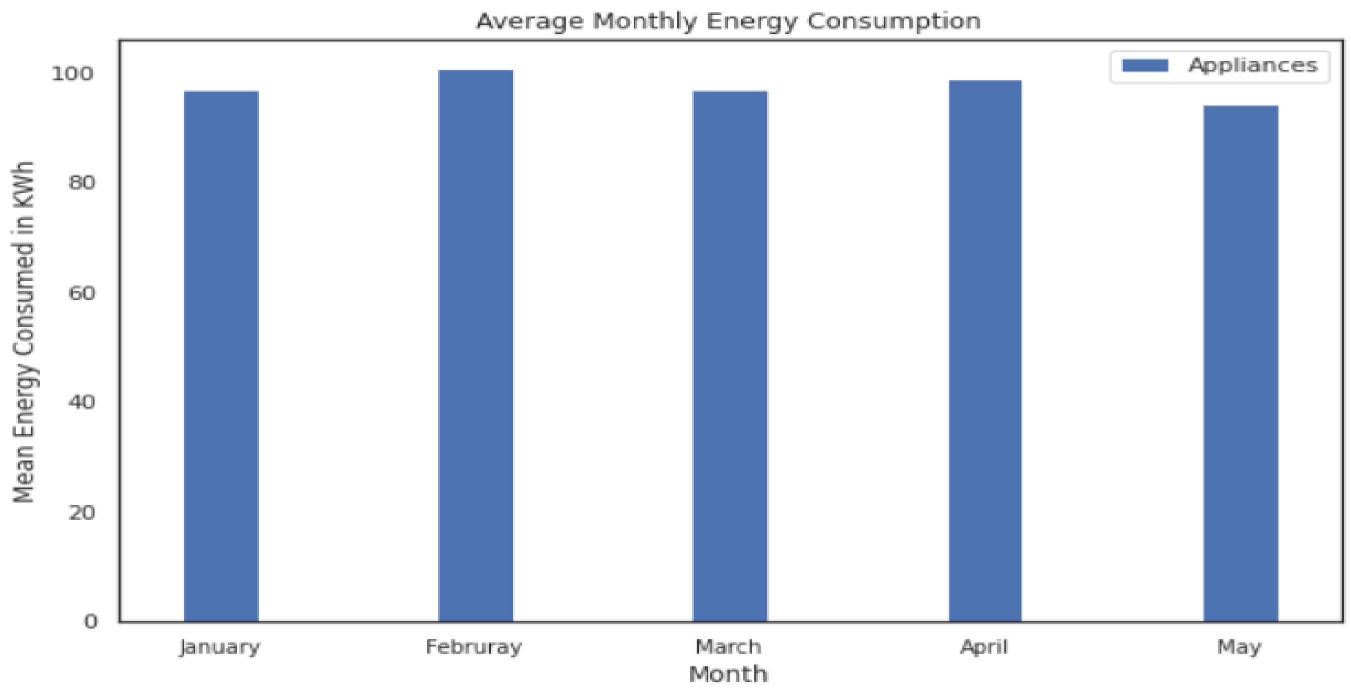
Average Energy Consumption over the Day.



Observation:

- **We can observe that in the evening Energy Consumption is higher compared to any other part of the day.**

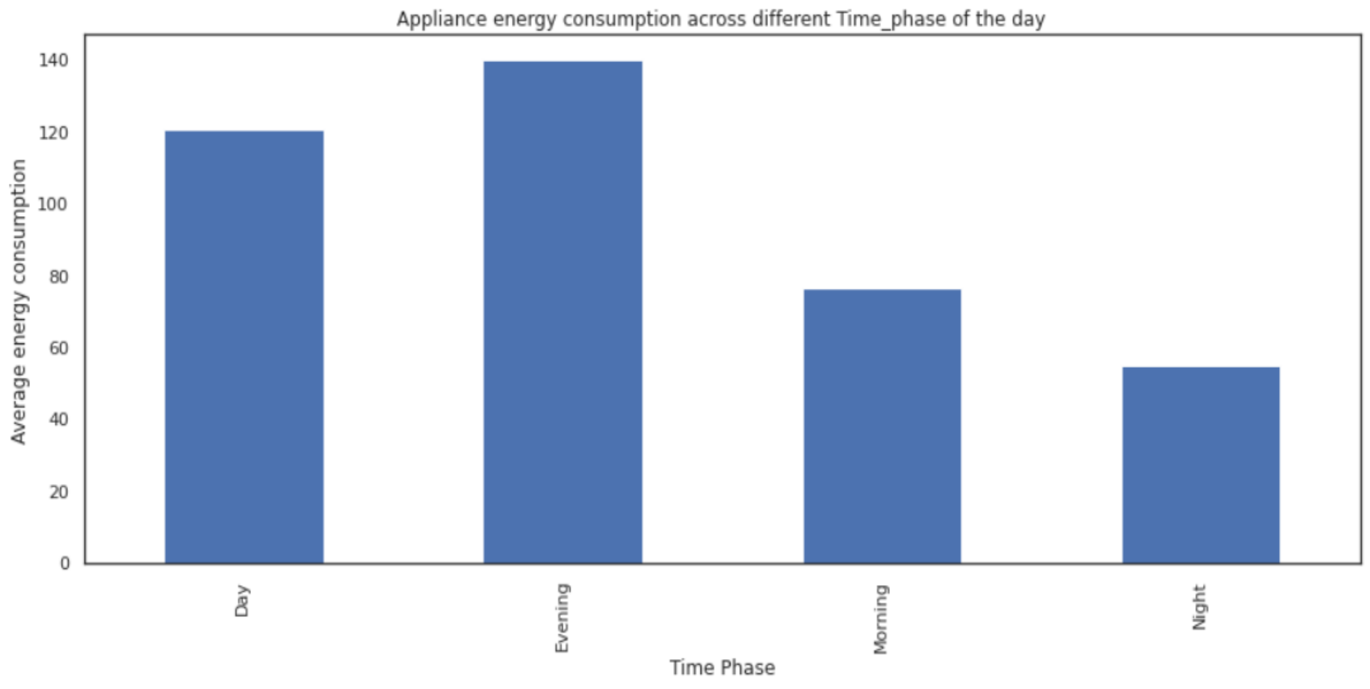
Average Monthly Energy Consumption.



Observation:

- **The average Appliances energy consumption is uniform for 5 months of given Data.**

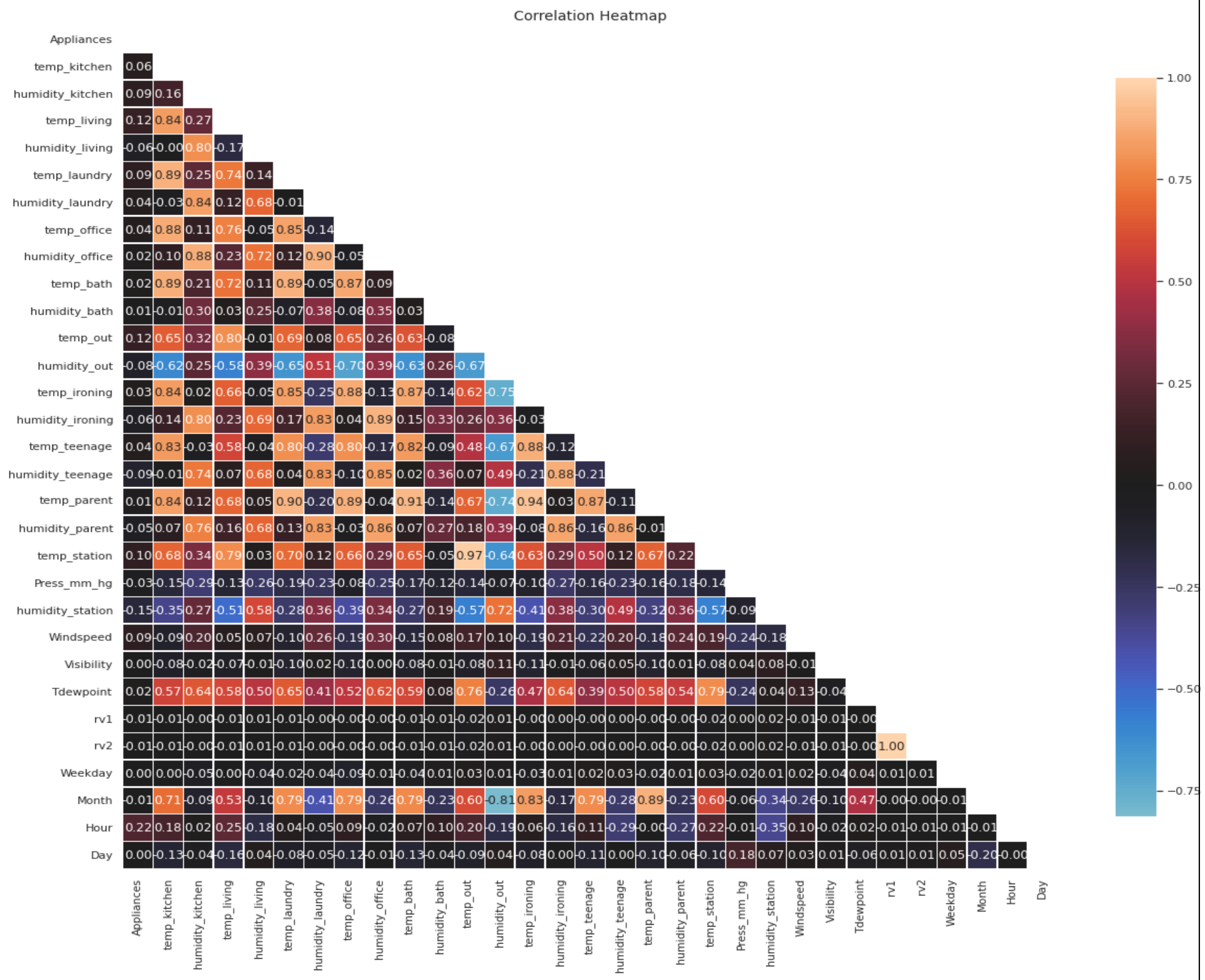
Appliance energy consumption across different Time_phase of the day.



Observation:

- **We can observe that the Energy Consumption is higher in the Evening compared to any other part of the day.**

Correlation Heat map.



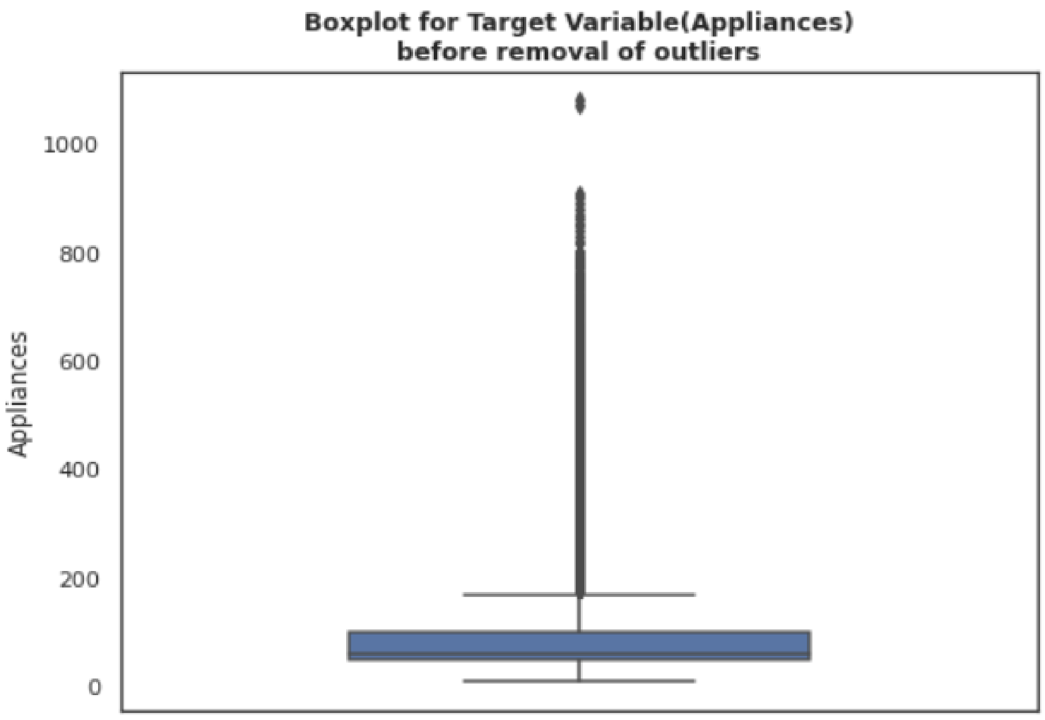
Observation:

- **The Target variable 'Appliances' don't have a good correlation of any of features column**
- **The columns'rv1' and 'visibility' have very low correlation with other variables**
- **The temperature variables have a good positive correlation among themselves.**

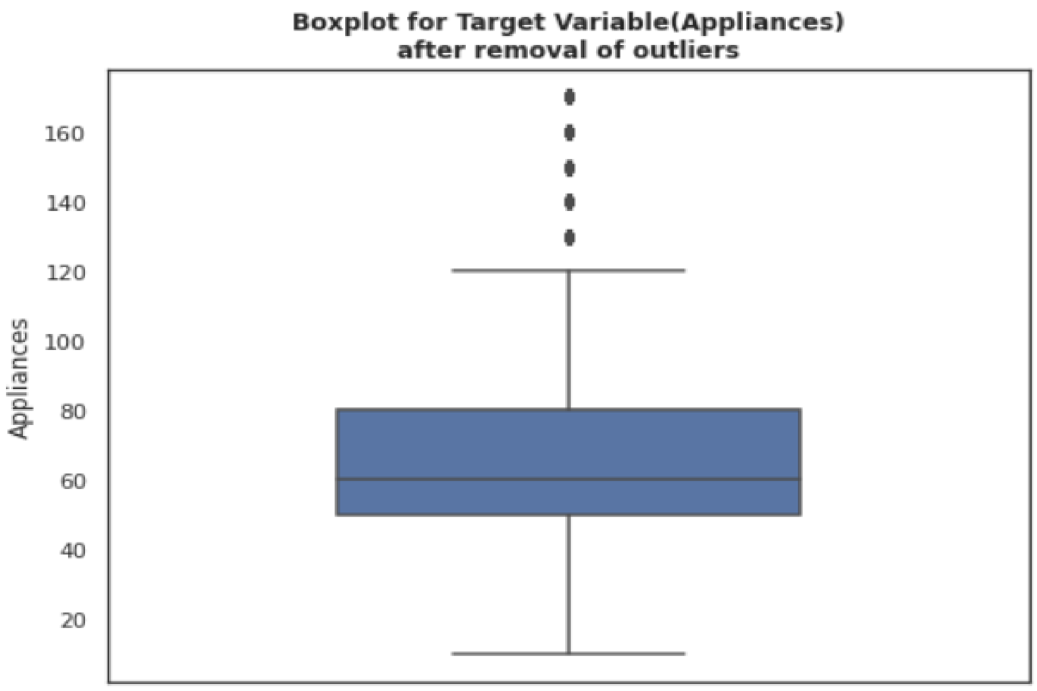
Feature Engineering:

Outlier Treatment:

Boxplot for Target Variable (Appliances) before removal of outliers.



Boxplot for Target Variable (Appliances) after removal of outliers.



Observation:

- **We observe that most of Outliers were removed from our dataset.**

Scaling the Data:

Normalization or Min-Max Scaling is used to transform features to be on a similar scale. The new point is calculated as:

$$X_{new} = (X - X_{min})/ (X_{max} - X_{min})$$

This scales the range to [0, 1] or sometimes [-1, 1]. Geometrically speaking, transformation squishes the n-dimensional data into an n-dimensional unit hypercube. Normalization is useful when there are no outliers as it cannot cope up with them. Usually, we would scale age and not incomes because only a few people have high incomes but the age is close to uniform.

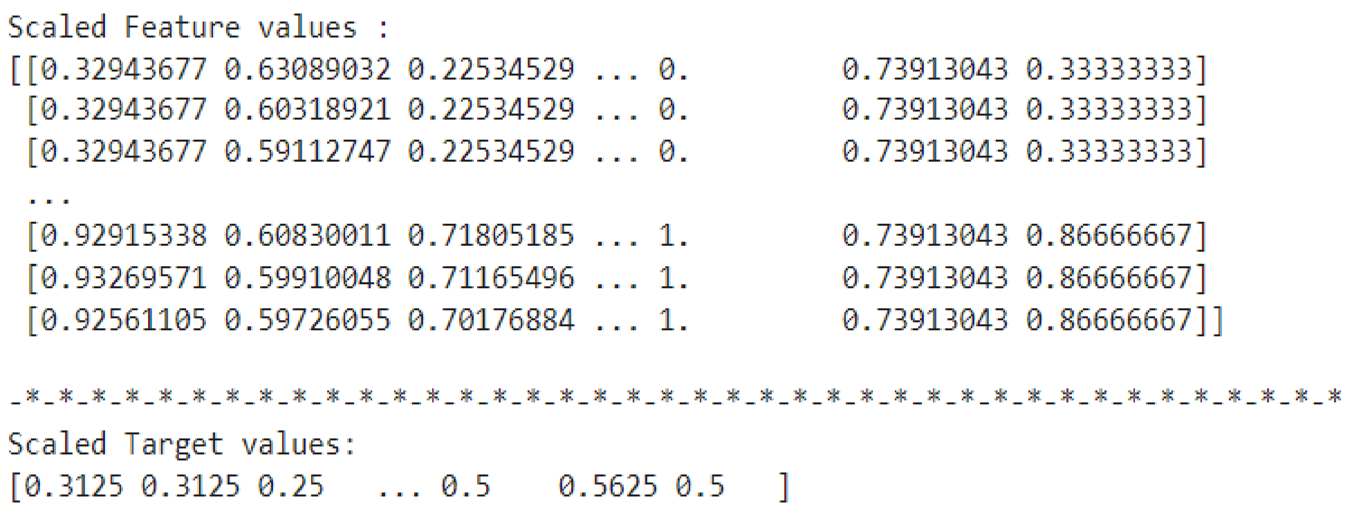
Standardization or Z-Score Normalization is the transformation of features by subtracting from mean and dividing by standard deviation. This is often called as Z-score.

$$X_{new} = (X - mean)/ Std$$

Standardization can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Geometrically speaking, it translates the data to the mean vector of original data to the origin and squishes or expands the points if std is 1 respectively. We can see that we are just changing mean and standard deviation to a standard normal distribution which is still normal thus the shape of the distribution is not affected.

Standardization does not get affected by outliers because there is no predefined range of transformed features.

- Since different columns are having values in different orders of magnitude, it is necessary to scale the Data.
- We have used min-max Scaler (Normalization) as most of the columns in the Data were not normally distributed.



```
Scaled Feature values :
[[0.32943677 0.63089032 0.22534529 ... 0.         0.73913043 0.33333333]
 [0.32943677 0.60318921 0.22534529 ... 0.         0.73913043 0.33333333]
 [0.32943677 0.59112747 0.22534529 ... 0.         0.73913043 0.33333333]
 ...
 [0.92915338 0.60830011 0.71805185 ... 1.         0.73913043 0.86666667]
 [0.93269571 0.59910048 0.71165496 ... 1.         0.73913043 0.86666667]
 [0.92561105 0.59726055 0.70176884 ... 1.         0.73913043 0.86666667]]

_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
Scaled Target values:
[0.3125 0.3125 0.25   ... 0.5    0.5625 0.5    ]
```

- **We see that our Feature values and Target values have been scaled between 0 and 1.**

Model Training and Model Selection

Model Training:

In Model Training we split our scaled dataset into two parts, ‘train’ and ‘test’, where we use the ‘train’ dataset for training different models(i.e. to learn the relationship between features and target values) and ‘test’ dataset for evaluating the results of models(i.e. to check models learning accuracy).

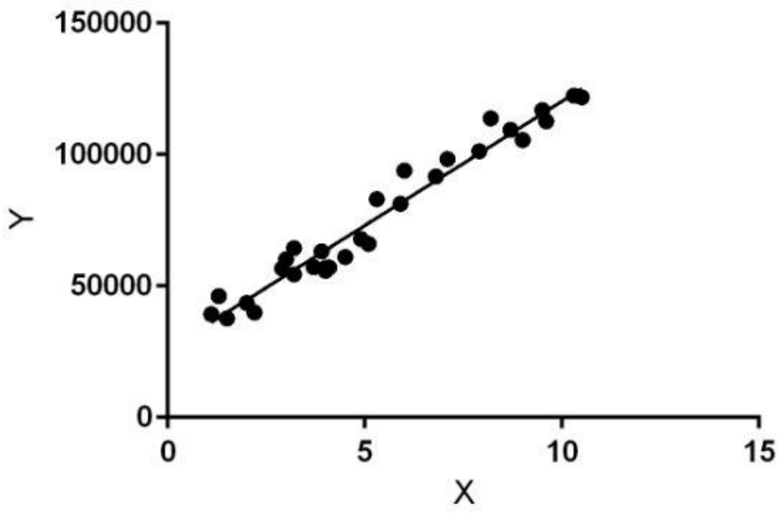
Model Selection:

In Mode Selection out of the models we trained in Model Training, we choose the models which are performing better and we will attempt to improve them in the next process.

Models:

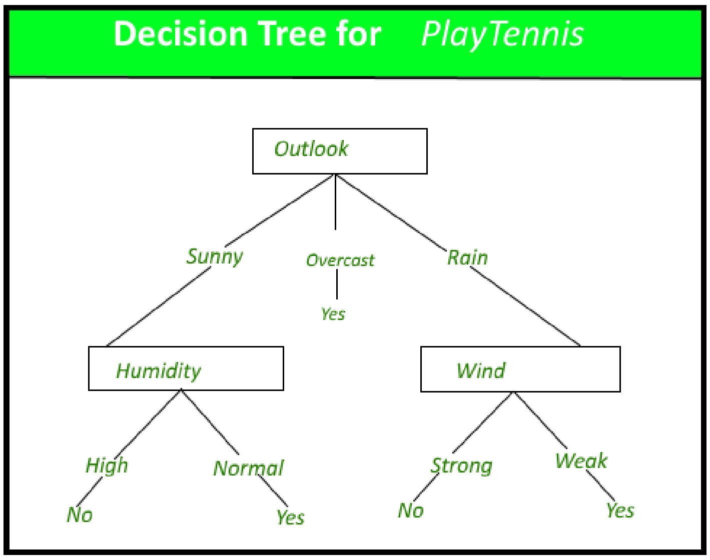
Linear Regression:

Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.
In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Decision Tree: Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



A decision tree for the concept Play Tennis.

Construction of Decision Tree:
A tree can be *"learned"* by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Decision Tree Representation:
Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.

The decision tree in above figure classifies a particular morning according to whether it is suitable for playing tennis and returning the classification associated with the particular leaf.(in this case Yes or No).

K-Nearest Neighbours:

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.
It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).
We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

Support Vector Machines:

Support Vector Machine (SVM) is a relatively simple **Supervised Machine Learning Algorithm** used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line.
In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyper-plane to separate the data. So by this, you must have understood that inherently, SVM can only perform binary classification (i.e., choose between two classes). However, there are various techniques to use for multi-class problems.

XGBoost:

XGBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in C++ which optimizes the training for Gradient Boosting. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

Random Forest:

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn't depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.

The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

Extra Trees Regressor:

Extremely Randomized Trees Regressor (Extra Trees Regressor) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a "forest" to output it's classification result. In concept, it is very similar to a Random Forest Regressor and only differs from it in the manner of construction of the decision trees in the forest.

Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria. This random sample of features leads to the creation of multiple de-correlated decision trees.

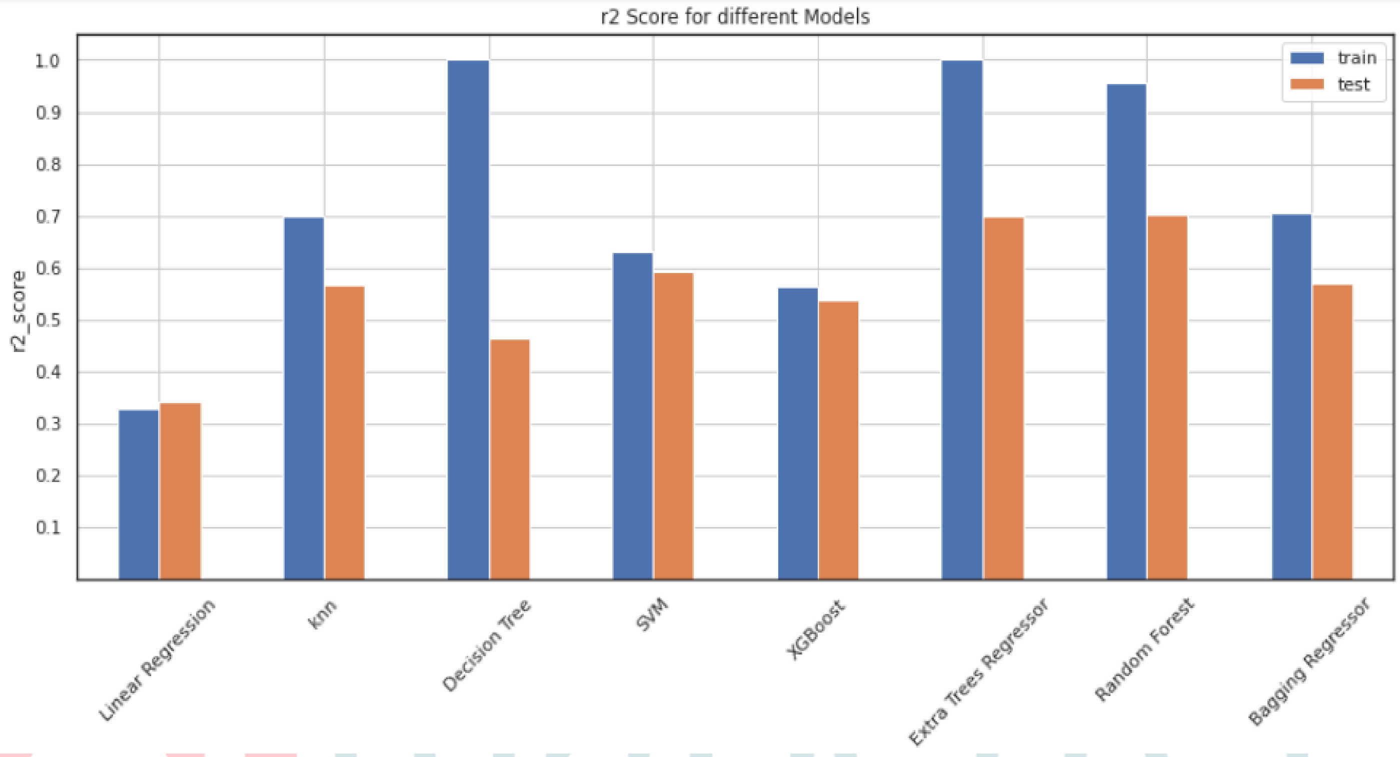
Bagging Regressor:

A Bagging regressor is an ensemble meta-estimator that fits base regression models each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.
Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples (or data) from the original training dataset – *where N is the size of the original training set*. Training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

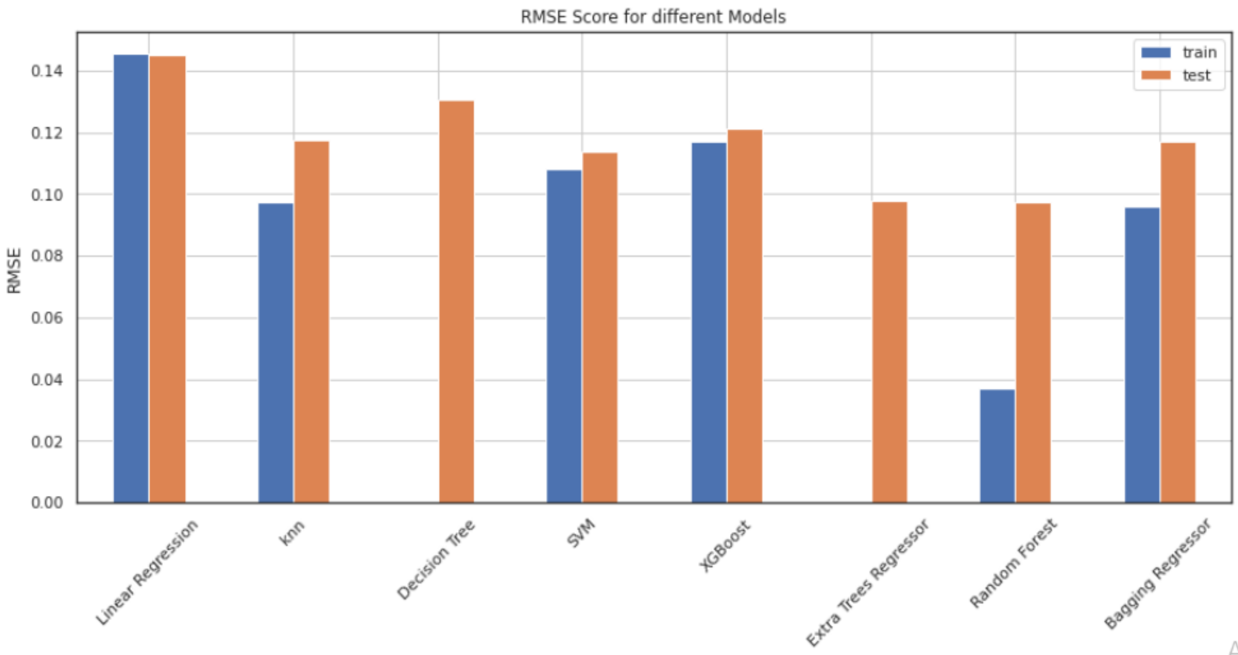
Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

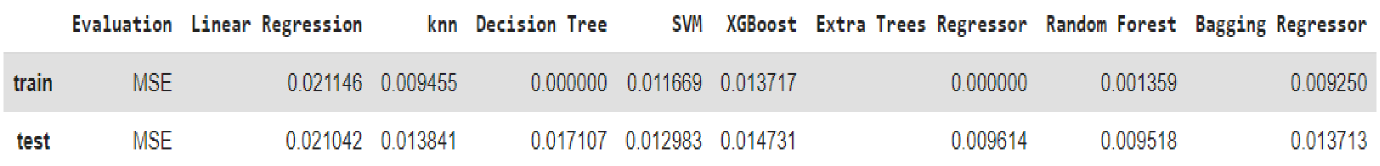
Combined Evaluation metric results of above models:

	Evaluation	Linear Regression	knn	Decision Tree	SVM	XGBoost	Extra Trees Regressor	Random Forest	Bagging Regressor
train	r2_score	0.329866	0.700366	1.000000	0.630196	0.565301	1.000000	0.956947	0.706840
test	r2_score	0.342125	0.567250	0.465146	0.594094	0.539436	0.699427	0.702426	0.571262

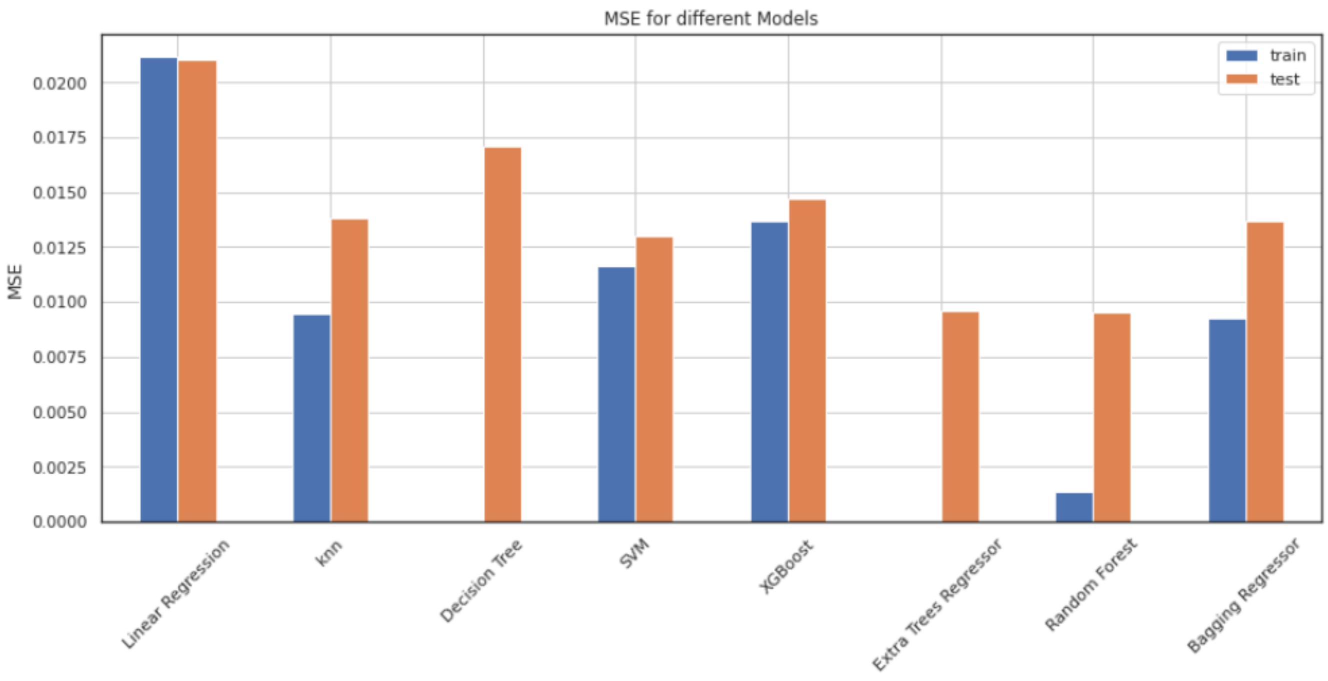


	Evaluation	Linear Regression	knn	Decision Tree	SVM	XGBoost	Extra Trees Regressor	Random Forest	Bagging Regressor
train	RMSE	0.145415	0.097235	0.000000	0.108023	0.117118	0.000000	0.036858	0.096179
test	RMSE	0.145057	0.117648	0.130793	0.113941	0.121370	0.098049	0.097558	0.117102





	Evaluation	Linear Regression	knn	Decision Tree	SVM	XGBoost	Extra Trees Regressor	Random Forest	Bagging Regressor
train	MSE	0.021146	0.009455	0.000000	0.011669	0.013717	0.000000	0.001359	0.009250
test	MSE	0.021042	0.013841	0.017107	0.012983	0.014731	0.009614	0.009518	0.013713



Observations:

- Taking 0.5 as a cut-off for R-squared score, Extra Trees regressor, Random Forest Regressor, SVM, KNN, XGBoost and Bagging Regressors are performing well.
- But, Random Forest regressor and Extra Trees Regressor outperformed above models without tuning.
- We can observe from above Graphs that Decision Tree is a strong learner compared to any other model. So, we choose to Hypertune only the ensemble methods of Decision Tree**.**

Hyperparameter Tuning

Grid Search CV:

In GridSearchCV approach, machine learning model is evaluated for a range of hyperparameter values. It searches for best set of hyperparameters from a grid of hyperparameters values.

Randomised Search CV:

RandomizedSearchCV solves the drawbacks of GridSearchCV, as it goes through only a fixed number of hyperparameter settings. It moves within the grid in random fashion to find the best set hyperparameters. This approach time efficient.

Hyperparameter tuning for ExtraTreesRegressor:

Training MSE	0.002281818285467869
Training RMSE	0.04776838165008177
Training R2 score	0.9276858234558366
Training Adjusted R2 score	0.9275151462262622
Testing MSE	0.009164526571338849
Testing RMSE	0.09573153383989441
Testing R2 score	0.7134668000757074
Testing Adjusted R2 score	0.711884045257078

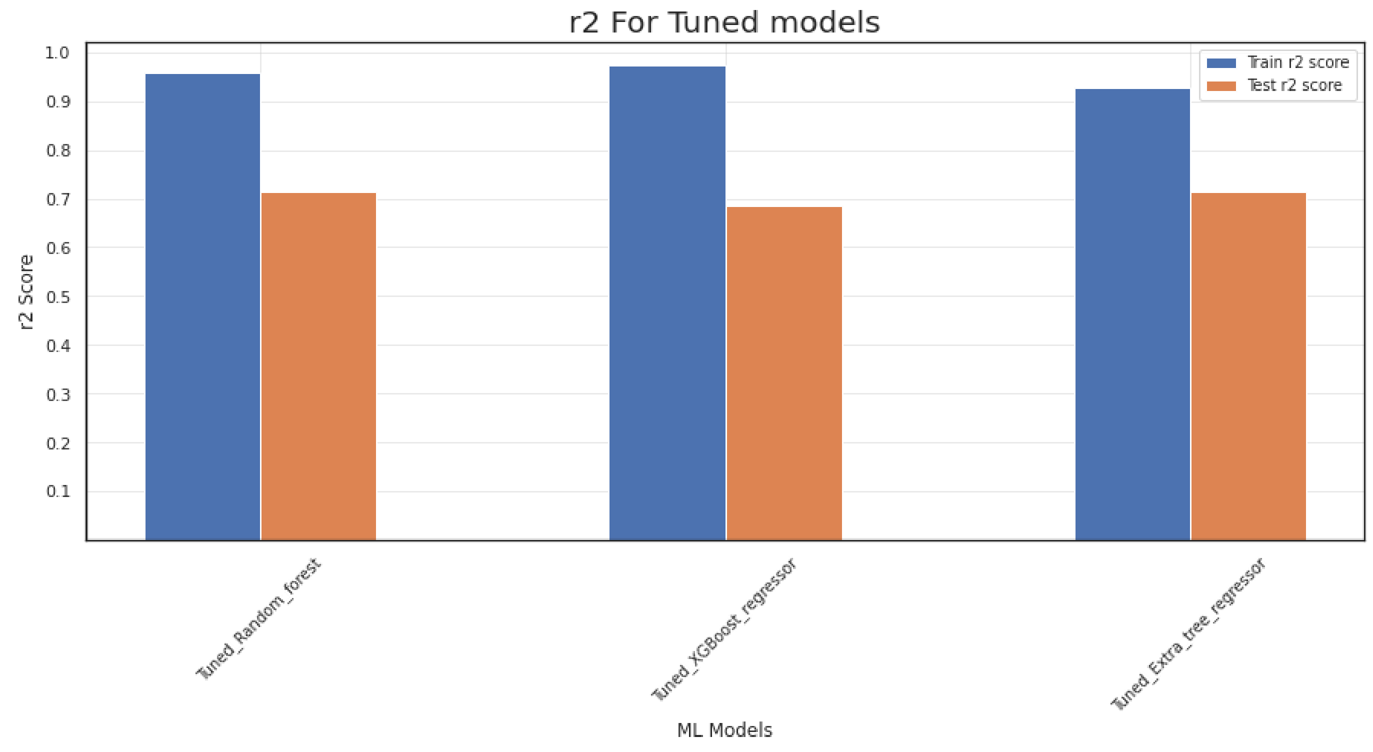
Hyperparameter tuning for RandomForest:

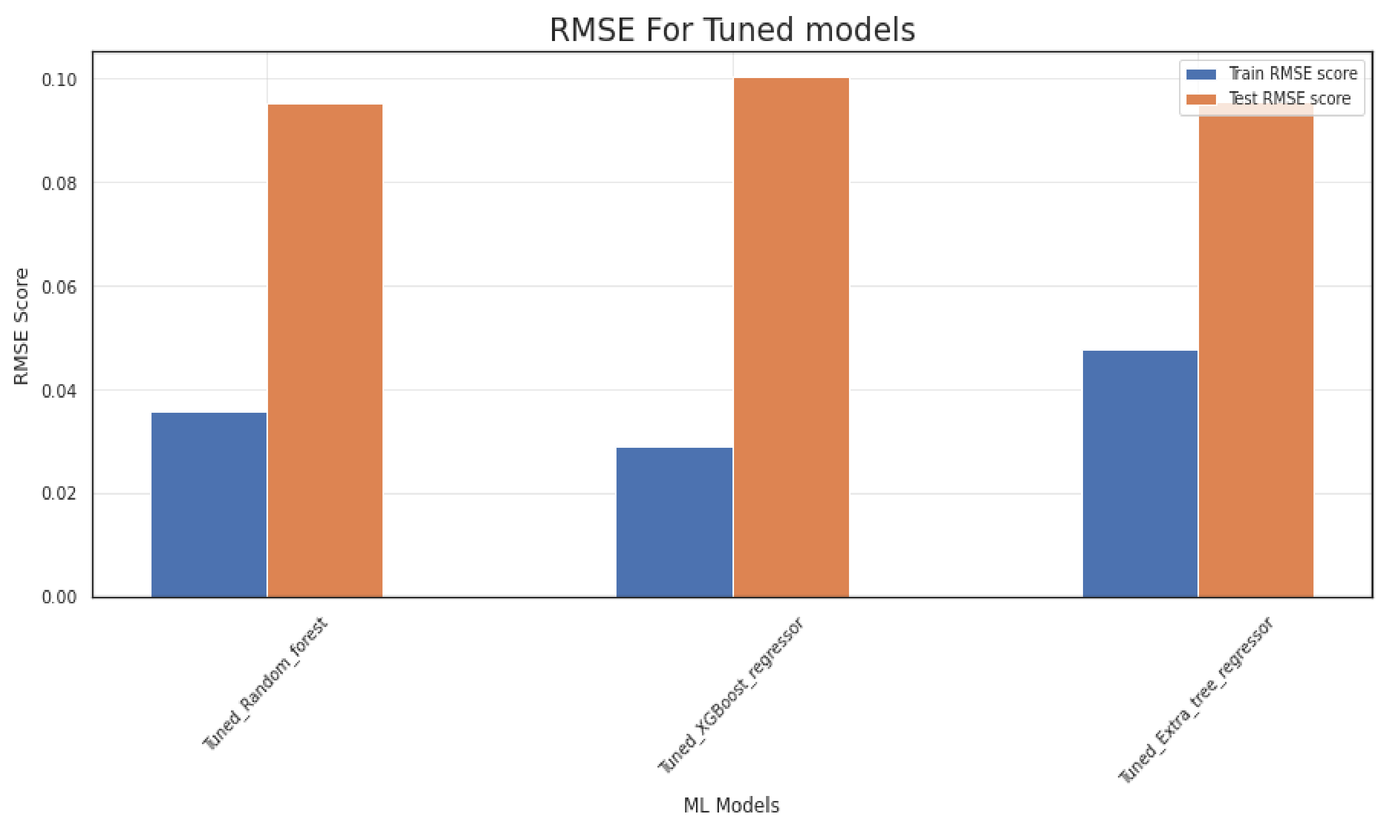
Training MSE	0.0012789474961086558
Training RMSE	0.03576237542597885
Training R2 score	0.9594682733444072
Training Adjusted R2 score	0.9593726096288532
Testing MSE	0.009109484343742601
Testing RMSE	0.09544361866433293
Testing R2 score	0.7151877210072302
Testing Adjusted R2 score	0.713614472228032

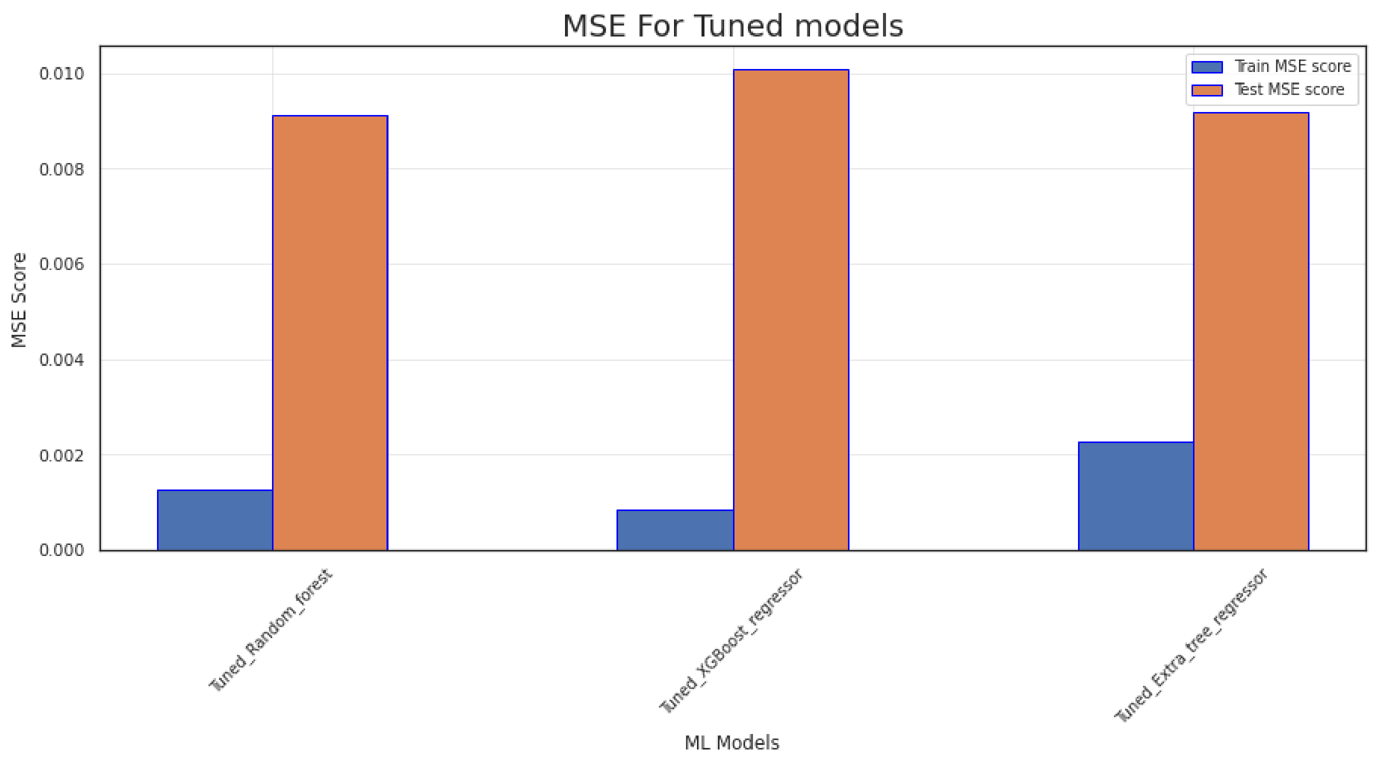
Hyperparameter tuning for XGBoost:

Training MSE	0.0008472341490312861
Training RMSE	0.02910728687169737
Training R2 score	0.9731499040841763
Training Adjusted R2 score	0.9730865320013604
Testing MSE	0.010069624808909699
Testing RMSE	0.10034752019312534
Testing R2 score	0.6851684813095121
Testing Adjusted R2 score	0.6834294119681742

Combined Evaluation metric results of Tuned models:







Observation:

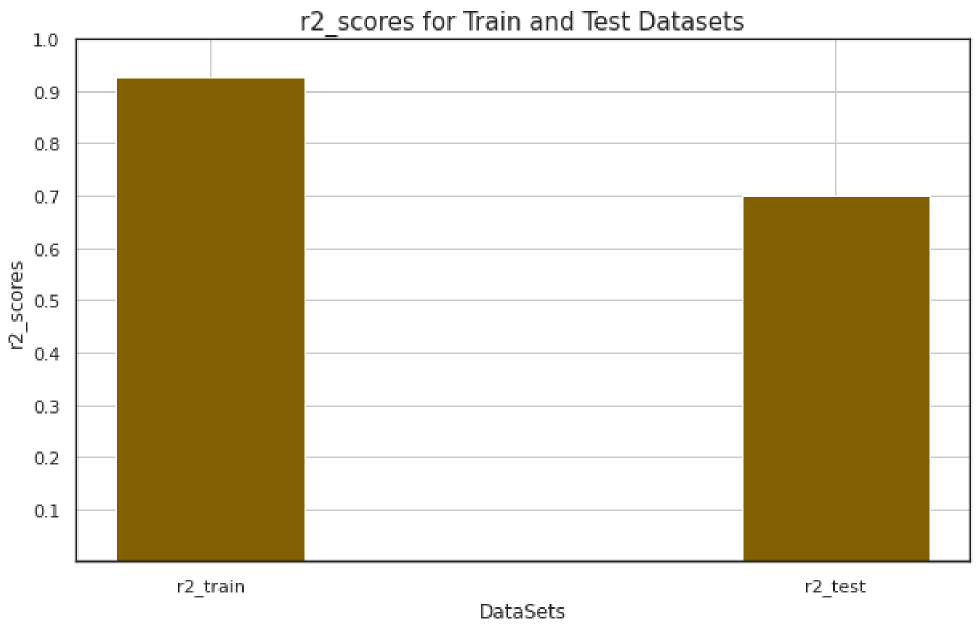
- We observe from above graphs that Extra Trees Regressor is giving more generalised model compar ed to Random Forest Regressor and XGBoost.
- R-squared value of test set for Extra Trees Regressor has improved after tuning from 0.700 to 0.712.
- R-squared value of train set for Extra Trees Regressor has decreased after tuning from 1.00 to 0.93, which means that the model is not overfitting anymore and it's a more generalised.

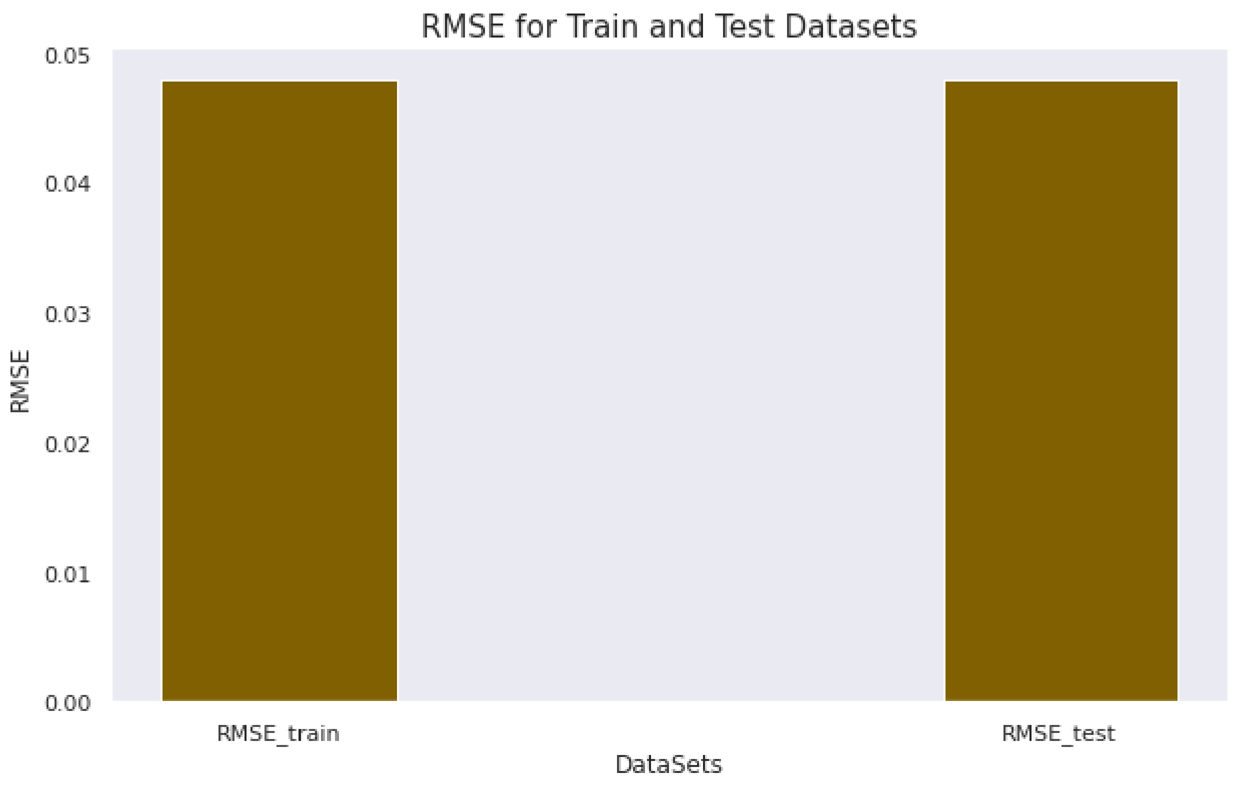
Stacking

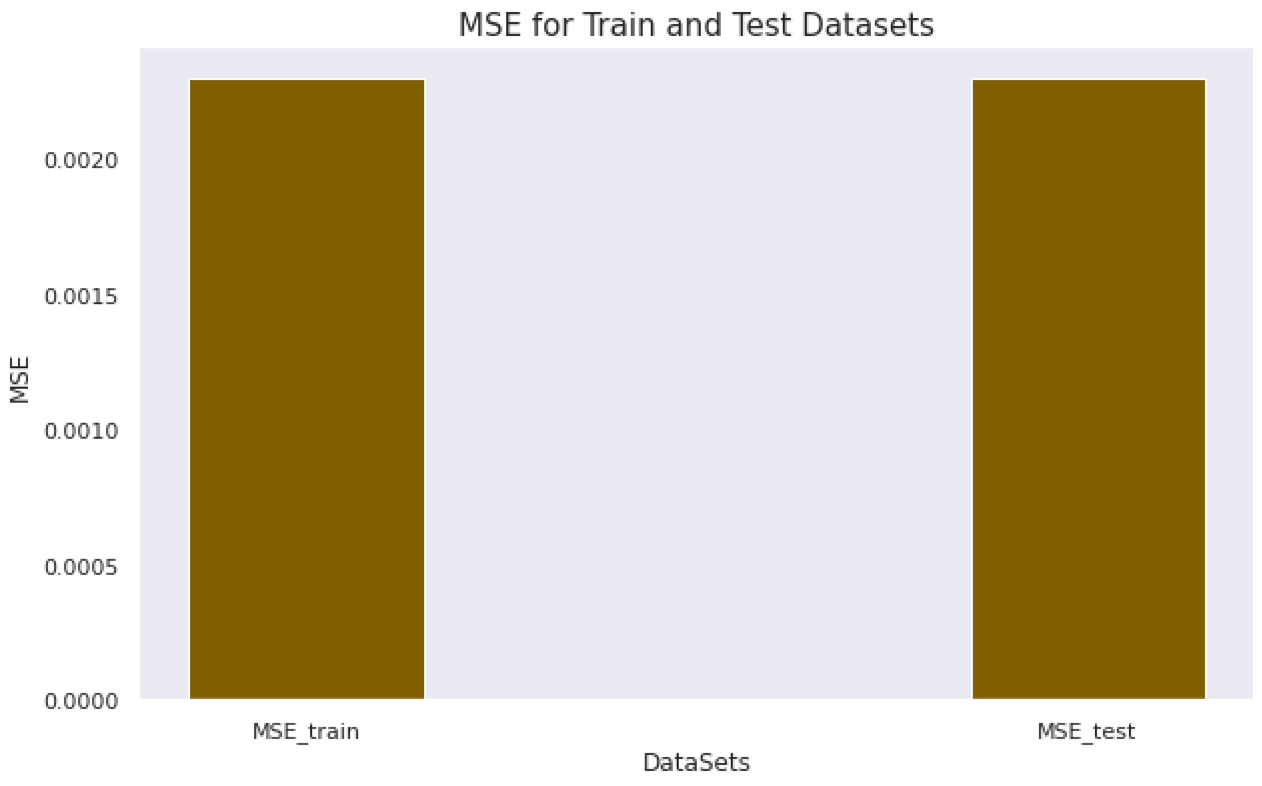
Stacking is a way to ensemble multiple classifications or regression model. There are many ways to ensemble models, the widely known models are Bagging or Boosting. Bagging allows multiple similar models with high variance are averaged to decrease variance. Boosting builds multiple incremental models to decrease the bias, while keeping variance small.

Evaluation metrics of the stacked model:

Train r2 for Stacking:	0.9270943016259064
Test r2 for Stacking:	0.7011097886391899





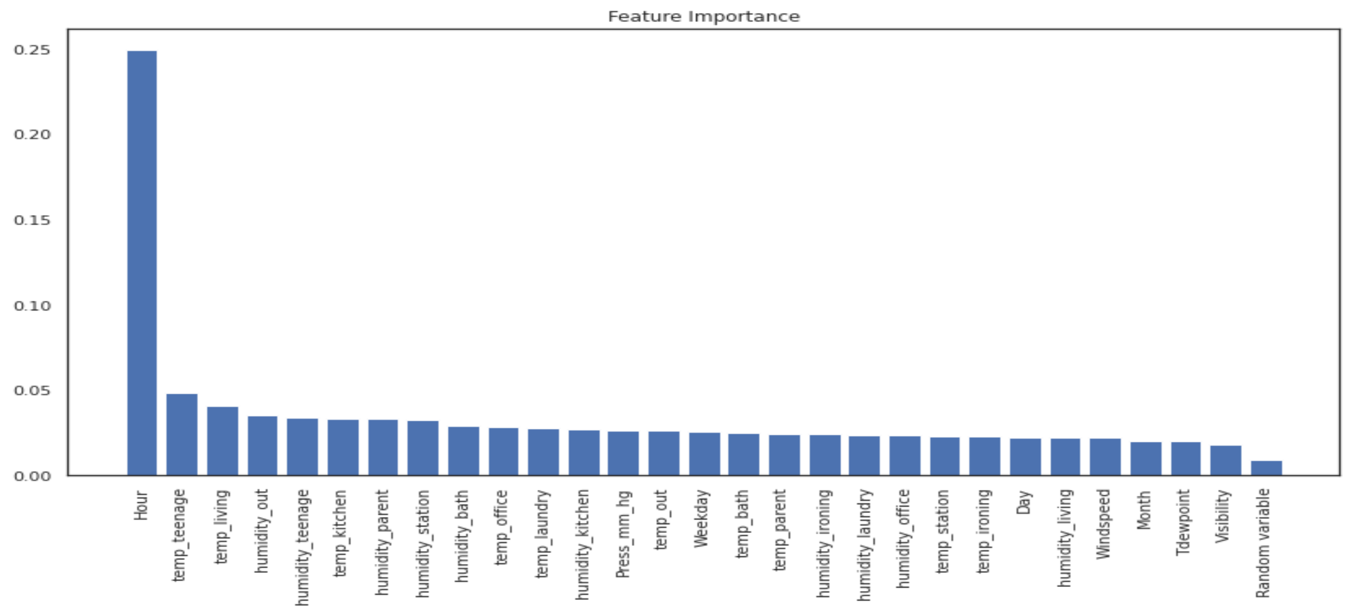


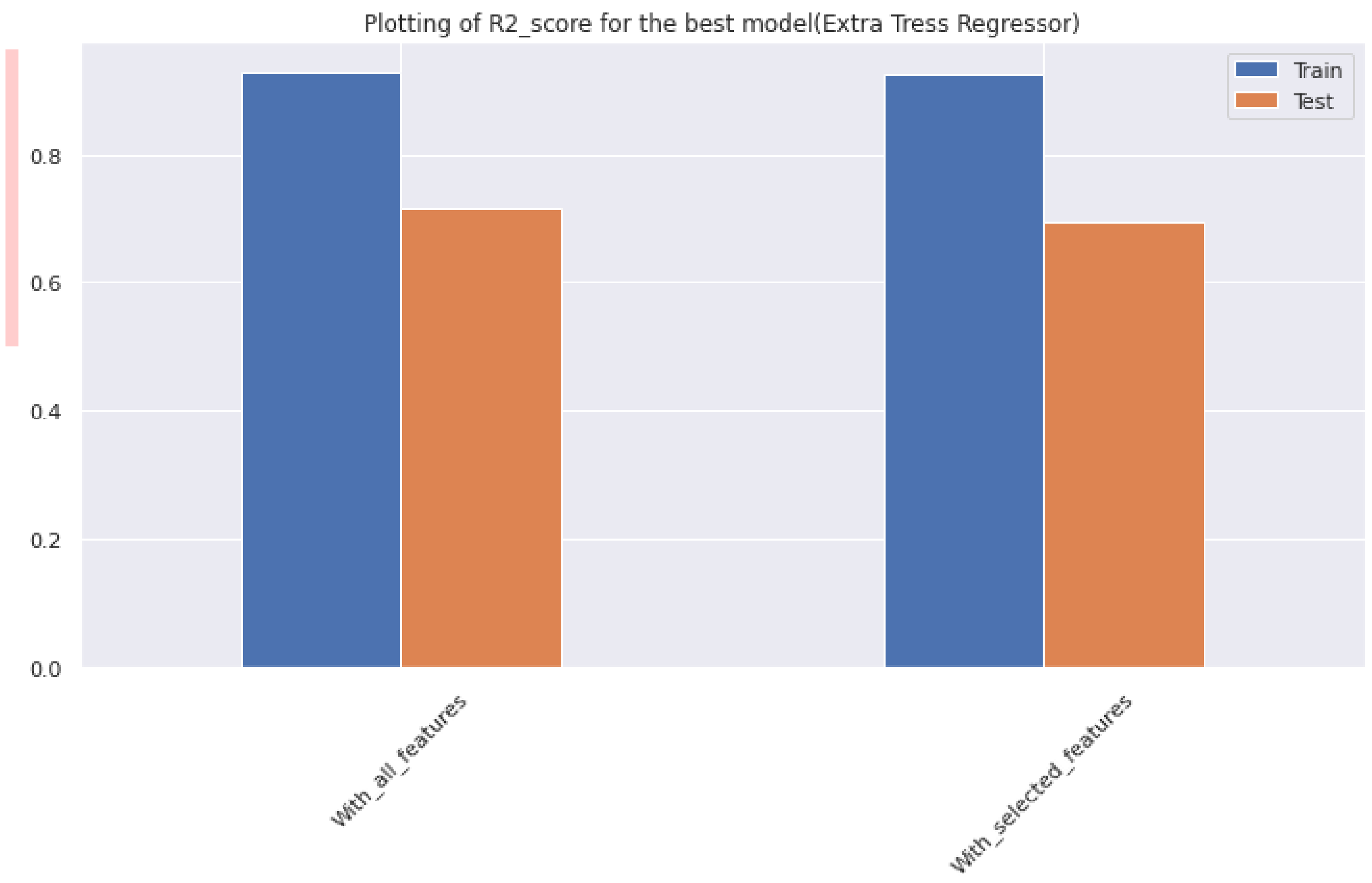
Observation:

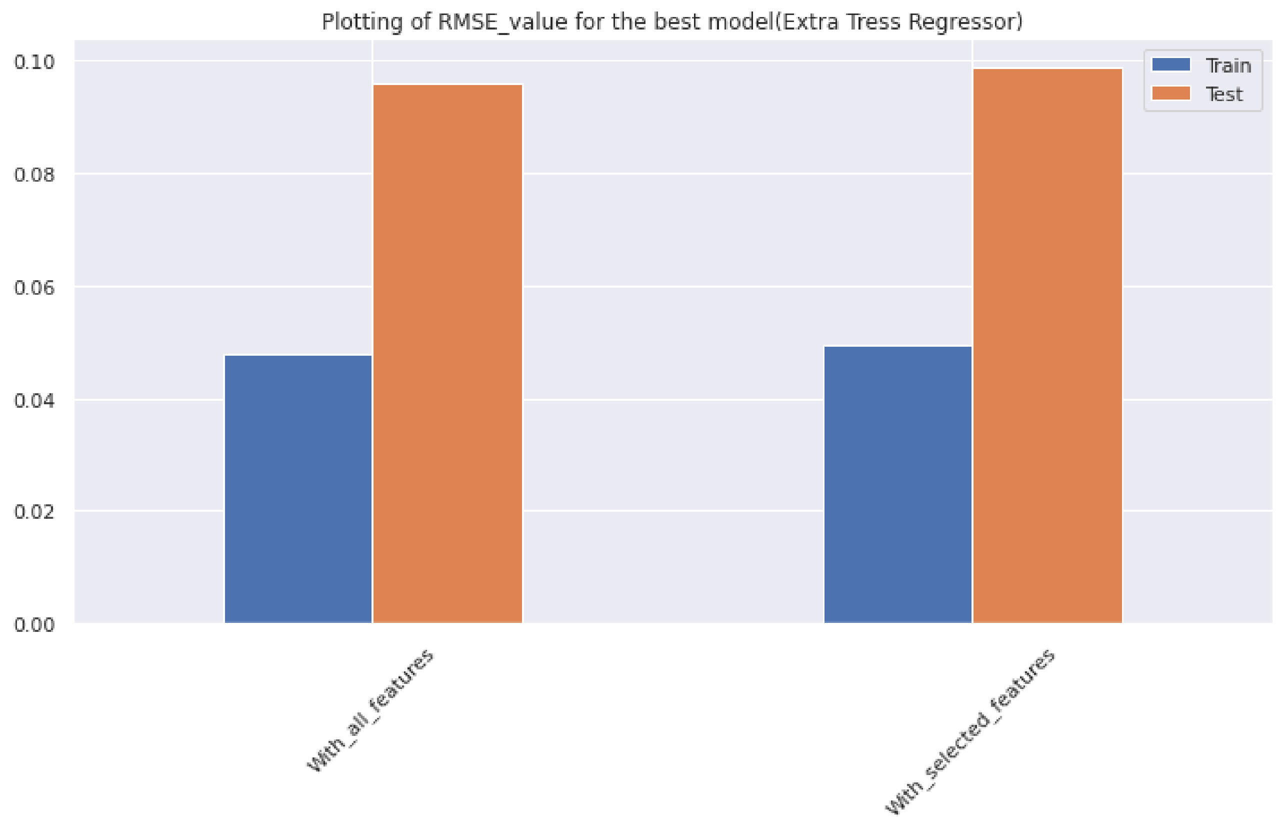
- We observe that as we performed Stacking with HyperTuned Bese models (Random Forest Regressor, XGBoost Regressor and Extra Trees Regressor), the results are closer to those of Hyper tuned Extra Trees Regressor, but Hypertuned Extra Trees Regressor is performing slightly better.

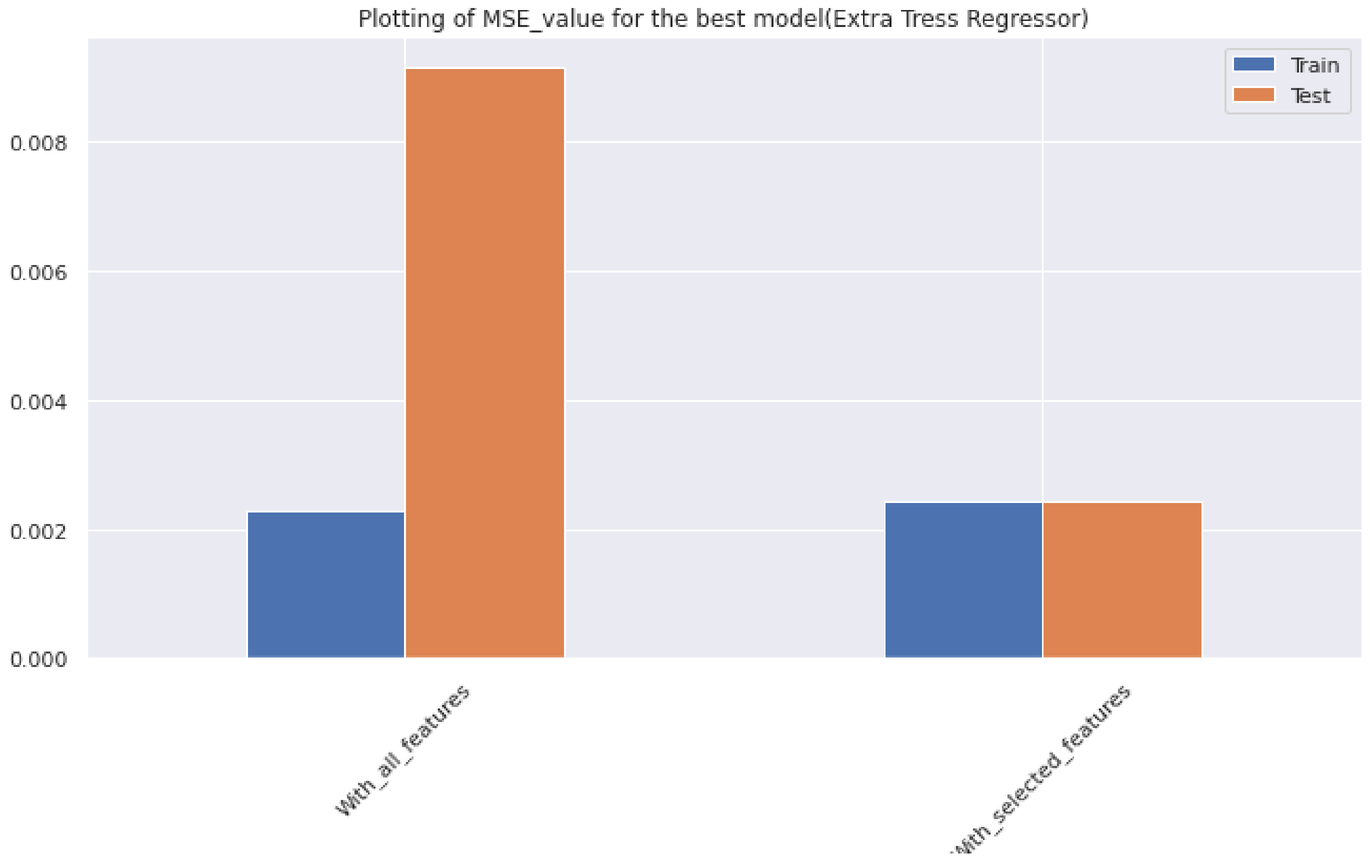
- Hence our choice of model is Hypertuned Extra Trees Regressor.

Features Contributing to the model output:






Plotting of RMSE_value for the best model(Extra Tress Regressor)
0.10
0.08
0.06
0.04
0.02
0.00
Train
Test
With_all_features
With_selected_features


Plotting of MSE_value for the best model(Extra Tress Regressor)
0.008
0.006
0.004
0.002
0.000
Train
Test
With_all_features
With_selected_features

Conclusion:

To sum up our analysis, at first, we can see from our explorations through various features that mean and standard deviations of temperature values inside and outside the house differ notably, which indicate that there has been significant use of appliances from the household. Then the average energy consumptions were highest in the evening than any other parts of the day and higher on Mondays, Fridays and Saturdays compared to any other days of the week. Temperatures both inside and outside were increasing over 5 months and Humidity values were decreasing.

Although both temperature features and humidity features had good positive correlations among them, their correlations with the dependent variable(Appliance) were poor. There were significant amount outliers in 'Appliance'. Many of the features were skewed, and so we used Normalization for scaling after removing the outliers.

We then used various Regression algorithms for modelling, out of which Random Forest, Extra-Trees Regressor and XGBoost performed better with respect to evaluation metrics, Mean Squared Error, Root Mean Squared Error and R-squared score. We then tuned various Hyper-parameters for these 3 models to improve their accuracy. Even though all 3 models improved after tuning, Extra-Trees Regressor had a slight edge over Random Forest and XGBoost, as it was a better generalization from train to test data than the other two. Extra-Trees Regressor yielded an adjusted R2-score of 0.923 for training set and 0.712 for the testing set.

Then we performed stacking by combining the hyper-tuned top 3 models, but its performance was close to, but was not better than the Extra-Trees Regressor. We also performed feature selection and 10 features were selected for modelling, but the results were not as satisfactory as those of Extra-Trees Regressor. Therefore, the Hyper-tuned Extra-Trees Regressor is the best model for Appliance Energy Consumption for the data provided.

References:

i. GeeksForGeeks
ii. MachineLearningMastery
iii. Towardsdatascience
iv. Analytics Vidhya