

Appliances Energy Prediction (Capstone Project)

Project Overview:

In our time, one of the things that is for sure is in need is energy and in increasing quantities for a better quality of life and to support economic and social progress, in particular in developing countries. But even now there are a lot of places in developing countries where there are outages .

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

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

Some related work:

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

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

Problem Statement:

We will predict Appliance energy consumption for a house based on factors like temperature, humidity and pressure. In order to achieve this, we need to develop a supervised learning model with regression algorithms .

Regression algorithms are used because data consists of continuous features and there are no identification of appliances in dataset

Metrics:

The metrics used is R2 score & RMSE (Root Mean Squared Error). They are used to measure quality and residual of regression problems.

Mathematically R2 score is defined as

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

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

Mathematically RMSE score is defined as

$$RMSE = \sqrt{\sum \frac{(y_{pred} - y_{ref})^2}{N}}$$

It represents the standard deviation of the differences between predicted values and true values .RMSE indicates how accurate is the predictions compared to actual values.

Data Exploration:

The main features are temperature, humidity and pressure readings. Each observation measures electricity in a 10-minutes intervals. The humidity and temperatures have been averaged for 10-minutes 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 = pd.read_csv("energydata_complete.csv")
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	Wi
0	2016-01-11 17:00:00	60	30	19.89	47.5966667	19.2	44.790000	19.79	44.730000	19.000000	...	17.033333	45.53	6.600000	733.5	92.0	
1	2016-01-11 17:10:00	60	30	19.89	46.6933333	19.2	44.722500	19.79	44.790000	19.000000	...	17.066667	45.56	6.483333	733.6	92.0	
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	
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	
4	2016-01-11 17:40:00	60	40	19.89	46.3333333	19.2	44.530000	19.79	45.000000	18.890000	...	17.000000	45.40	6.133333	733.9	92.0	

5 rows × 29 columns

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 will add one more column temporarily (WEEKDAY)which focus on if a day was a weekday or a weekend in order to check the difference in appliance consumption
2. Lights column will be removed as they are the reading of a submeter and we are not focusing on appliance specific reading and more than 75% of the readings is zero
5. Total number of rows - 19735
6. The data set will be split 80-20 % between train & test.
7. Total # of rows in training set - 15788

8. Total # of rows in test set - 3947
9. All the features have numerical values. There are no categorical features.
10. Number of missing values & null values = 0

Data used as input to train:

Dropping lights column as i said above

```
# Due to Lot of zero enteries this column is of not much use and will be ignored in rest of the model
feature_vars.drop(['lights'], axis=1, inplace=True);
```

```
feature_vars.head()
```

	T1	T2	T3	T4	T5	T6	T7	T8	T9	RH_1	...	RH_8	RH_9	T_out	Tdewpoint
2133	19.890000	19.200000	20.390000	19.10	17.511111	11.100000	17.50	18.111111	17.166667	45.50	...	50.000000	48.700000	10.300000	7.950000
19730	25.566667	25.890000	27.200000	24.70	23.200000	24.796667	24.50	24.700000	23.200000	46.56	...	50.074000	46.790000	22.733333	13.333333
3288	22.500000	21.533333	21.963333	22.00	19.100000	6.530000	19.29	20.566667	18.600000	44.43	...	41.331111	45.530000	6.600000	0.200000
7730	19.790000	17.200000	20.600000	18.39	18.290000	2.790000	18.10	20.500000	18.390000	38.06	...	42.590000	40.723333	2.100000	1.233333
8852	20.600000	17.100000	20.290000	19.50	18.200000	-0.666667	20.70	22.700000	18.926667	35.29	...	39.260000	40.090000	-0.866667	-1.933333

5 rows × 26 columns

```
feature_vars.describe()
```

	T1	T2	T3	T4	T5	T6	T7	T8	T9	RH_1	...
count	15788.000000	15788.000000	15788.000000	15788.000000	15788.000000	15788.000000	15788.000000	15788.000000	15788.000000	15788.000000	...
mean	21.688684	20.345215	22.269113	20.854106	19.598492	7.913931	20.266936	22.029474	19.488493	40.266486	...
std	1.609561	2.196357	2.007629	2.051129	1.850812	6.102256	2.116903	1.959194	2.021992	3.957219	...
min	16.790000	16.100000	17.200000	15.100000	15.330000	-6.030000	15.390000	16.306667	14.890000	27.023333	...
25%	20.775937	18.823333	20.790000	19.533333	18.290000	3.595000	18.700000	20.790000	18.000000	37.399167	...
50%	21.600000	20.000000	22.100000	20.633333	19.390000	7.300000	20.060000	22.100000	19.390000	39.663333	...
75%	22.600000	21.500000	23.290000	22.100000	20.632083	11.263333	21.600000	23.390000	20.600000	43.060000	...
max	26.260000	29.856667	29.236000	26.200000	25.795000	28.290000	25.963333	27.230000	24.500000	57.423333	...

8 rows × 26 columns

Target variable:

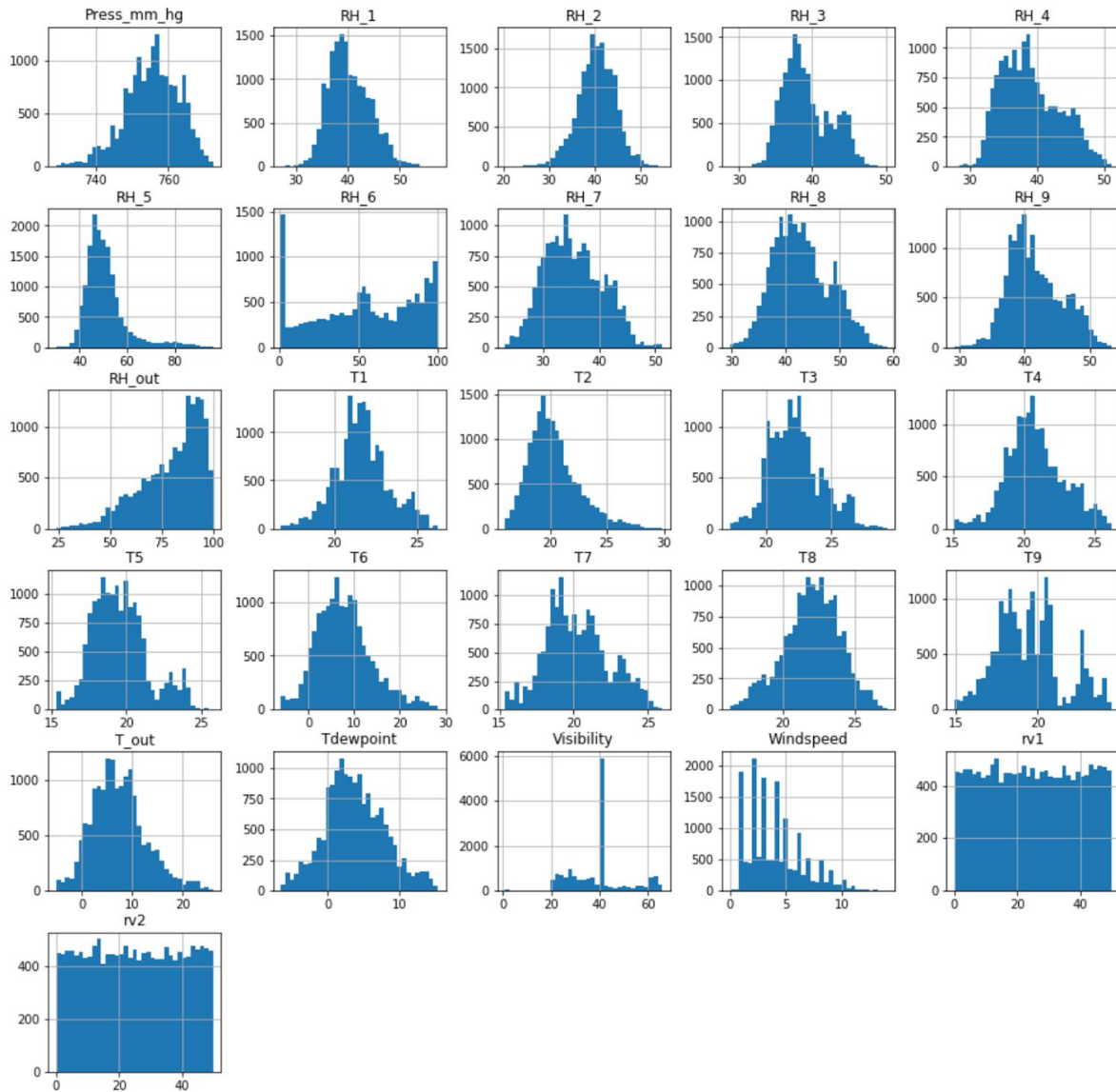
```
target_vars.describe()
```

Appliances	
count	15788.000000
mean	97.949709
std	103.136328
min	10.000000
25%	50.000000
50%	60.000000
75%	100.000000
max	1080.000000

Exploratory Visualization:

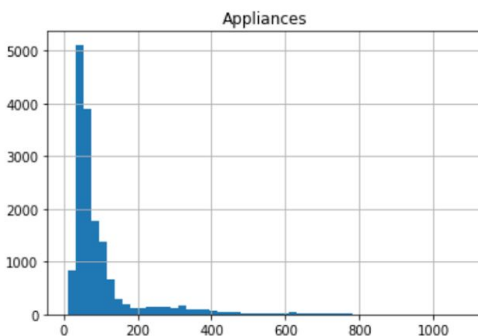
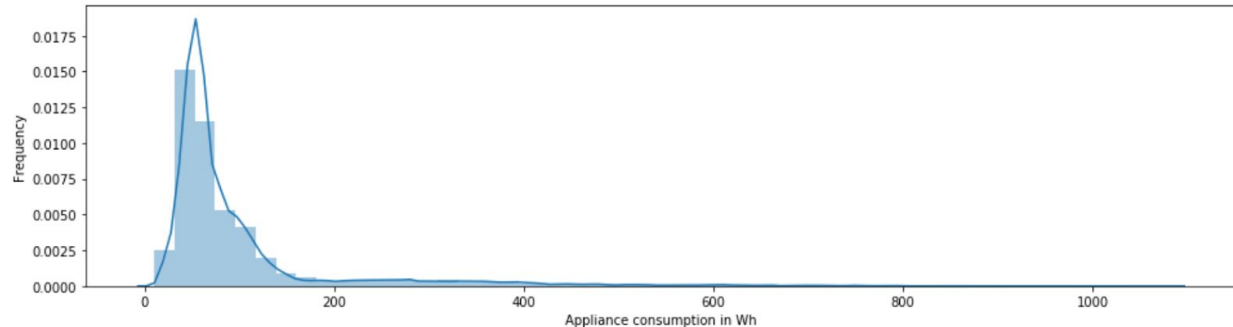
Histogram of all features:

```
feature_vars.hist(bins = 35 , figsize= (16,16)) ;
```



Target:

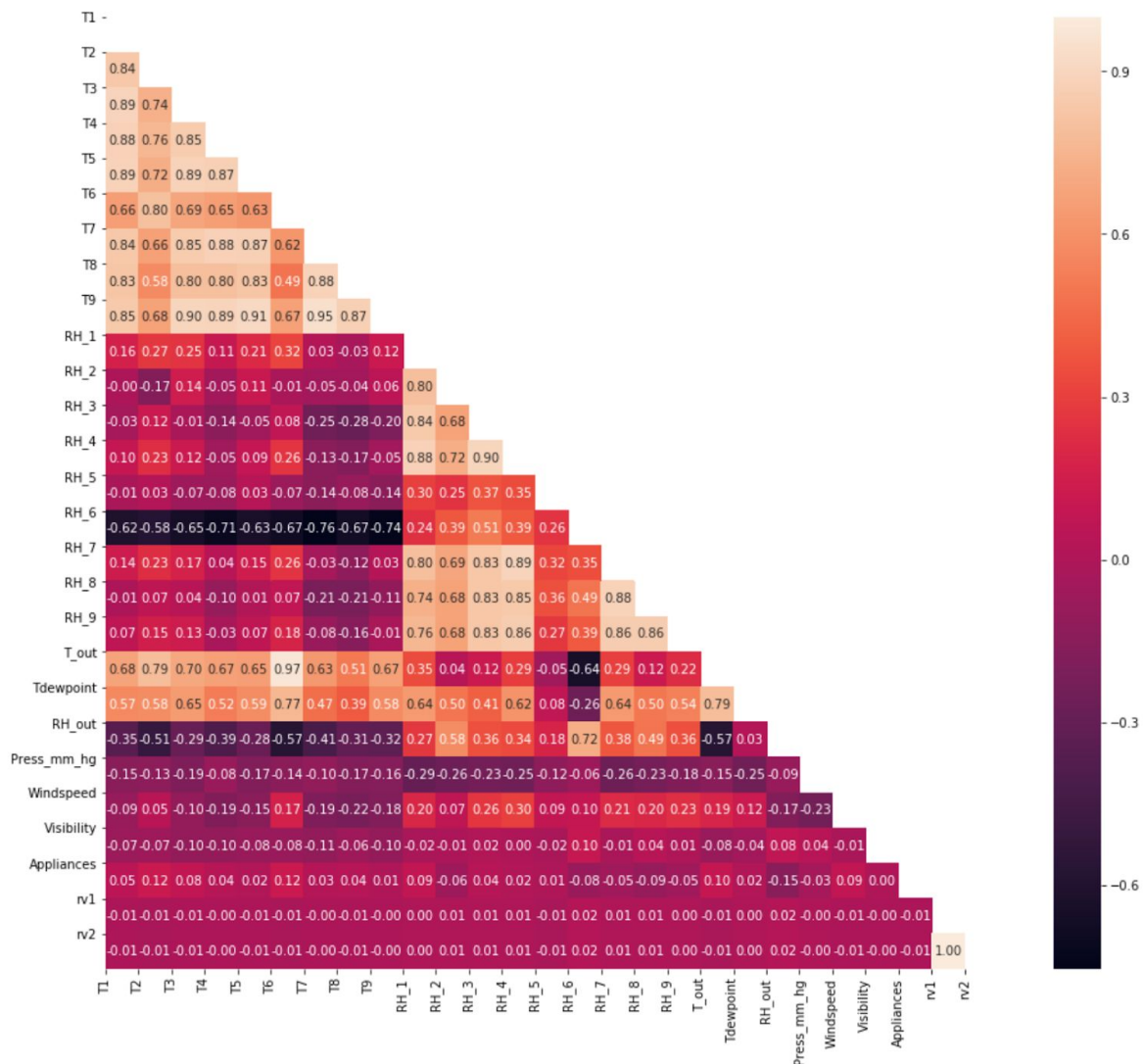
```
# Distribution of values in Appliances column
f = plt.figure(figsize=(16,4))
plt.xlabel('Appliance consumption in Wh')
plt.ylabel('Frequency')
sns.distplot(target_vars , bins = 50 ) ;
target_vars.hist(bins=50);
```



Observations on histograms:

1. All humidity values except RH_6 and RH_out follow a Normal distribution
2. Similarly, all temperature readings follow a Normal distribution except T9.
3. we can see that Visibility, Wind speed are skewed.
4. The random variables rv1 and rv2 have close values
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 with a linear relationship with the target.

Correlation 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, Four columns have a high degree of correlation with T9 - T3, T5, T7, T8 also T6 & T_Out has high correlation also . so T6 & T9 will be removed from input data.

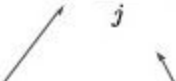
2. Weather attributes - Visibility, Tdewpoint, Press_mm_hg have low correlation
3. Humidity - RH_4 highly correlate with RH_3,7,1,9 so RH_4 will be dropped
4. Random variables have no role to play so they will be dropped
5. Visibility, Tdewpoint, Press_mm_hg have low correlation with the target variable so they will be dropped

Due to the above Observations , the dropped features will be rv1, rv2, Visibility, Press_mm_hg , Tdewpoint, T6, T9 so Number of Input Variables will be 19 (reduced from 26)

Algorithms and techniques:

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:

Ridge Regression:

$$\mathcal{L}(\beta) = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2 + \lambda \sum_j \beta_j^2$$


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 β coefficients 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.

Ridge regression uses a type of shrinkage estimator called a *ridge estimator*. Shrinkage estimators theoretically produce new estimators that are shrunk closer to the “true” population parameters. The ridge estimator is especially good at improving the least-squares estimate when multicollinearity is present.

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.

SVR is accomplished by introducing an ε -insensitive region around the function, called the ε -tube. This tube reformulates the optimization problem to find the tube that best approximates the continuous-valued function, while balancing model complexity and prediction error. More specifically, SVR is formulated as an optimization problem by first defining a convex ε -insensitive loss function to be minimized and finding the flattest tube that contains most of the training instances.

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.

KNN can be used for both classification and regression problems. The algorithm uses ‘**feature similarity**’ to predict values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.

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.

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a prediction and then the average of predictions is the output prediction

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.

Benchmark :

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

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:

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 appliance Energy Usage(10-1080). Due to the difference in ranges of features, it is possible that some features can dominate the Regression algorithm. To avoid that, 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 libraries that were used

1. sklearn.linear_model.Ridge
2. sklearn.ensemble.RandomForestRegressor
3. sklearn.ensemble.ExtraTreesRegressor
4. sklearn_neighbors
5. sklearn.svm.SVRPipeline

Refinement:

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 :

```
| from sklearn.model_selection import GridSearchCV
| param_grid = [{
|     'max_depth': [20, 30, 40, 50, 60],
|     'n_estimators': [50, 100, 150, 200, 250],
|     'max_features': ["auto", "sqrt", "log2"]
| }]
```

The R2 score improved to 0.65% from 0.61% after using the parameters suggested by GridSearchCV

Model Evaluation and Validation:

The final model is Extra Trees Regressor

1. `n_estimators`: The number of trees to be used
2. `max_features`: The number of features to be 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

The R2 score improved to 0.65% from 0.61% after using the parameters suggested by GridSearchCV

justification:

The R2 score 0.65% compared to 0.57% of benchmark model is about 14% improvement to the benchmark model

That still is not a very good model overall i think the problem still needs a bit more work to address the problem

Refiction:

1. It is very important to check the intercorrelation between all the variables in order to remove the redundant features with high correlation values.
2. While scaling data , it is useful to maintain separate copies of dataframe which can be created using index and column names of original dataframe
3. The pipeline of adding algorithms should be easy to manage
4. Seaborn and pyplot are good libraries to plot various properties of dataframe

5. 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)
6. normalizing is very important for regression models , I initially tried without it and the results were not good

Improvement:

We can try to apply regularization to the model as the model tends to over fits data

This will result in a better test scores