APPLIANCE ENERGY PREDICTION

CAPSTONE PROJECT – II
TECHNICAL DOCUMENT

MADE BY
SANKET CHOURIYA

INSTRUCTIONS

In this time of global uncertainty 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 analysing 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 this case regression analysis will be used to predict Appliance energy usage based on data collected from various sensors. My code is available on Github.

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 algorithms are used as data consist of continuous features and there is no identification of appliances in dataset.

ABOUT DATASET

Number of entries: 19735
Independent variables: 28 (11 temperatures, 10 humidity, 1 pressure, 2 random)
Dependent variable: 1 (Appliances)

All features are numerical. No categorical variables. There seems to be no null values in our data set.

TIME-SERIES ANALYSIS

```
appliance_data.head(3)
```

	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	RH_4	T5	RH_5	T6
0	2016-01-11 17:00:00	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.000000	45.566667	17.166667	55.20	7.026667
1	2016-01-11 17:10:00	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.000000	45.992500	17.166667	55.20	6.833333
2	2016-01-11 17:20:00	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.926667	45.890000	17.166667	55.09	6.560000

```
# Set datetime index due to time series analysis
appliance_data_2 = appliance_data_2.set_index('date')
appliance_data_2 = appliance_data_2.asfreq(pd.infer_freq(appliance_data_2.index))

# Setting Data in 1 hour basis
hourly_appliance_data = appliance_data_2.resample('1H').mean()
```

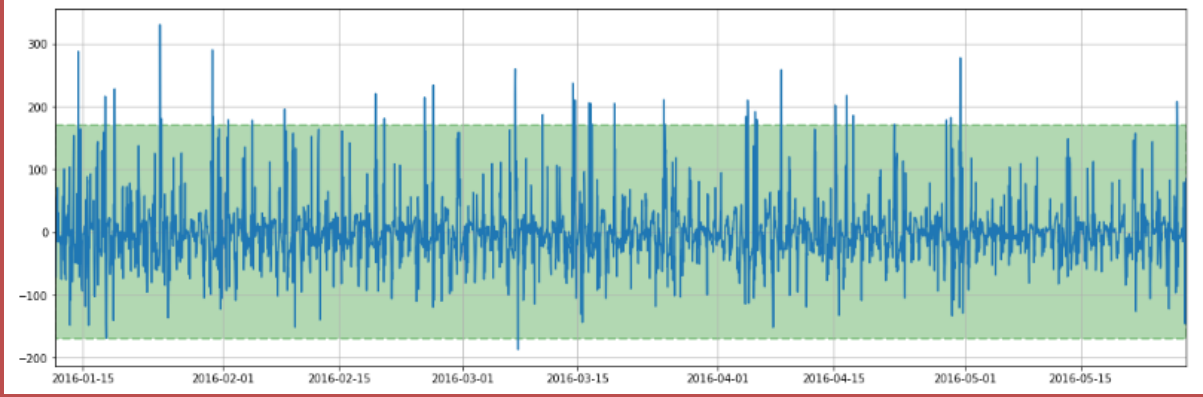
The dataset was collected by sensors placed inside the house and outside readings came from the nearby weather station. The main attributes are temperature, humidity and pressure readings. Each observation measures electricity in a 10-minute interval. The temperatures and humidity have been averaged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

Key Observations:

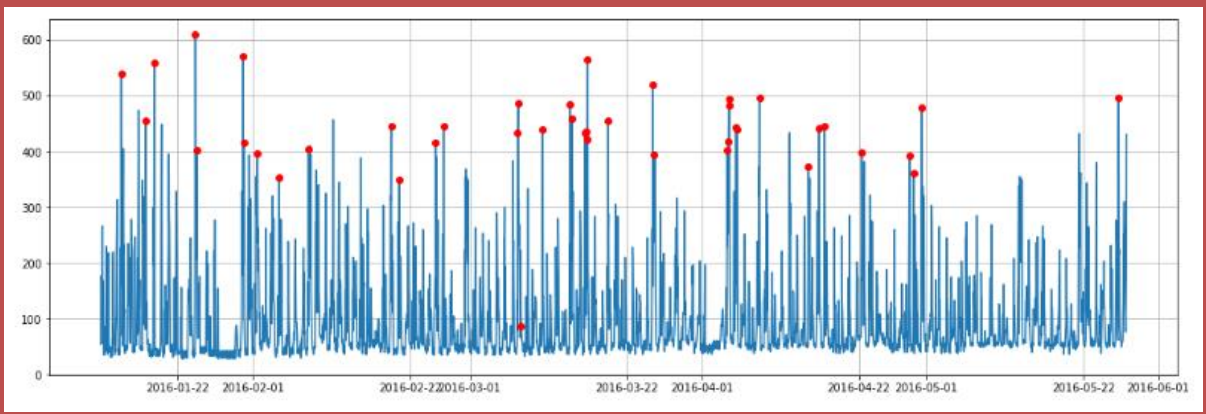
1. Date column is only used for understanding the consumption vs date time behavior thus I have considered date column as index. And whole, Data has been reshaped to hourly basis to make sense out of it and reduce noise.
2. Now for time-series analysis, 3 more columns have been added i.e. Hour, Weekday and Month.
3. Number of Independent variables at this stage – 31
4. Number of Dependent variable at this stage – 1
5. Total number of rows — 3290
6. All the features have numerical values. There are no categorical or ordinal features.
7. Number of missing values & null values = 0

ANOMALY DETECTION AND OUTLIER REMOVAL

```
resid_mu = resid.mean()
resid_dev = resid.std()

lower = resid_mu - 3.5*resid_dev
upper = resid_mu + 3.5*resid_dev

plt.figure(figsize=(18.5, 6))
plt.plot(resid)

plt.fill_between([hourly_appliance_data.index[0], hourly_appliance_data.index[-1]], lower, upper, color='g', alpha=0.3, linestyle='--', linewidth=
plt.xlim((hourly_appliance_data.index[0]), (hourly_appliance_data.index[-1]))
plt.grid()
plt.show()
```



So, the main idea of using STL in anomaly detection is to calculate the mean and standard deviation of the Residuals component of the STL decomposition and exclude all point for which residuals differs from the average by more than 3.5 times the standard deviation.



As visualized above, Outliers are considered as the less than 5% of top values of appliances' load because it is fact that recordings of power average load higher than 400Wh per hour from a house appliance are not logical. These may be false recordings caused by an instant fault on the recording devices. Thus, I have to remove these marked outliers to reduce Noise.

Observations:

1. Number of outliers detected and removed – 42
2. Portion of outlier w.r.t the dataframe – 1.28%
3. Number of Non-outliers – 3248
4. Portion of Non-outlier w.r.t the dataframe – 98.71%
5. We observe that over the sleeping hours (11 PM - 6 AM) the energy consumption of appliances is around 50 Wh.
6. After about 6 AM, energy consumption starts to rise gradually up until 11 AM (probably due to morning chores). And then gradually decreases to around 100 Wh at about 3 PM.
7. Above is the Hour-Month trend of the energy consumption of appliances. Above pattern resembles similar traits with Average Hourly consumption plot. We can say that the pattern is quite repetitive because of the same fact that as in sleeping hours (11 PM - 6 AM) the energy consumption of appliances is lowest at around 50Wh and above 100 Wh in Evening (5PM - 10 PM).
8. We observe that the energy consumption of appliances during the office hours (8 AM - 4 PM) is higher in weekends compared to the weekdays. Also, average overall consumption is higher in weekends is pretty high.

DESCRIPTIVE STATISTICS

1. Temperature

	temp_kitchen	temp_living	temp_laundry	temp_office	temp_bath	temp_outside	temp_iron	temp_teen_room	temp_parents_room	temp_station	Avg_Inside_temp
count	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000
mean	21.693382	20.346158	22.266552	20.861393	19.601422	7.905989	20.275229	22.034873	19.493246	7.413093	20.821532
std	1.609381	2.196202	2.006722	2.046574	1.843375	6.094394	2.111586	1.957988	2.018350	5.327452	1.816891
min	16.790000	16.100000	17.245000	15.100000	15.347500	-5.927685	15.410370	16.364074	14.890000	-4.961111	16.024225
25%	20.750000	18.820833	20.790000	19.544444	18.279190	3.606528	18.715179	20.790000	18.024306	3.650000	19.669887
50%	21.602778	19.996944	22.100000	20.650556	19.390000	7.251111	20.048611	22.149444	19.390000	6.891667	20.602132
75%	22.650000	21.511111	23.300069	22.111935	20.631319	11.229306	21.620119	23.390000	20.600000	10.416667	21.764605
max	26.203333	29.727778	28.975286	26.144762	25.506389	28.136619	25.926667	27.187778	24.500000	25.933333	25.993426

2. Humidity

	humid_kitchen	humid_living	humid_laundry	humid_office	humid_bath	humid_outside	humid_iron	humid_teen_room	humid_parents_room	humid_station	AVg_Inside
count	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000	3248.(
mean	40.243854	40.420735	39.233585	39.030839	50.965958	54.615172	35.408276	42.956861	41.558216	79.796541	41.2
std	3.934956	4.056864	3.242154	4.334516	8.637317	31.072614	5.108270	5.212226	4.144927	14.826053	3.9
min	27.509167	21.010000	29.700556	28.715571	30.188611	1.000000	23.340278	29.738611	29.218889	25.250000	31.(
25%	37.328333	37.914351	36.874583	35.528611	45.491667	30.202250	31.508405	39.144444	38.528125	70.500000	38.3
50%	39.637014	40.495556	38.530278	38.401389	49.271019	55.123750	34.925069	42.416488	40.845681	83.833333	40.9
75%	43.060833	43.281731	41.750833	42.172222	53.894583	83.305278	39.028562	46.579479	44.355278	91.583333	44.7
max	53.980139	53.914975	49.472222	50.747222	94.884074	99.900000	51.191296	58.707315	53.140000	100.000000	52.9

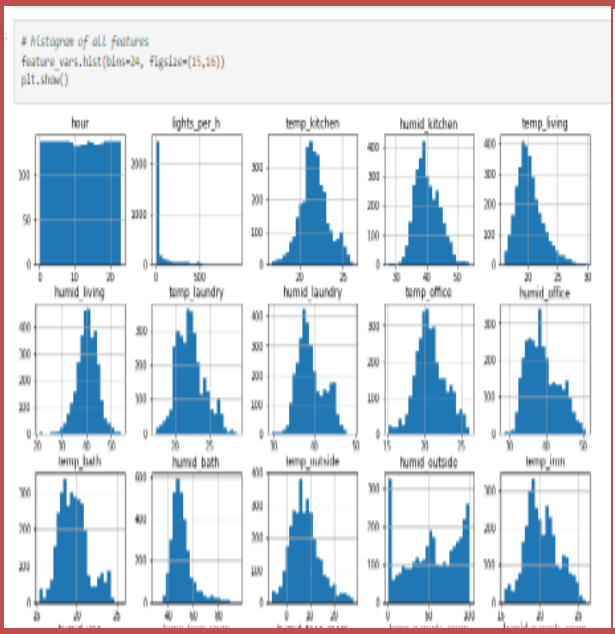
3. Weather

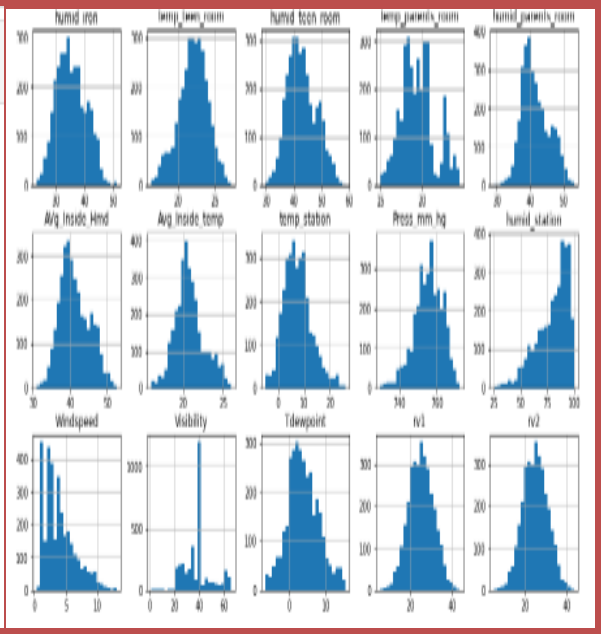
	lights_per_h	Tdewpoint	Press_mm_hg	Windspeed	Visibility
count	3248.000000	3248.000000	3248.000000	3248.000000	3248.000000
mean	57.242406	3.770138	755.512762	4.030763	38.338696
std	120.172537	4.196575	7.405786	2.433438	11.234136
min	0.000000	-6.475000	729.383333	0.416667	1.000000
25%	0.000000	0.947917	750.916667	2.000000	31.833333
50%	0.000000	3.412500	756.100000	3.583333	40.000000
75%	36.666667	6.568750	760.925000	5.416667	40.000000
max	930.000000	15.250000	772.258333	13.000000	66.000000

Feature ranges:

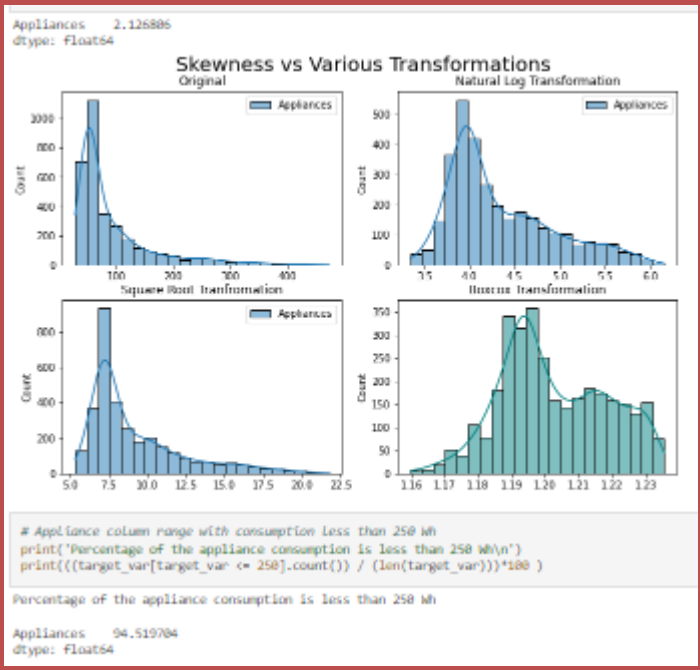
1. Temperature: -5 to 26 deg
2. Humidity: 25 to 100 %
3. Windspeed: 0 to 13 m/s
4. Visibility: 1 to 66 km
5. Pressure: 729 to 772 mm Hg
6. Appliance Energy Usage: 0 to 589 Wh

INDEPENDENT VARIABLE DISTRIBUTION





DEPENDENT VARIABLE DISTRIBUTION

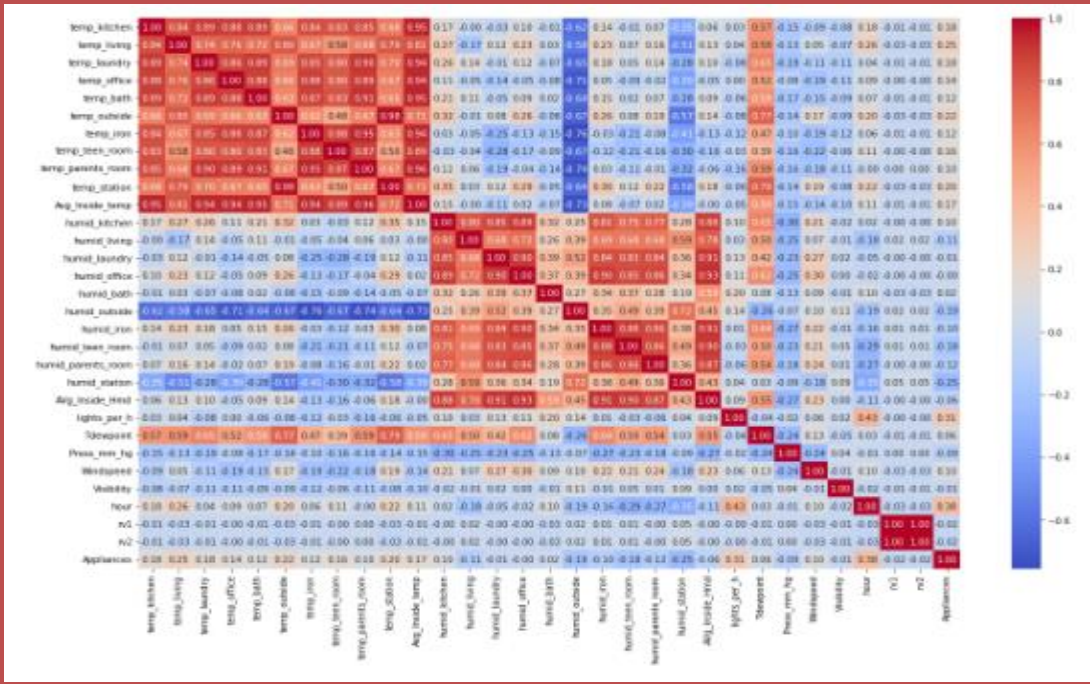


Observations based on distribution plot:

1. All features values except 'lights_per_h' ,'humid_outside' and 'Visibility' follow a Normal distribution.
2. Similarly, all temperature readings follow a Normal distribution except for 'temp_parents_room' which is sort of unstable near tail end.
3. The random variables rv1 and rv2 have more or less the same values for all the recordings.
4. No column has a distribution like the target variable Appliances.
5. Appliance - The distribution is right skewed and majority of appliances uses less than 250 Wh of energy. With the maximum consumption of 1080 Wh , there will be outliers in this column and there are small number of cases where consumption is very high.
6.
7. After applying Log, Square root and Reciprocal Transformation we can see that the Log transformation works the best to remove the skewness of the Target Variable.

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

CORRELATION METRIX



OBSERVATIONS:

1. Humidity outside have a strong negative correlation with temperature levels as already discussed.

2. Apart from that we observe that a couple features such as humidity at station, temperature outside the building and temperature in the living room have a comparatively high absolute correlation (above 0.12) with Appliances energy consumption.

3. Four columns have a high degree of correlation with *temp_parent_room* - *temp_laundry, temp_bath, temp_iron, temp_teen_room* also *temp_outside & temp_station* has high correlation (both temperatures from outside). Hence *temp_outside & temp_parents_room* can be removed from training set as information provided by them can be provided by other fields.

4. Tdewpoint shows a high correlation with most of the inside temperature and humidity level features than any other weather parameters. Pressure, windspeed and visibility show little to no correlation.

5. The random variables rv1, rv2 and Visibility, Tdewpoint, Press_mm_hg has low correlation with the target variable.

DATA PREPROCESSING AND SCALING

TRAIN-TEST SPLIT: It Ratio of 80:20 has been kept for train and test set respectively.

SCALING: The feature set has data in varying ranges. To remove domination of curtain features in the Regression algorithm, all features need to be scaled. Thus, I have used StandardScaler to remove mean and scale the unit variance.

EVALUATION MATRICS: Following are the main evaluation metrics used for models,

- Root Mean Squared Error (RMSE)
- R2

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:**

   Lasso Regression

   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.

   Polynomial Regression

   Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth degree polynomial. The Polynomial Regression equation is given below:

$$y= b_0+b_1x_1+ b_2x_1^2+ b_2x_1^3+..... b_nx_1^n$$

2. **Tree based Regression models**

   Random Forest Regressor

   A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

   Extra Tree Regressor

   Extra-trees differ from classic decision trees in the way they are built. When looking for the best split to separate the samples of a node into two groups, random splits are drawn for each of the max_features randomly selected features and the best split among those is chosen. When max_features is set 1, this amounts to building a totally random decision tree.

3. **Gradient Boosting Machines**

   XGBoost

   XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

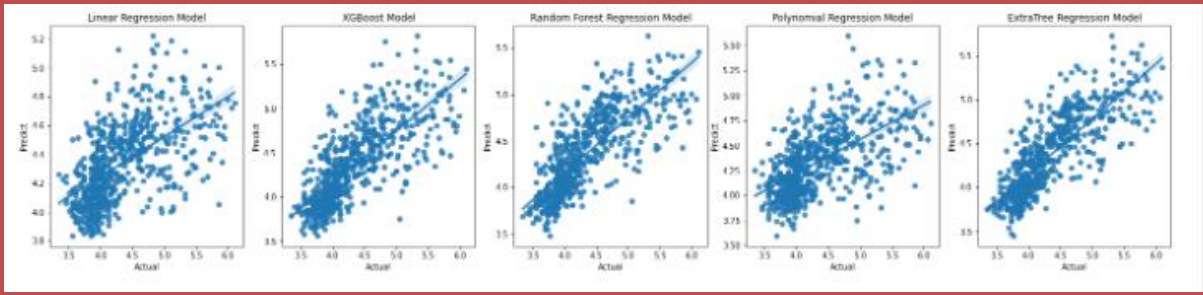
IMPLIMENTATION

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

- sklearn.preprocessing.PolynomialFeatures
- sklearn.linear_model.Lasso
- sklearn.ensemble.RandomForestRegressor
- sklearn.ensemble.ExtraTreesRegressor
- import xgboost

MODEL COMPARISON

	Model	Test RMSE	Test R2 Square	Train RMSE	Train R2 Square
0	Lasso Regression	0.474316	0.327783	4.847785e-01	0.309533
1	Polynomial Regression	0.463741	0.357423	4.667119e-01	0.360039
2	Random Forest Regressor	0.360689	0.611278	1.372019e-01	0.944693
3	XGBoost regression	0.377674	0.573804	2.546554e-01	0.809471
4	ExtraTrees Regressor	0.339331	0.655950	6.134726e-15	1.000000



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

HYPER PARAMETER TUNING

I have used RandomizedSearch cross validation function of the sklearn.model selection library as it is much faster than GridSearchCV.

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
4. **min_samples_leaf:** The minimum number of samples required to be at a leaf node.

This will try out 4 * 2 * 4 * 2 = 64 combinations in 5 Folds under current settings which is totalling 320 random combnations. We can fit the model, display the best hyper parameters, and evaluate performance:

```
Fitting 5 folds for each of 64 candidates, totalling 320 fits
RandomizedSearchCV(cv=5, estimator=ExtraTreesRegressor(random_state=0),
                   n_iter=120, n_jobs=-1,
                   param_distributions={'max_depth': [225, 350, 475, 550],
                                        'max_features': ['auto', 'log2'],
                                        'min_samples_leaf': [2, 3],
                                        'n_estimators': [450, 600, 750, 900]},
                   random_state=0, scoring='r2', verbose=2)
```

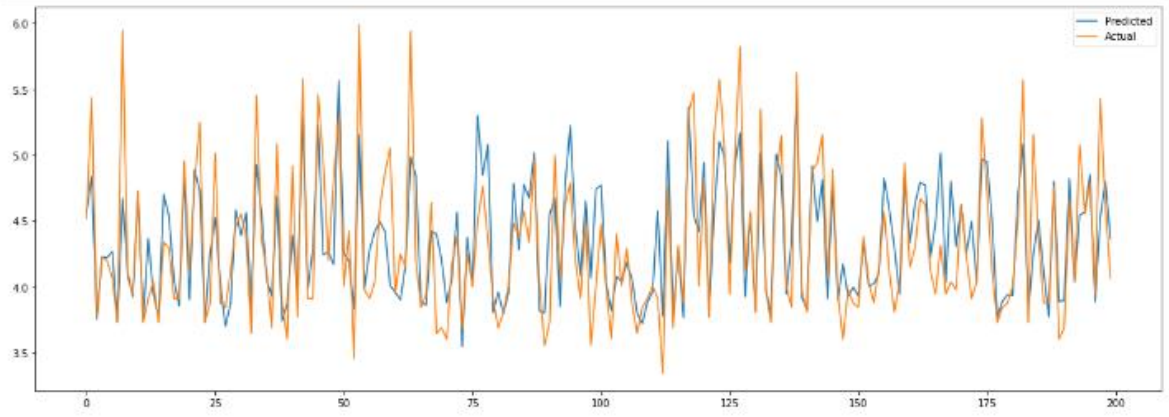
```
# Checking performance
ETR_RSCV_test_pred = ETF_random_SCV.best_estimator_.predict(X_test_sc)
evaluate_model(best_para, y_test, ETR_RSCV_test_pred)
```

```
ExtraTreesRegressor(max_depth=225, min_samples_leaf=2, n_estimators=600,
                    random_state=0)

Average Error          :  0.57181 degrees
Variance score R^2     :  65.52 %
RMSE                   :  0.33971
Accuracy               :  86.93 %
```

PREDICTIBILITY

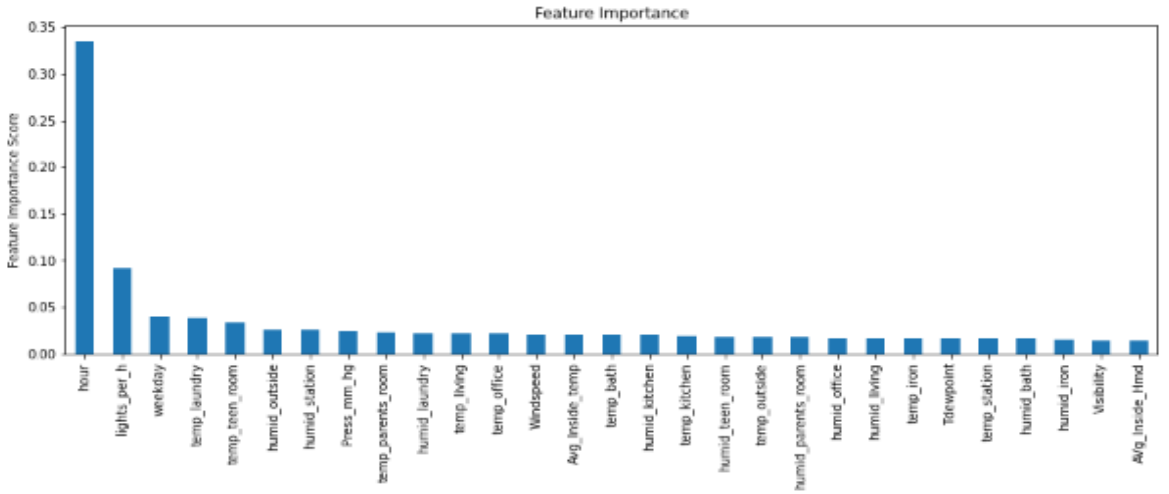
There little to no improvement has been absorbed aven after hyper tuning.

```
make_plot(y_test[:200], ETR_RSCV_test_pred[:200])
```



FEATURE SELECTION FOR IMPROVEMENT

```
#Feature importance for top 50 predictors
predictors = [features for features in X.columns]

feature_imp = pd.Series(ETF_random_SCV.best_estimator_.feature_importances_, predictors).sort_values(ascending=False)

plt.rcParams['figure.figsize'] = 15, 5
feature_imp.plot(kind='bar', title='Feature Importance')
plt.ylabel('Feature Importance Score')
plt.show()
```




```
# Reduce test & training set to top 10 feature set
train_important_feature = X_train[['hour','lights_per_h','temp_parents_room','temp_laundry','temp_teen_room',
                                   'Press_mm_hg','humid_laundry','humid_bath','temp_bath','temp_office']]
test_important_feature = X_test[['hour','lights_per_h','temp_parents_room','temp_laundry','temp_teen_room',
                                 'Press_mm_hg','humid_laundry','humid_bath','temp_bath','temp_office']]
```

```
# Checking performance
HT_test_pred = cloned_model.predict(test_important_feature)
evaluate_model(best_para, y_test, HT_test_pred)
```

```
ExtraTreesRegressor(max_depth=225, min_samples_leaf=2, n_estimators=600,
                    random_state=0)

Average Error        :  0.57218 degrees
Variance score R^2   :  62.38 %
RMSE                 :  0.35485
Accuracy             :  86.93 %
```

Performed Feature selection on the basis of importance in the predictive relationship with the target veriable. Top 10 feature where selected for the model and yet the model performance decrease by ~3% compared to Tuned model.

CONCLUTION

- The best Algorithm to use for this dataset Extra Trees Regressor.
- Although, Extra trees regressor over-fits our training data with a R2 score of 1, it also gives, by far, the best performance on test set as well compared to other models, with a R2 score of 0.653
- Hyper parameter tuning of our model doesn't have any significant impact on our test results. We were able to improve the test R2 score results by less than 0.01
- The final model had 22 features.
- Feature reduction was not able to add to better R2 score.