Team 10:
Jai Soni
Pramod Nagare
Saurabh Kulkarni

Assignment 2: Machine learning with Energy datasets

Part 2 : Exploratory Data Analysis

Dataset Overview:

- IOT Dataset for appliance energy consumption.
- Data available for 11/01/2016 to 27/05/2016 with 10 min interval.
- Number of rows: 19735
- Total number of columns: 29
- We have 0% missing values in our dataset.
- Date feature has a data type Object. We have numerical dataset.
- Various temperature and humidity features available for each record.
- Also, windspeed, visibility and pressure readings are given.

Variable Description

date time year-month-day hour:minute:second

Appliances, energy use in Wh

lights, energy use of light fixtures in the house in Wh

T1, Temperature in kitchen area, in Celsius

RH_1, Humidity in kitchen area, in %

T2, Temperature in living room area, in Celsius

RH_2, Humidity in living room area, in %

T3, Temperature in laundry room area

RH_3, Humidity in laundry room area, in %

T4, Temperature in office room, in Celsius

RH_4, Humidity in office room, in %

T5, Temperature in bathroom, in Celsius

RH_5, Humidity in bathroom, in %

T6, Temperature outside the building (north side), in Celsius

RH_6, Humidity outside the building (north side), in %

T7, Temperature in ironing room , in Celsius

RH_7, Humidity in ironing room, in %

T8, Temperature in teenager room 2, in Celsius

RH_8, Humidity in teenager room 2, in %

T9, Temperature in parents room, in Celsius

RH_9, Humidity in parents room, in %

To, Temperature outside (from Chièvres weather station), in Celsius

Pressure (from Chièvres weather station), in mm Hg

RH_out, Humidity outside (from Chièvres weather station), in %

Windspeed (from Chièvres weather station), in m/s

Visibility (from Chièvres weather station), in km

Tdewpoint (from Chièvres weather station), °C

rv1, Random variable 1, nondimensional

rv2, Rnadom variable 2, nondimensional

Temperature Variable Analysis

- Almost all temperature feature ranges between 14.89°C to 29.86°C
- Exceptions:
- T6, Tdewpoint and T_out is ranges between -6.6°C to 28.9°C.
- Reason: T6 and T_out are temperature outside the building.
- All the temperature features are correlated with each other.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T_out	Tdewpoint
T1	1.000000	0.836834	0.892402	0.877001	0.885247	0.654769	0.838705	0.825413	0.844777	0.682846	0.571309
T2	0.836834	1.000000	0.735245	0.762066	0.720550	0.801186	0.663660	0.578191	0.675535	0.792255	0.582602
T3	0.892402	0.735245	1.000000	0.852778	0.888169	0.686882	0.847374	0.795283	0.901324	0.699417	0.645886
T4	0.877001	0.762066	0.852778	1.000000	0.871813	0.652350	0.877763	0.796256	0.889439	0.663478	0.519471
T5	0.885247	0.720550	0.888169	0.871813	1.000000	0.629161	0.870624	0.824981	0.911055	0.651321	0.588362
T6	0.654769	0.801186	0.686882	0.652350	0.629161	1.000000	0.619085	0.482836	0.667177	0.974787	0.764242
T7	0.838705	0.663660	0.847374	0.877763	0.870624	0.619085	1.000000	0.882123	0.944776	0.631293	0.466625
T8	0.825413	0.578191	0.795283	0.796256	0.824981	0.482836	0.882123	1.000000	0.869338	0.502842	0.391810
T9	0.844777	0.675535	0.901324	0.889439	0.911055	0.667177	0.944776	0.869338	1.000000	0.668220	0.581483
T_out	0.682846	0.792255	0.699417	0.663478	0.651321	0.974787	0.631293	0.502842	0.668220	1.000000	0.790661
Tdewpoint	0.571309	0.582602	0.645886	0.519471	0.588362	0.764242	0.466625	0.391810	0.581483	0.790661	1.000000

Humidity Variable Analysis

- Almost all humidity features ranges between 20.46% to 63.36%
- Exceptions:
- RH_5 & RH_out: 24% to 100%
- RH_6: 1% to 99.9%
- Reason:
- RH_5: Humidity recorded at the bathroom.
- RH_6 and RH_out: Humidity recorded at outside of building.
- RH_1, RH_2, RH_3, RH_4, RH_7, RH_8, RH_9 has significant correlation with each other and very less correlation to other humidity features.

	RH_1	RH_2	RH_3	RH_4	RH_5	RH_6	RH_7	RH_8	RH_9	RH_out
RH_1	1.000000	0.797535	0.844677	0.880359	0.303258	0.245126	0.801122	0.736196	0.764001	0.274126
RH_2	0.797535	1.000000	0.678326	0.721435	0.250271	0.389933	0.690584	0.679777	0.676467	0.584911
RH_3	0.844677	0.678326	1.000000	0.898978	0.375422	0.514912	0.832685	0.828822	0.833538	0.356192
RH_4	0.880359	0.721435	0.898978	1.000000	0.352591	0.392178	0.894301	0.847259	0.856591	0.336813
RH_5	0.303258	0.250271	0.375422	0.352591	1.000000	0.263797	0.325808	0.359840	0.272197	0.185941
RH_6	0.245126	0.389933	0.514912	0.392178	0.263797	1.000000	0.357222	0.489580	0.391943	0.718587
RH_7	0.801122	0.690584	0.832685	0.894301	0.325808	0.357222	1.000000	0.883984	0.858686	0.378519
RH_8	0.736196	0.679777	0.828822	0.847259	0.359840	0.489580	0.883984	1.000000	0.855812	0.487355
RH_9	0.764001	0.676467	0.833538	0.856591	0.272197	0.391943	0.858686	0.855812	1.000000	0.359377
RH_out	0.274126	0.584911	0.356192	0.336813	0.185941	0.718587	0.378519	0.487355	0.359377	1.000000

Weather variable analysis

- Pressure data ranges from 729.3 mmHg to 772.30 mmHg
- Windspeed recorded ranges from 0 to 14 m/s
- Visibility data ranges from 1 to 66 kms.
- These features has very low correlation among each other.

	T_out	Tdewpoint	RH_out	Press_mm_hg	Windspeed	Visibility
T_out	1.000000	0.790661	0.574197	0.143249	0.192936	0.077367
Tdewpoint	0.790661	1.000000	0.036506	0.244098	0.125972	0.042190
RH_out	0.574197	0.036506	1.000000	0.092017	0.176458	0.083125
Press_mm_hg	0.143249	0.244098	0.092017	1.000000	0.235032	0.040315
Windspeed	0.192936	0.125972	0.176458	0.235032	1.000000	0.007516
Visibility	0.077367	0.042190	0.083125	0.040315	0.007516	1.000000

Random variable analysis

- Features rv1 and rv2, 2 random variable which was just added to validate the performance of the feature selections models.
- We found the perfect correlation between these two variables.
- Also, it is proved that the both columns are identical, giving redundancy in the dataset.

```
energy[["rv1","rv2"]].corr().abs()
```

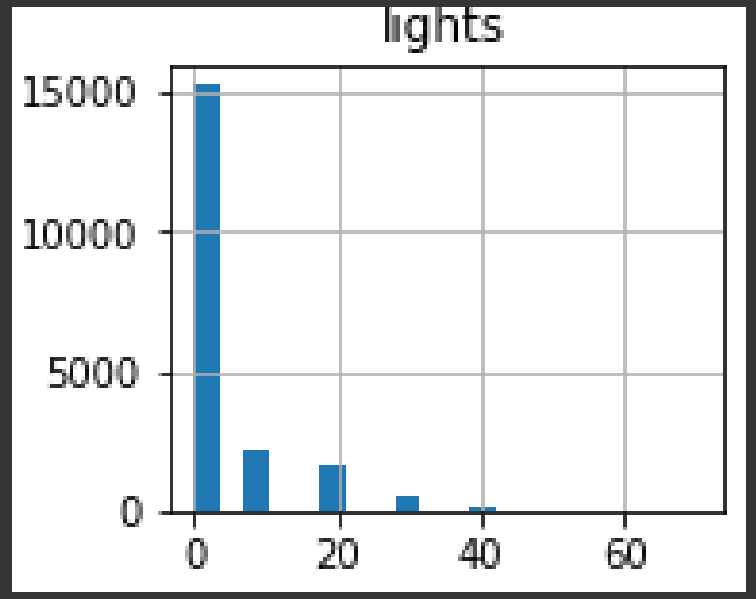
	rv1	rv2
rv1	1.0	1.0
rv2	1.0	1.0

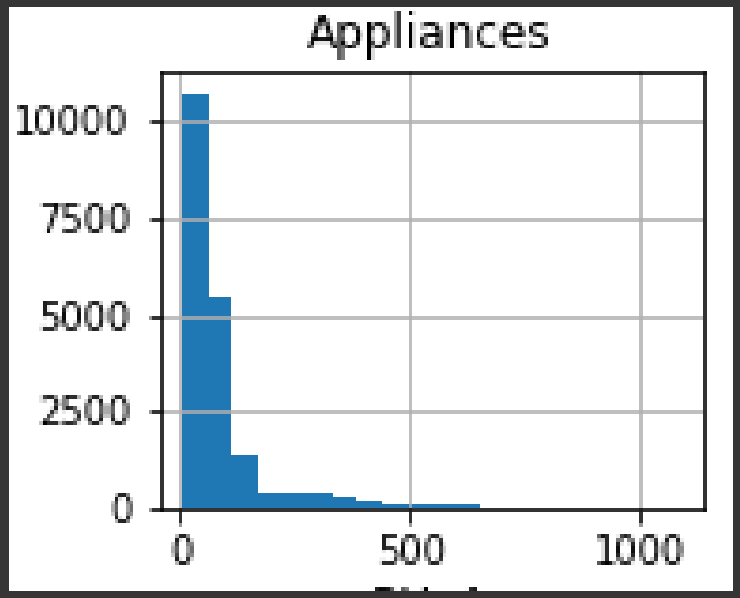
```
(energy['rv1']-energy['rv2']==0).count() - energy.shape[0]
```

```
0
```

Target variable analysis

- We can see that Maximum appliance consumption is 1080 Wh.
- However, we have 90.21% of data for which the appliance energy consumption is less than 200 Wh.
- Similarly, the maximum lights consumption is 70 Wh.
- But 77.28% of data shows that the lights were off.
- This observations show that there are less number of cases when the appliance energy consumption were high.





High Correlation Analysis (> 0.88)

- rv1 and rv2 shows perfect correlation. And as derived in above observations, rv2 is redundant feature.
- Feature T9 shows high correlation with T3, T4, T5, T7.
- Thus T9 can be consider as a overhead in the dataset.
- Similarly, T5 has a high correlation with the T9, T3 and T1 which thus acts added redundancy.
- Also, we know T6 and T_out are exterior temperature features, and shows high degree of correlations.
- We can get rid of either of the above two features to lower the data redundancy.

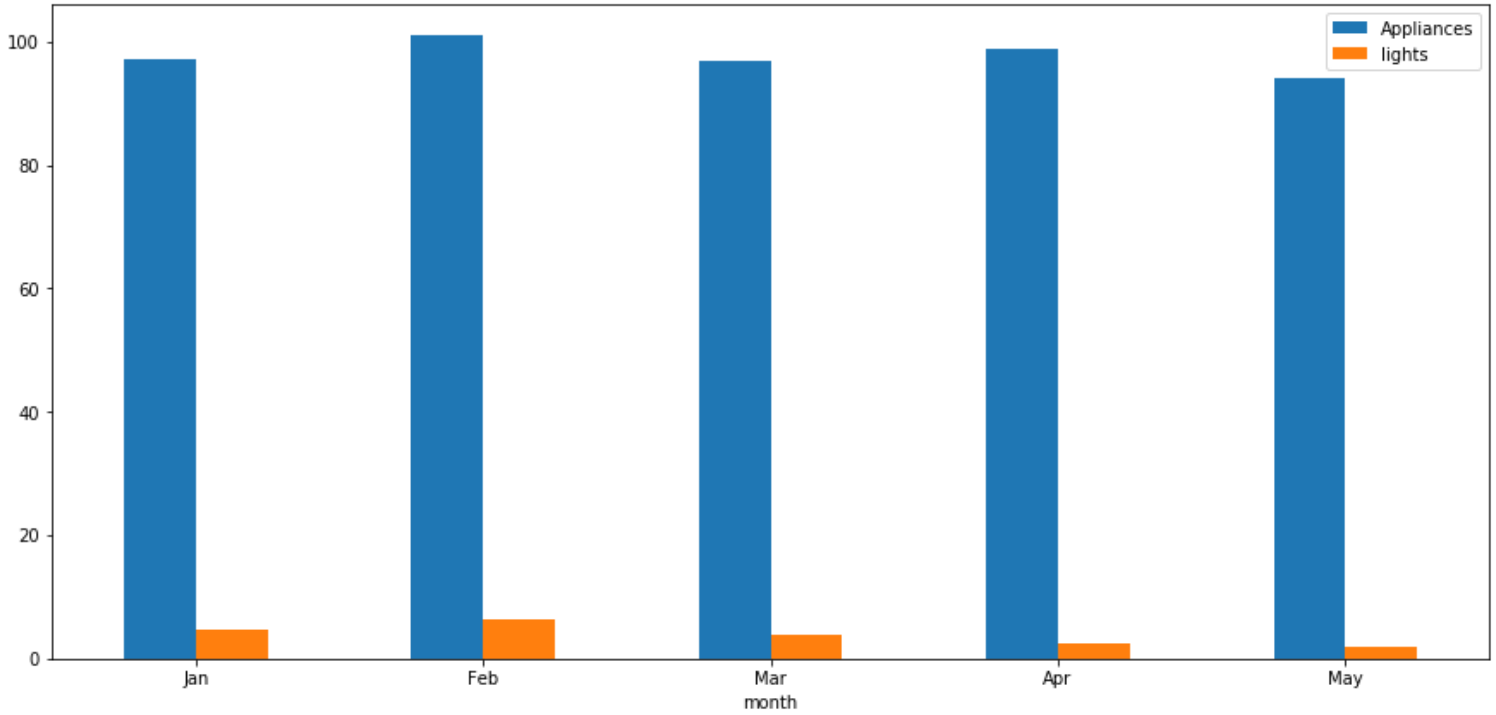
```
Column pair : T1, T3
Correlation coefficient : 0.89240228511056
Column pair : T1, T5
Correlation coefficient : 0.88524687342665
Column pair : RH_1, RH_4
Correlation coefficient : 0.88035853518919
Column pair : T3, T5
Correlation coefficient : 0.88816891277498
Column pair : T3, T9
Correlation coefficient : 0.90132358508256
Column pair : RH_3, RH_4
Correlation coefficient : 0.89897829028307
Column pair : T4, T9
Correlation coefficient : 0.88943915114025
Column pair : RH_4, RH_7
Correlation coefficient : 0.89430124888178
Column pair : T5, T9
Correlation coefficient : 0.91105511780679
Column pair : T6, T_out
Correlation coefficient : 0.97478669006645
Column pair : T7, T8
Correlation coefficient : 0.88212322628440
Column pair : T7, T9
Correlation coefficient : 0.94477642356874
Column pair : RH_7, RH_8
Correlation coefficient : 0.88398396431770
Column pair : rv1, rv2
Correlation coefficient : 1.0
```

Time series analysis

- As we have time series data, we can drill down our observation for time specific analysis for the appliance energy consumption.
- For that we need to add some new features which will help us to get more insight.
- We will add some more derived features like month, time, Day of week, NSM (minutes from midnight), DOY (Day of year), Only_Date
- Also we need to scale our features to get effective visual graphs.

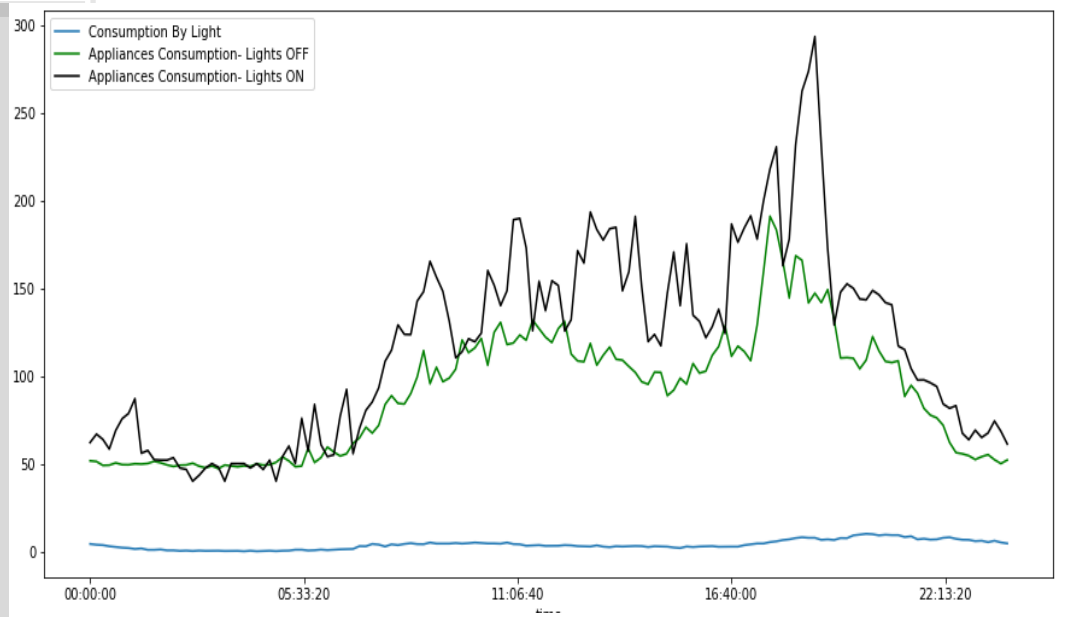
Appliance energy consumption over months

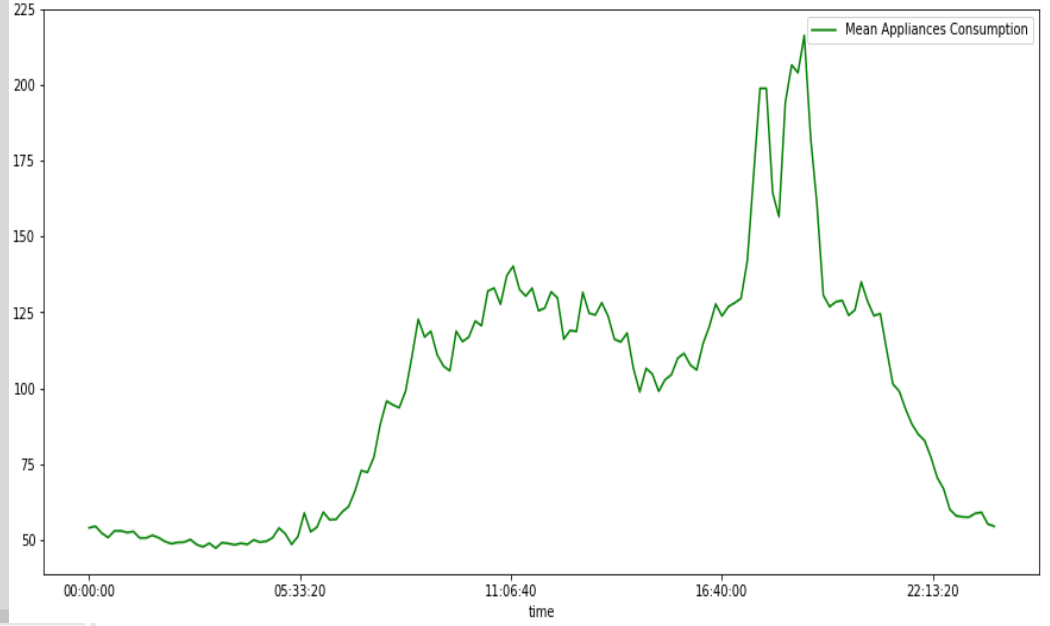
- From the month January to February the appliance energy consumption increases and then decreases in the month of March.
- Again the rise in the appliance energy increase in April and further drop in the month of May.
- Similarly, there was a rise in the light usage for the month February and then gradual decrease in March, April and May.



Appliance energy consumption with light status

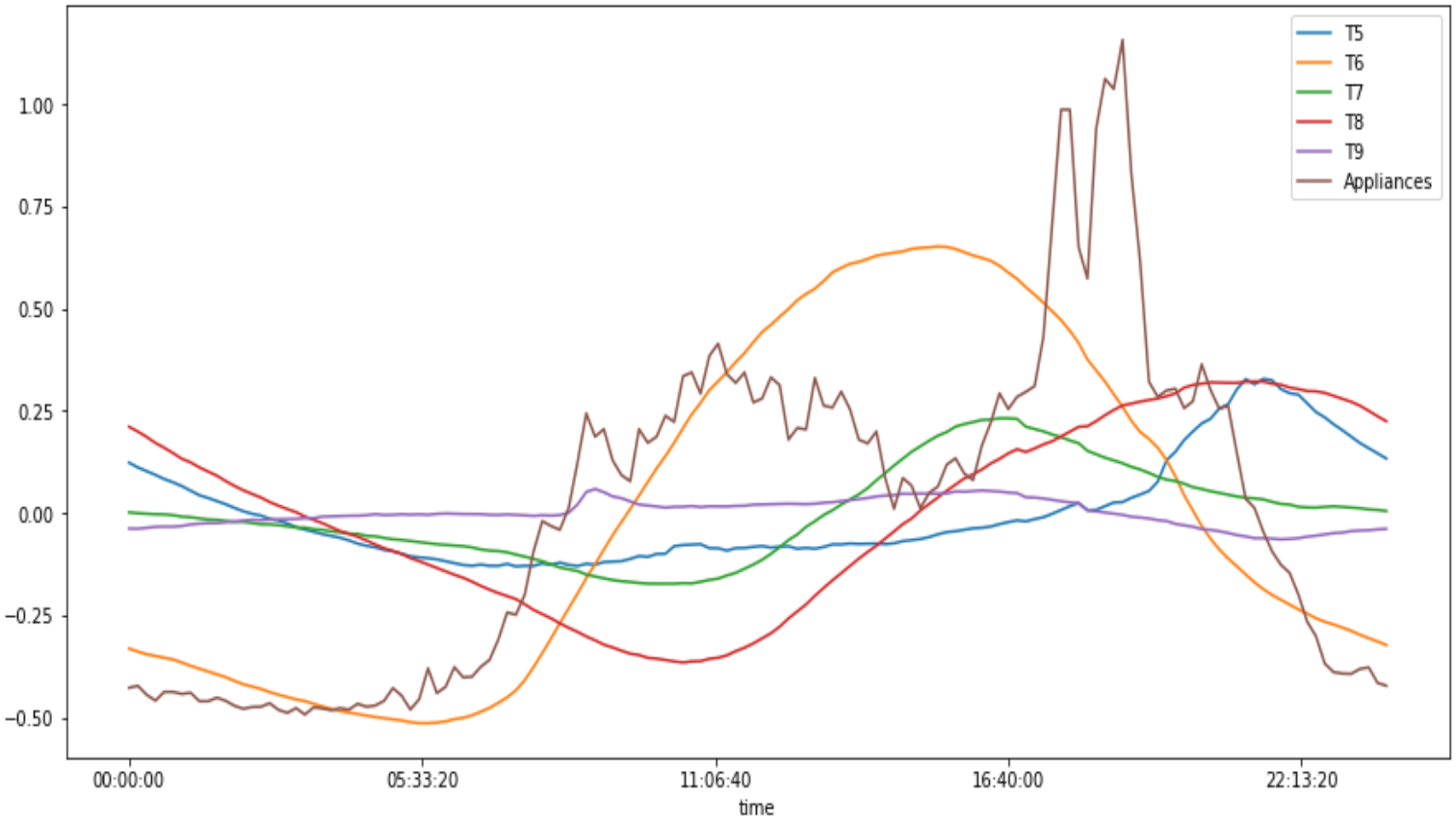
- (Average) Appliance energy utilizations is low when lights are OFF.
- (Average) Appliance energy utilizations is high when lights are ON.
- Also, Appliance energy utilizations follows a trend with respect to time.
- For the night time (00:00:00 to 06:00:00) the energy consumption is lowest.
- From the time 06:00:00 energy consumption started rising, and got steady till time 17:20:00
- Then the peak hours started till 19:00:00
- After that, the energy utilization started decreasing.





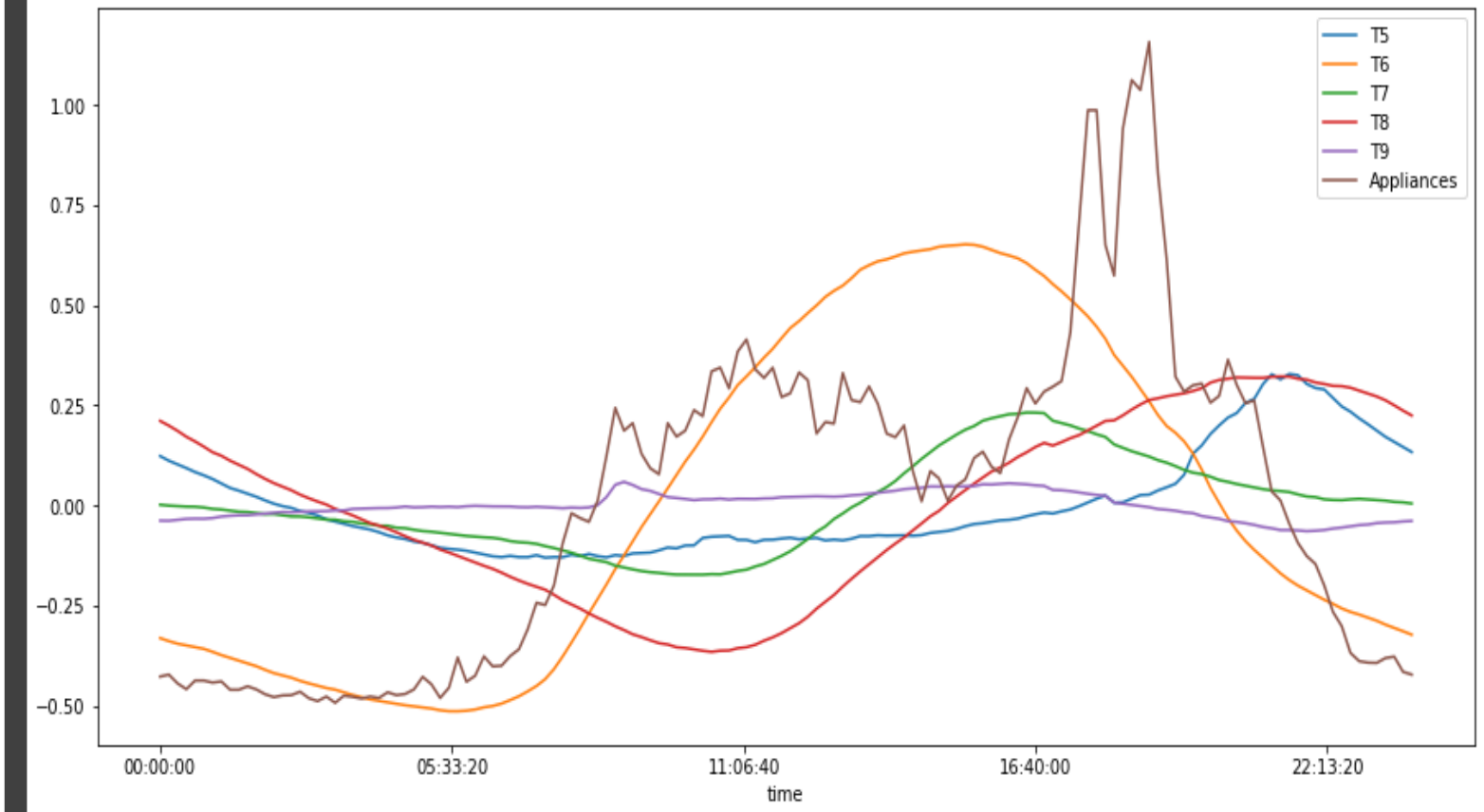
1st Floor Temperature Analysis

- Features T1, T2, T3, T4 are the temperature belongs to the same floor but for different rooms.
- From the above graph we can say that, Features T1, T2, T3, T4 Follows each other.
- However, among all these features, T2 shows significant follow-up with the Appliances energy usage.



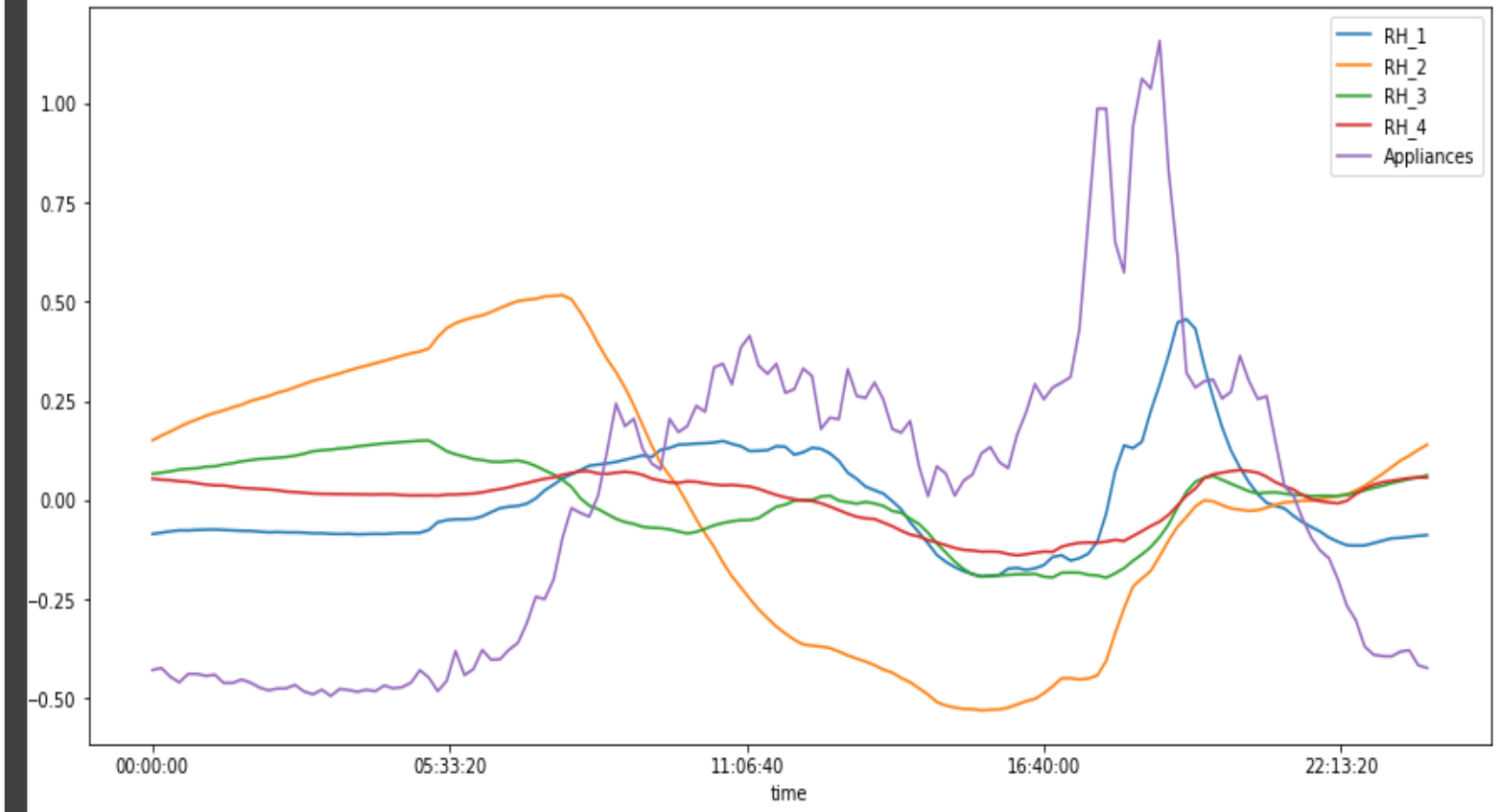
2nd Floor Temperature Analysis

- Features T5, T6, T7, T8 and T9 are the temperature for the 2nd floor of the house.
- Among all these features T6 shows the significant pattern for the Appliance energy utilization.



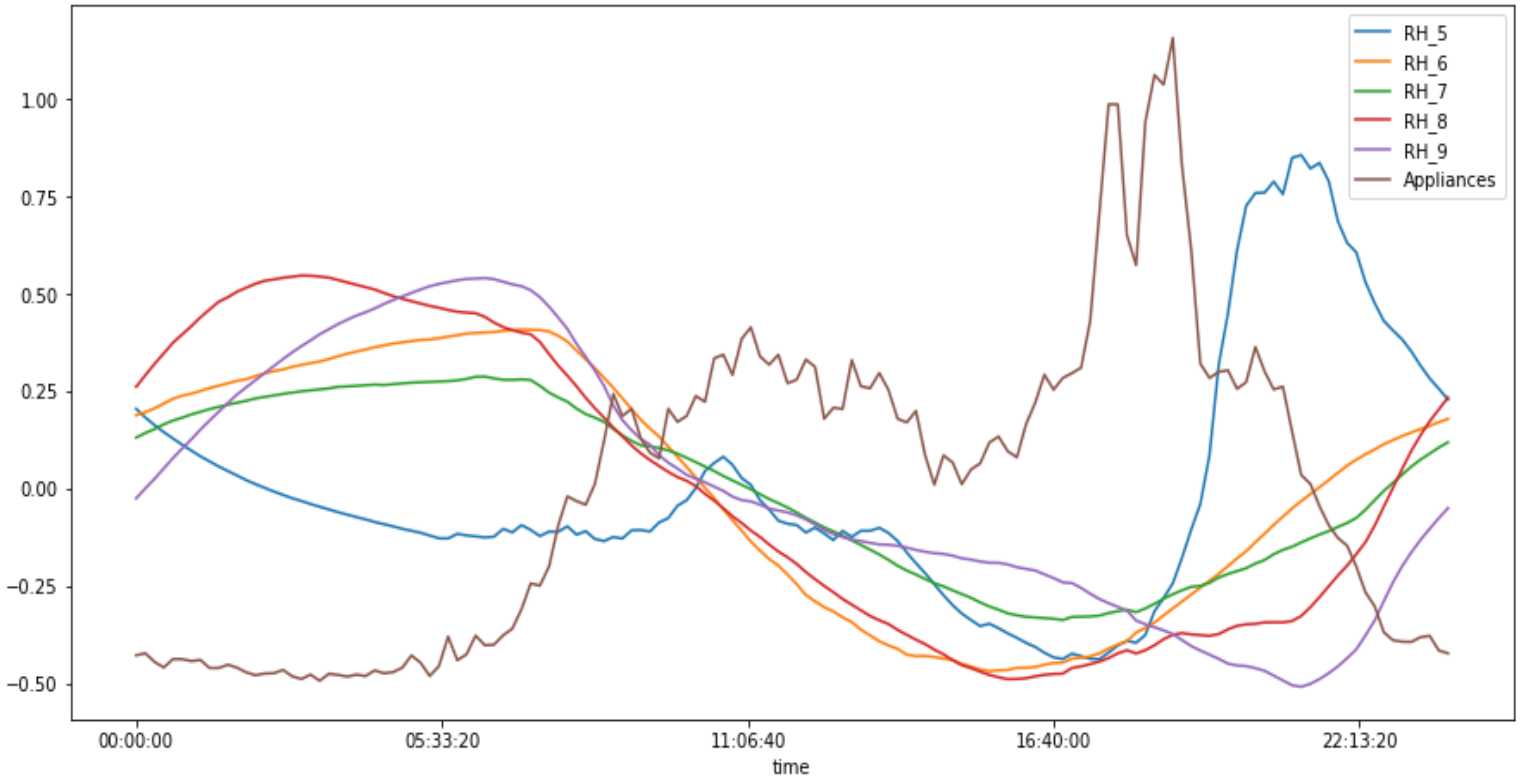
1st Floor Humidity Analysis

- Plot shows the humidity for the different rooms on the 1st floor of the house.
- All these features shows the significant pattern for the appliance energy consumption.



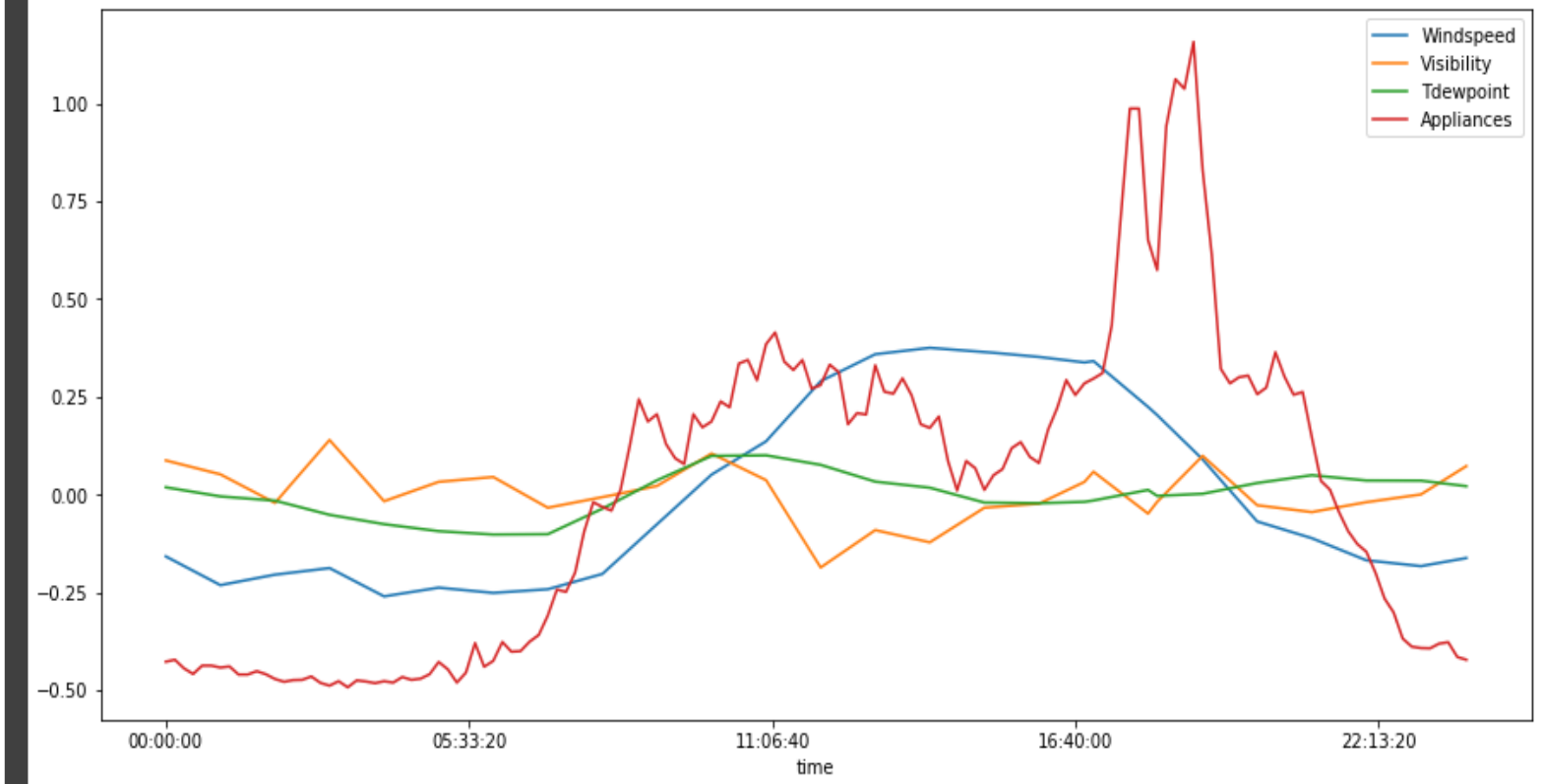
2nd Floor Humidity Analysis

- The above plot shows the curve for the appliances energy utilization over time.
- Among all these features RH_6, RH_7, RH_8 and RH_9 shows the great follow-up for the appliance energy utilization.



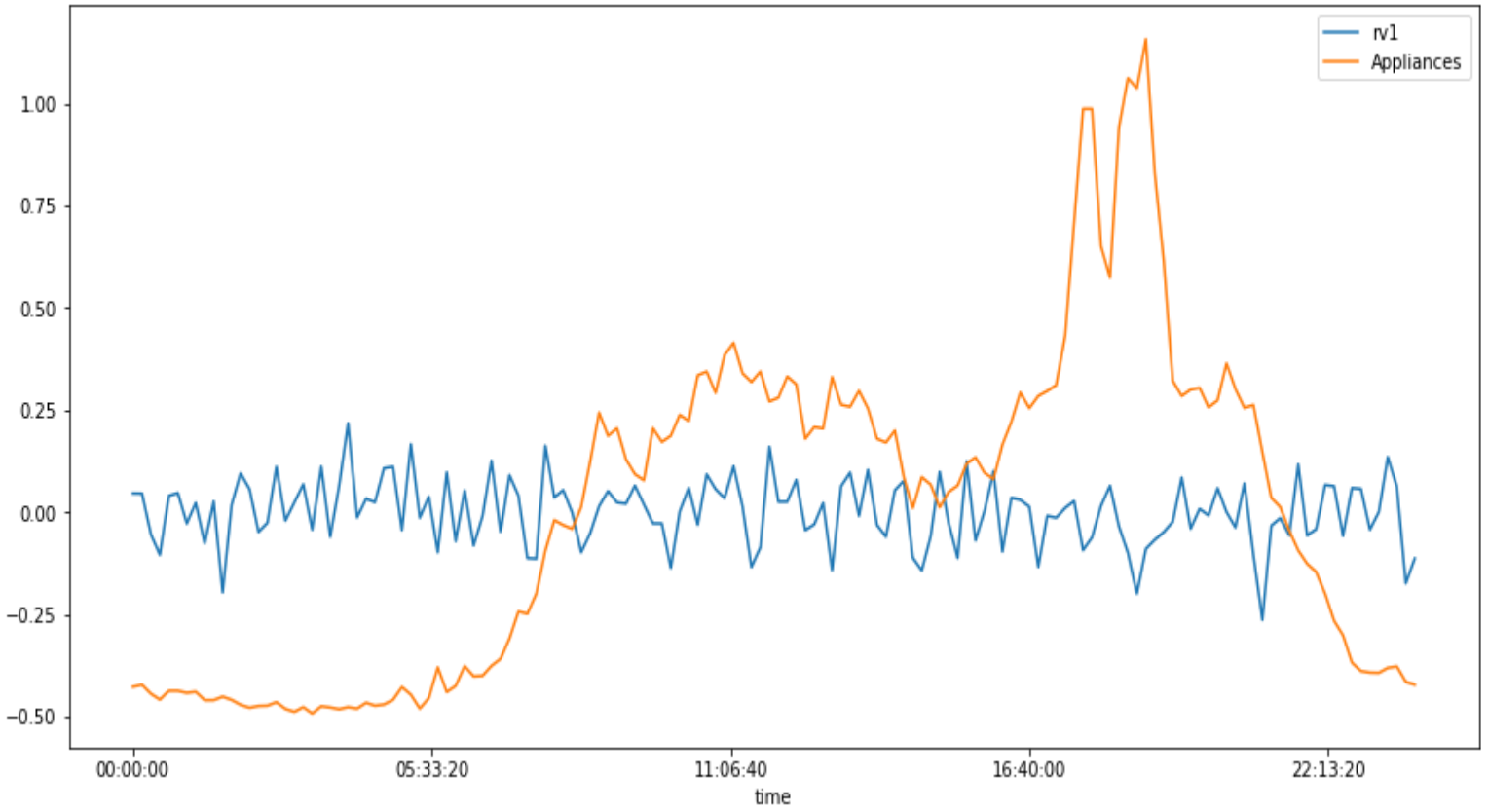
Windspeed, Visibility and Tdewpint Analysis

- Features windspeed and Tdewpoint shows some significant follow-up pattern with the Appliances energy consumption.
- Whereas, visibility does not shows any useful information for the energy prediction.



Random Variable Analysis

- Feature rv1 does not have any specific pattern in the Appliances energy consumption over time.
- It was just added to check the feature selection tools performance.



Part 3: Feature Engineering

Observations from EDA

- Feature date was not with the proper data type. Data type conversion needed for the same.
- Features rv1 and rv2 are perfectly correlated.
- Thus we can remove rv2.
- Feature T9 shows the high correlation with the T3, T4, T5 and T7, introducing redundancy in the dataset and hence can be eliminated.
- T6 and T_out are building exterior features and are highly correlated to each other.
- Thus we can get rid either of these features.

Observations from EDA

- To get more insight on the appliance energy utilization we will introduce some more variables.
- time : time of the record
- month: Month of the record
- Only_Date: Date part of the record
- NSM: Minutes from midnight for the record.
- DOY: Day of year of record.
- Day of Week: Day of week for the record.

Creating Feature selection function for the dataset

```
def featureengineering(energy):
    # Converting datatype of Date column to date time
    energy['date'] = pd.to_datetime(energy['date'])

    # Removing rv2 feature
    del energy['rv2']

    # Removing T9 feature
    del energy['T9']

    # Removing T6 feature
    del energy['T6']

    # Removing rv1
    del energy['rv1']

    # To get the month for that record
    energy['month'] = energy['date'].dt.month

    # To get the specific time for that record
    energy['time'] = energy['date'].dt.time

    #-----------------------------------------------------
    p = []
    q = []
    for i in energy['date']:
        p.append(i.strftime("%j"))
        q.append(i.hour * 60 + i.minute)

    p = list(map(int, p))
    #-----------------------------------------------------
```

```
    # To get the minutes from midnight for that record
    energy['NSM'] = pd.DataFrame({'NSM': q})

    # To get the only date
    energy['Only_Date'] = energy['date'].dt.date
    energy['Only_Date'] = pd.to_datetime(energy['Only_Date'])

    # To get the weekday for that record
    energy['Day of Week'] = energy['date'].dt.weekday

    # Generating training and testing dataset
    from sklearn.model_selection import train_test_split

    energy_train, energy_test = train_test_split(energy, test_size=0.2)

    return energy_train, energy_test
```

Part 4: Prediction Algorithms:

Base Feature:

- After exploratory data analysis and feature engineering, we came up with the base features on which we have to build predictive models.
- Following features are selected as a base line:

```
energy.columns
```

```
Index(['date', 'Appliances', 'lights', 'T1', 'RH_1', 'T2', 'RH_2', 'T3',
       'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8',
       'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed', 'Visibility',
       'Tdewpoint', 'month', 'time', 'DOY', 'NSM', 'Only_Date', 'Day of Week'],
      dtype='object')
```

Training and Testing dataset

- After performing feature engineering on our dataset, we will split it to training and testing datasets which will help us to build effective prediction model.

```
def getXY(df):

    X = df.drop(['Appliances'],axis =1)
    Y = df['Appliances']
    return X,Y
```

```
X_trn, y_trn = getXY(df_trn)
```

```
X_test, y_test =  getXY(df_test)
```

Benchmark Model:

- Let’s select Linear regression model with base features.
- We will use train and test datasets for the model performance.
- Below are the performance metrics for these benchmark model.

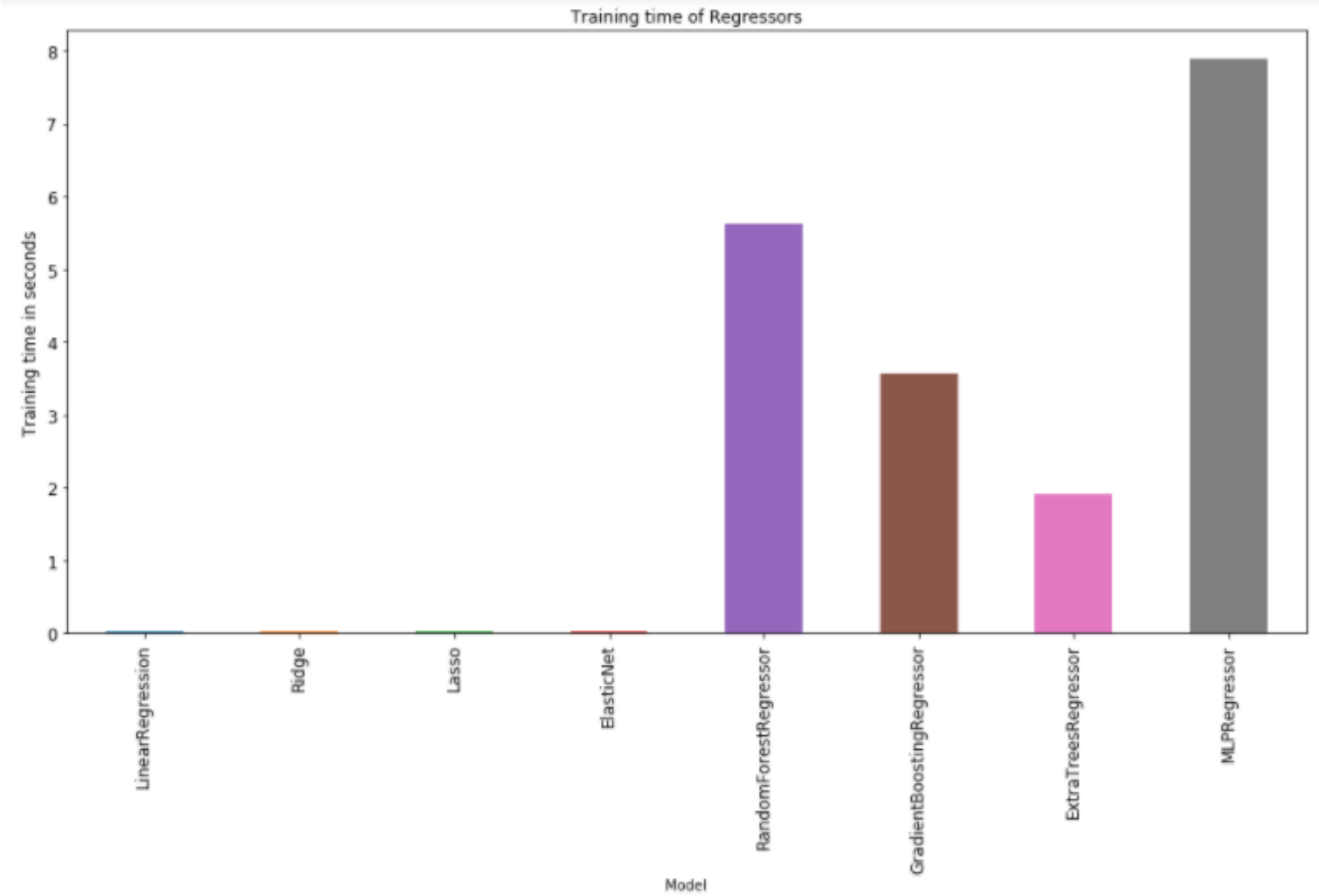
	Train	Test
MAE	**52.337**	**54.558**
MAPE	**60.426**	**62.371**
MSE	**8619.534**	**9472.858**
R2	**0.166**	**0.154**

- Now let’s try to improve these performance with different models and feature selections.

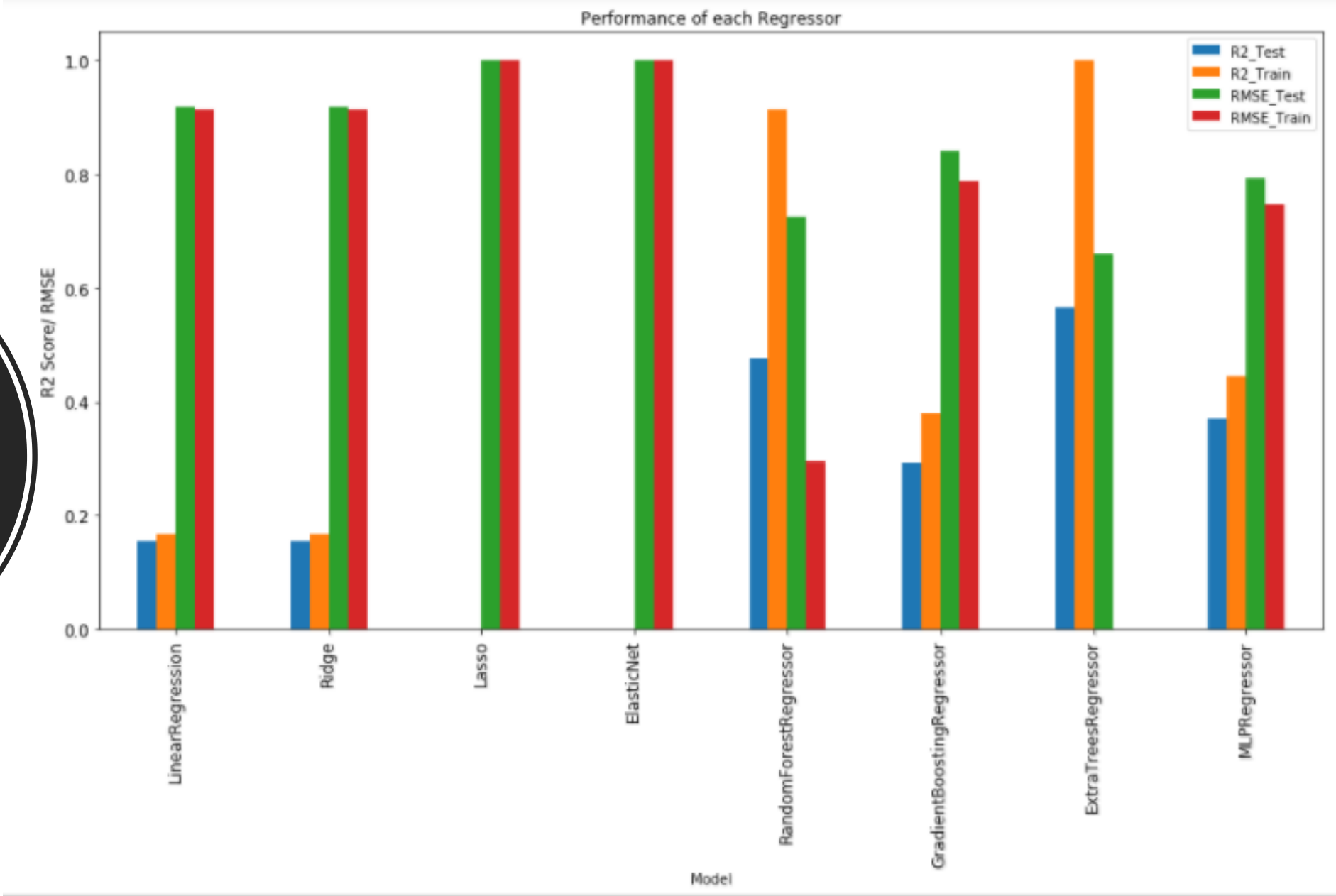
Performance Metrics of different models with base features.

Model	MAE_Test	MAE_Train	MAPE_Test	MAPE_Train	MSE_Test	MSE_Train	R2_Test	R2_Train	RMSE_Test	RMSE_Train	Testing Score(%)	Training Score(%)	Training Time
ExtraTreesRegressor	30.063	0	28.617	0	4269.744	0	0.619	1	65.343	0.011	61.886	100	1.905
RandomForestRegressor	33.515	13.044	32.716	12.613	5079.75	899.584	0.547	0.913	71.272	29.993	54.655	91.297	5.552
GradientBoostingRegressor	47.03	43.204	50.692	47.657	7898.475	6406.759	0.295	0.38	88.873	80.042	29.493	38.021	3.543
MLPRegressor	55.283	53.106	63.524	61.643	9390.062	8583.123	0.162	0.17	96.902	92.645	16.179	16.967	2.17
LinearRegression	54.558	52.337	62.371	60.426	9472.858	8619.534	0.154	0.166	97.329	92.841	15.44	16.615	0.036
Ridge	54.557	52.337	62.371	60.425	9472.847	8619.534	0.154	0.166	97.329	92.841	15.44	16.615	0.042
Lasso	54.522	52.243	62.137	60.25	9510.086	8653.339	0.151	0.163	97.52	93.023	15.107	16.288	0.096
ElasticNet	54.791	52.476	62.283	60.322	9662.773	8817.921	0.137	0.147	98.299	93.904	13.744	14.695	0.042

Training time performance



Performance of Regressors



Part 5: Feature selection

Different Approaches

- Boruta
- Tsfresh
- Forward and Backward selection
- RFE (Regressive features elimination)
- Tpot
- Base Features: (31)

'lights', 'T1', 'RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T4', 'RH_4', 'T5','RH_5', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'RH_9', 'T_out','Press_mm_hg', 'RH_out', 'Windspeed', 'Visibility', 'Tdewpoint','month', 'DOY', 'NSM', 'Day of Week’

Boruta: Feature selection

- Features using Boruta: (13)

'lights','RH_2','T3','RH_3','RH_4','RH_5','RH_7','T8','Press_mm_hg','RH_out','DOY','NSM'

- Boruta only selects those columns whose rank is 1 and recommends to drop other ones.

```
R2_test: 0.541504207278
MAE_test: 30.4276665822
MSE_test: 4142.00253357
RMSE_test: 64.3583913221
R2_train: 0.999999848588
MAE_train: 0.00114010640993
MSE_train: 0.0016468203699
RMSE_train: 0.040581034608 6
```

```
lights : True, rank: 1
T1 : False, rank: 12
RH_1 : False, rank: 9
T2 : False, rank: 7
RH_2 : True, rank: 1
T3 : True, rank: 1
RH_3 : True, rank: 1
T4 : False, rank: 13
RH_4 : True, rank: 1
T5 : False, rank: 4
RH_5 : True, rank: 1
RH_6 : False, rank: 3
T7 : False, rank: 4
RH_7 : True, rank: 1
T8 : True, rank: 1
RH_8 : False, rank: 11
RH_9 : False, rank: 9
T_out : False, rank: 6
Press_mm_hg : True, rank: 1
RH_out : True, rank: 1
Windspeed : False, rank: 9
Visibility : False, rank: 15
Tdewpoint : False, rank: 2
month : False, rank: 16
DOY : True, rank: 1
NSM : True, rank: 1
Day of Week : False, rank: 14
```

Forward and Backward selection

- The forward and backward selection process evaluates the relevant features by calculating P-VALUES.
- In our case it has recommended to add features which are as follows:
- 'NSM', 'lights', 'RH_1', 'RH_7', 'RH_2', 'RH_8', 'Windspeed', 'T3', 'T2', 'month', 'T4', 'Tdewpoint', 'RH_3', 'T8', 'T1', 'T7'

```
Add  NSM                  with p-value 3.23065e-166
Add  lights               with p-value 5.6708e-85
Add  RH_out               with p-value 9.75729e-45
Add  RH_1                 with p-value 1.71079e-41
Add  RH_7                 with p-value 8.39661e-66
Add  RH_2                 with p-value 9.87499e-49
Add  RH_8                 with p-value 6.2177e-13
Drop RH_out               with p-value 0.623901
Add  Windspeed            with p-value 4.38084e-11
Add  T3                   with p-value 2.61048e-09
Add  T2                   with p-value 8.59357e-90
Add  month                with p-value 3.33014e-26
Add  T4                   with p-value 2.2809e-12
Add  Tdewpoint            with p-value 5.3945e-05
Add  RH_3                 with p-value 0.00025239
Add  T8                   with p-value 7.20872e-05
Add  T1                   with p-value 3.6947e-05
Add  T7                   with p-value 0.00683078

resulting features:
['NSM', 'lights', 'RH_1', 'RH_7', 'RH_2', 'RH_8', 'Windspeed',
'T3', 'T2', 'month', 'T4', 'Tdewpoint', 'RH_3', 'T8', 'T1', 'T7']

R2_test: 0.56511887411
MAE_test: 28.9969597162
MSE_test: 3928.67012921
RMSE_test: 62.6791044066
R2_train: 0.999999161408
MAE_train: 0.00152014187991
MSE_train: 0.00912085127945
RMSE_train: 0.0955031480081
```

Tsfresh

- TSFRESH deals with creation of calculated features based on individual entities present in the dataset.
- In the First extraction the features calculated are more than 1200 and, we can further impute the results to get more relevant features, Or it has a extract_relevant_feature method which does all of the above in one step.

```
R2_test: 0.787207136716
MAE_test: 14.9866070614
MSE_test: 392.614890038
RMSE_test: 19.814512107
R2_train: 1.0
MAE_train: 9.02080342791e-15
MSE_train: 1.99314282279e-28
RMSE_train: 1.4117871025e-14
```

RFE:

- RFE (RECURSIVE FEATURE ELIMINATION) is somewhat similar to boruta as it also ranks the features, but the good aspect using this method is we can select number of relevant features we want.
- In this case we have chosen value of 13 so it returns the best 13 features present in the dataset.

```
R2_test: 0.608160981771
MAE_test: 27.7562705853
MSE_test: 3539.83227768
RMSE_test: 59.4964896248
R2_train: 0.999999976706
MAE_train: 0.000253356979985
MSE_train: 0.000253356979985
RMSE_train: 0.0159171913347
```

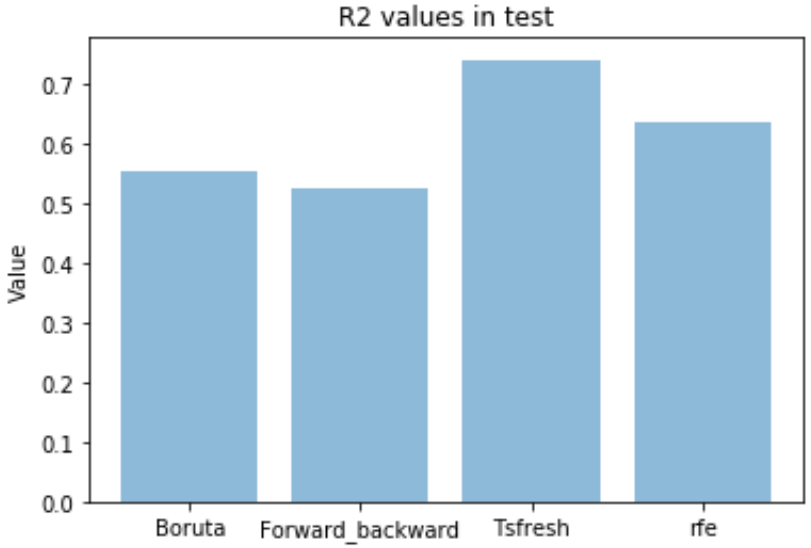
```
lights : False, rank: 10
T1 : False, rank: 9
RH_1 : True, rank: 1
T2 : False, rank: 2
RH_2 : True, rank: 1
T3 : True, rank: 1
RH_3 : True, rank: 1
T4 : False, rank: 6
RH_4 : False, rank: 8
T5 : False, rank: 4
RH_5 : True, rank: 1
RH_6 : True, rank: 1
T7 : False, rank: 7
RH_7 : True, rank: 1
T8 : True, rank: 1
RH_8 : True, rank: 1
RH_9 : False, rank: 3
T_out : False, rank: 5
Press_mm_hg : True, rank: 1
RH_out : True, rank: 1
Windspeed : False, rank: 11
Visibility : False, rank: 13
Tdewpoint : True, rank: 1
month : False, rank: 15
DOY : False, rank: 12
NSM : True, rank: 1
Day of Week : False, rank: 14
```

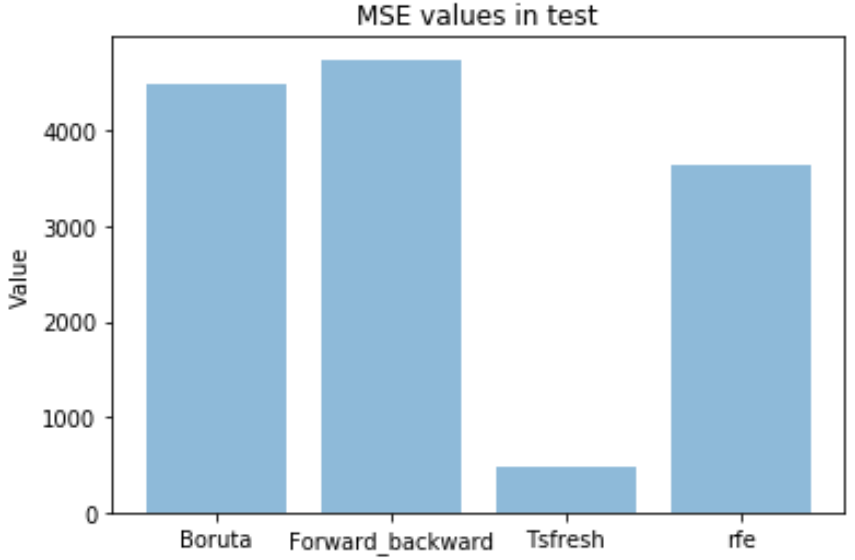
Tpot:

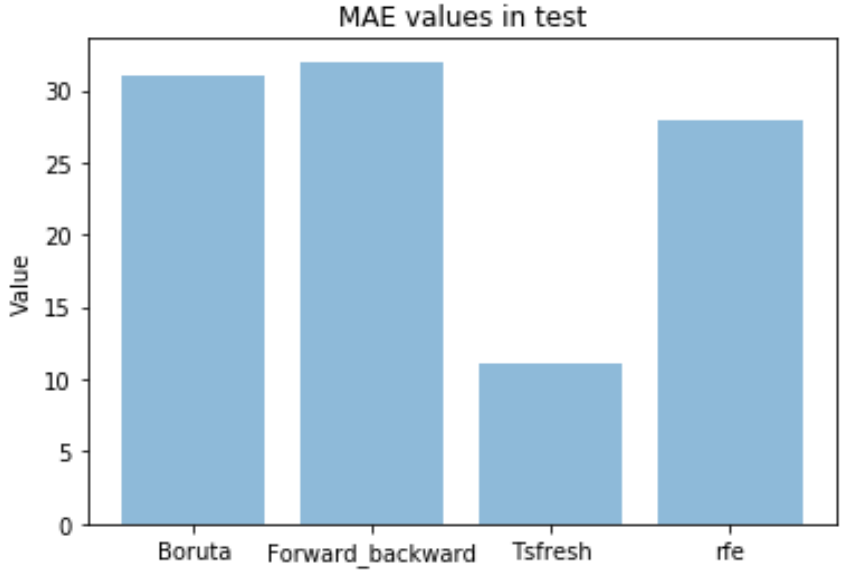
- TPOT is a Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming.
- TPOT will automate the most tedious part of machine learning by intelligently exploring thousands of possible pipelines to find the best one for your data.
- Once TPOT is finished searching (or you get tired of waiting), it provides you with the Python code for the best pipeline it found so you can tinker with the pipeline from there.

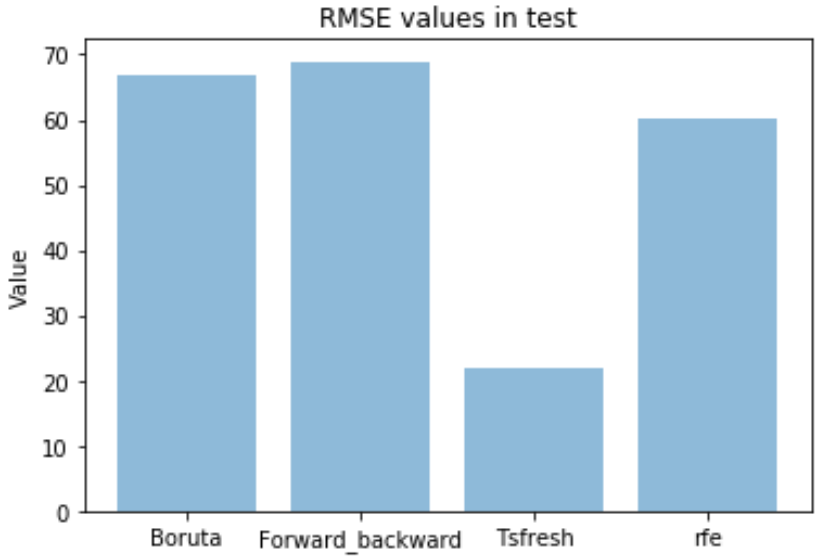
```
Best pipeline: ExtraTreesRegressor(RandomForestRegressor(input_matrix, bootstrap=False, max_features=0.5, min_samples
_leaf=20, min_samples_split=8, n_estimators=100), bootstrap=True, max_features=0.55, min_samples_leaf=5, min_samples_
split=11, n_estimators=100)
-5108.85878747
```

Performance Metrics : Feature Selections tools









Part 6: Model Validation

Model Validation approaches

- Hyper Parameter tuning
  - Randomized Search
  - Grid Search
- Cross Validation
- Bias Variance Tradeoff
- Regularization

Hyperparameter Tuning

- Randomized Search:
- The most important arguments in RandomizedSearchCV are n_iter, which controls the number of different combinations to try, and cv which is the number of folds to use for cross validation (we use 20 and 5 respectively). More iterations will cover a wider search space and more cv folds reduces the chances of overfitting, but raising each will increase the run time. Machine learning is a field of trade-offs, and performance vs time is one of the most fundamental.
- **Grid Search**
- Random search allowed us to narrow down the range for each hyperparameter. Now that we know where to concentrate our search, we can explicitly specify every combination of settings to try. We do this with GridSearchCV, a method that, instead of sampling randomly from a distribution, evaluates all combinations we define. To use Grid Search, we make another grid based on the best values provided by random search:

Model Performance:

- Randomized Search:

```
Model Performance
R2 Test: 0.579
Average Error: 30.4862 degrees.
RMSE Test: 66.9650
```

```
Model Performance
R2 Test: 0.654
Average Error: 27.3151 degrees.
RMSE Test: 60.7378
Wall time: 990 ms
```

```
Improvement of 12.90%.
```

- Grid Search:

```
Model Performance
R2 Test: 0.655
Average Error: 27.2678 degrees.
RMSE Test: 60.6289
```

```
Improvement of 13.12%.
```

Cross Validation

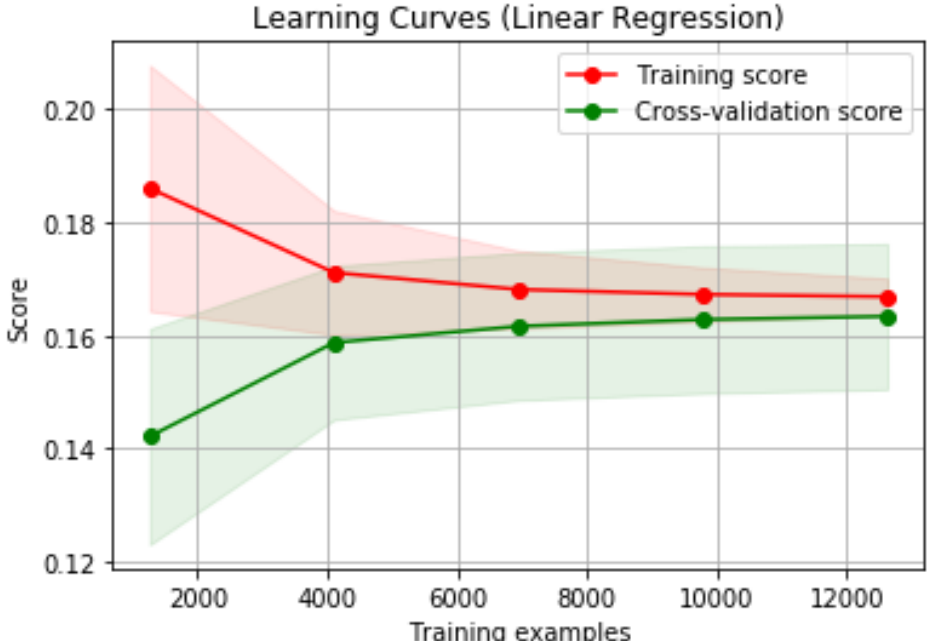
Cross Validation

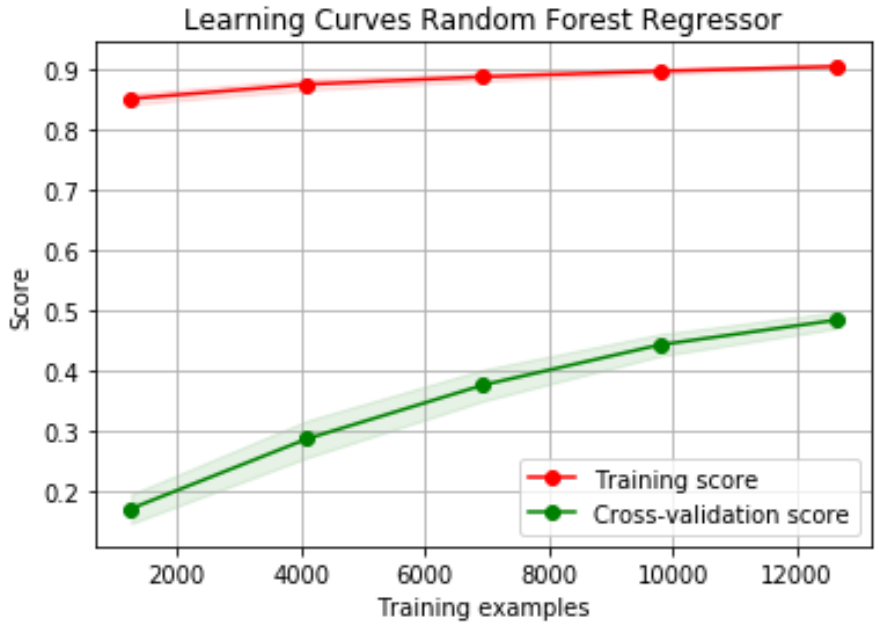
```
from sklearn.cross_validation import cross_val_score

sel_model = LinearRegression()
scores = cross_val_score(sel_model, X_test, y_test, cv=5)
scores.mean()
```

```
0.14452103047948056
```

Bias Variance Tradeoff





Regularization

```
from sklearn.linear_model import Ridge

## training the model

ridgeReg = Ridge(alpha=0.05, normalize=True)

ridgeReg.fit(X_trn,y_trn)

predictions = ridgeReg.predict(X_test)
getResults(X_test,y_test,predictions)
```

```
R2 :0.143
MSE: 9120.449
RMSE: 95.501
MAE: 53.669
MAPE:63.295
```

Ridge

Lasso

```
from sklearn.linear_model import Lasso

## training the model

LassoReg = Lasso(alpha=0.005, normalize=True)

LassoReg.fit(X_trn,y_trn)

predictions = LassoReg.predict(X_test)
getResults(X_test,y_test,predictions)
```

```
R2 :0.143
MSE: 9083.028
RMSE: 95.305
MAE: 53.772
MAPE:63.765
```

```
from sklearn.linear_model import ElasticNet

## training the model

ElasticNetReg = ElasticNet(alpha=0.001, normalize=True)

ElasticNetReg.fit(X_trn,y_trn)

predictions = ElasticNetReg.predict(X_test)
getResults(X_test,y_test,predictions)
```

```
R2 :0.143
MSE: 10313.626
RMSE: 101.556
MAE: 58.989
MAPE:74.507
```

ElasticNet

Part 7: Final Pipeline

Appliance Energy Utilization Prediction Model

- Exploratory Data Analysis
- Feature Engineering
- Different Prediction model
- Feature selection
- Model Validation
- We have to automate the whole process from data injection to final prediction model.
- We will automate these with the customized functions.

Approach

- Create a Jupyter notebook.
- Install all required libraries and packages
- Import data
- Create feature engineering function which will perform the required feature engineering tasks on the imported dataset.
- Create a function to generate the training and testing datasets.
- Create function for the feature selection.
- Create function for model Implementation.
- Create function for model performance evaluation.
- Create function for hyperparameter tuning.
- Final function to execute the whole preccess in a go.

```python
#Import Required Libraries
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from time import time
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestRegressor,ExtraTreesRegressor,GradientBoostingRegressor
from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.neural_network import MLPRegressor
from boruta import BorutaPy
from sklearn.model_selection import RandomizedSearchCV
```

```python
energy_dataset = pd.read_csv('energydata_complete.csv')
```

```python
def featureengineering(energy):
    # Converting datatype of Date column to date time
    energy['date'] = pd.to_datetime(energy['date'])

    # Removing rv2 feature
    del energy['rv2']

    # Removing T9 feature
    del energy['T9']

    # Removing T6 feature
    del energy['T6']

    # Removing rv1
    del energy['rv1']

    # To get the month for that record
    energy['month'] = energy['date'].dt.month

    # To get the specific time for that record
    #energy['time'] = energy['date'].dt.time

    #----------------------------------------------------
```

```python
def getXY(df):

    X = df.drop(['Appliances'],axis =1)
    Y = df['Appliances']
    return X,Y
```

```python
def select_important_features(df_trn):
    X_trn, y_trn = getXY(df_trn)
    # load X and y
    # NOTE BorutaPy accepts numpy arrays only, hence the .values attribute
    X = X_trn.values
    y = y_trn.values

    # define random forest classifier, with utilising all cores and
    # sampling in proportion to y labels
    rf = RandomForestRegressor(n_estimators=20, max_depth=5)

    # define Boruta feature selection method
    feat_selector = BorutaPy(rf, n_estimators=10, verbose=2)

    # find all relevant features
    feat_selector.fit(X, y)
```

```python
def x_sel_columns(X_trn,X_test,sel_columns):
    X_trn = X_trn[sel_columns]
    X_test = X_test[sel_columns]
    return X_trn,X_test
```

```python
def model_Implementation(X_trn,y_trn,X_test,y_test):

    models = [LinearRegression(),
              Ridge(random_state=20),
              Lasso(random_state=20),
              ElasticNet(random_state=20),
              RandomForestRegressor(random_state=20),
              GradientBoostingRegressor(random_state=20),
              ExtraTreesRegressor(random_state=20),
              MLPRegressor(random_state=20)
              ]

    TestModels = pd.DataFrame()
    tmp = {}

    for model in models:
        # get model name
        m = str(model)
        tmp['Model'] = m[:m.index('(')]
        # fit model on training dataset
```

```python
def evaluate(model, test_features, test_labels):
    predictions = model.predict(test_features)
    errors = abs(predictions - test_labels)
    mape = 100 * np.mean(errors / test_labels)
    rmse = np.sqrt(mean_squared_error(test_labels,predictions))
    r2 = model.score(test_features, test_labels)
    print('Model Performance')
    print('R2 Test: {:0.3f}'.format(r2))
    print('Average Error: {:0.4f} degrees.'.format(np.mean(errors)))
    print('RMSE Test: {:0.4f}'.format(rmse))
    return r2
```

```python
def hypertuning(X_trn, y_trn,X_test, y_test):

    # Initialize the model based on best performance from above, We got ExtraTreesRegressor
    sel_model = ExtraTreesRegressor(random_state=42)

    # Define the parameter subset

    param_grid = {
        "n_estimators": [10, 50, 100, 200, 250, 300, 500, 800],
        "max_features": ["auto", "sqrt", "log2"],
        "max_depth": [None, 10, 50, 100, 200, 500]
    }

    # Use Randomized search to try 20 subsets from parameter space with 5-fold cross validation
    random_search = RandomizedSearchCV(sel_model, param_grid, n_iter=20, scoring="r2", cv=5, n_jobs=-1,
    random_search.fit(X_trn, y_trn)
    base_model = ExtraTreesRegressor(random_state = 42)
```

```
def exec_pipeline(energy):
    print('Steps 1. Process started------------')
    df_trn,df_test = featureengineering(energy)
    print('Steps 2.Feature Engineering completed------------')
    X_trn, y_trn = getXY(df_trn)
    X_test, y_test =  getXY(df_test)
    print('Steps 3. Starting Boruta Implementaion to select features------------')
    sel_columns = select_important_features(df_trn)
    print('Selected Features:'+str(sel_columns)+'\n')
    print('Steps 4. Features Selected, Training the models with selected features------------')
    X_trn,X_test = x_sel_columns(X_trn,X_test,sel_columns)
    results = model_Implementation(X_trn,y_trn,X_test,y_test)
    print('Steps 5. Hyper tuning Parameters of ExtraTreesRegressor Model------------')
    hypertuning(X_trn, y_trn,X_test, y_test)
    print('Steps 6. Process Completed------------')
    return results
```

In [13]:
```
results
```

Out[13]:

Model	MAE_Test	MAE_Train	MAPE_Test	MAPE_Train	MSE_Test	MSE_Train	R2_Test	R2_Train	RMSE_Test	RMSE_Train	Testing Score(%)	Tı Sc
LinearRegression	54.903	54.836	64.028	63.371	9371.682	9489.172	0.097	0.100	96.807	97.412	9.697	
Ridge	54.903	54.836	64.028	63.370	9371.682	9489.172	0.097	0.100	96.807	97.412	9.697	
Lasso	54.812	54.720	63.846	63.124	9375.973	9493.499	0.097	0.100	96.830	97.435	9.656	
ElasticNet	54.756	54.625	63.804	62.987	9400.582	9521.047	0.094	0.097	96.957	97.576	9.419	
RandomForestRegressor	33.575	13.587	32.913	13.110	5108.024	959.737	0.508	0.909	71.470	30.980	50.781	
GradientBoostingRegressor	46.388	45.302	50.776	49.285	7517.148	7046.176	0.276	0.332	86.701	83.942	27.567	
ExtraTreesRegressor	31.175	0.001	30.789	0.001	4678.601	0.001	0.549	1.000	68.400	0.034	54.918	1
MLPRegressor	58.267	58.112	72.189	71.293	9009.402	9120.380	0.132	0.135	94.918	95.501	13.188	