Capstone project

Machine learning report

Title: Appliances energy prediction

Author:venkat

Definition

Project overview:

My project is about the aim to the prediction of the energy consumption in a home.in the present world scenario smart homes is rising the energy management because in the now a days reduction of the energy prediction is the play main role .if we predict the energy consumption it is very useful to the future generation. Actually the energy consumption from day to day life is increases and in the same scenario many companies provide some electronic equipments also to predict the energy consumption. and here I predict and reduce this problem of the energy consumption by using the supervised learning.it is the important to reduce the energy consumption so I decided to choose this project.

Dataset link: Reference:https://www.sciencedirect.com/science/article/pii/S0378778816300305

Problem statement:

This is project about the energy consumption using the regression in the supervised learning model, here I predict the appliance energy by the based on the sensor data and weather data as the features. Actually in my problem at 1st read the data sets and after that if there any high coefficient of the correlation between the features then it said that it means there is no linear relation between them and we can simply neglect or drop those features as in this in my problem for the 28 features we simply neglect the light, date features because those are not affect on the problem. After that I check for the null values if there any null values or outliers we can also remove those values. In my project I used the linear regression at first and after the training and accuracy and r_2 scores on the many other methods lasso, ridge, random forest, mlp regressor and some other and I fixed the extra tree regressor of the supervised learning is best to my problem .and I used appliances energy as the target variable and I take features as sensor data and the weather data.

Metrics:

Here in this project I used the regression so the common metrics of the regression is r2_score I.e coefficient of determination and by this metric which will measure the variance of the target variable explained by the our given featues as the input.

And it is mathematically shown as : r2=1-(residual sum of the squares / total sum of the squares)

link: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

link: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html

i used the pearsonr for the finding the r2_score form the scify library.

Here the r2_score defines the measure of the how well the model fits the data. and the root mean squared error gives the absolute measure of better the model fits the data than the r2 and how close our predicted values to the actual values.

Link for the rmse: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html I

used

=>>**np.sqrt(mean_squared_error(y_t, benchmark_model.predict(X_t) for finding the rmse score and from the scilit library.

Finally my metrics are R2_socre and rmse and these two are helpful in our project because my problem is based on the regression and by the r2 score it will explain the statically robustness of the model and Rmse give the idea about the how accurate the predictions are which compared to the actual values .

Analysis:

i.Data exploration:

The dataset has 19,375 instances and 29 attributes including the predictors and the target variables. And I here take all the features to predict more accurately and make correct predictions. The 29 attributes are explained below Give below are the my input features:

1. T1:temperature in kitchen area, in Celsius
2. RH_1: humidity in kitchen area in %
3. T2: Temperature in living room area, in Celsius
4. RH_2: Humidity in living room area, in %
5. T3: Temperature in laundry room area
6. RH_3: Humidity in laundry room area, in %
7. T4: Temperature in office room, in Celsius
8. RH_4: Humidity in office room, in %
9. T5: Temperature in bathroom, in Celsius

10. RH_5: Humidity in bathroom, in %
11. T6: Temperature outside the building, in Celsius
12. RH_6: Humidity outside the building, in %
13. T7: Temperature in ironing room, in Celsius
14. RH_7: Humidity in ironing room, in %
15. T8: Temperature in teenager room 2, in Celsius
16. RH_8: Humidity in teenager room 2, in %
17. T9: Temperature in parents' room, in Celsius
18. RH_9: Humidity in parents' room, in %
19. T_out: Temperature outside, in Celsius
20. Pressure: in mm Hg
21. RH_out: Humidity outside, in %
22. Wind speed: in m/s
23. Visibility: in km
24. T_dewpoint: Â°C
25. rv1: Random variable 1, non-dimensional
26. rv2: Random variable 2, non-dimensional
27. Lights: energy use of light fixtures in the house in Wh and appliances is the

my target variable .

For my project I used the data set which contains the above features and sample of the data from the dataset is here :

	T1	RH_1	T2	RH_2	T3	RH_3	T4	RH_4	T5	RH_5	...	RH_8	T9	RH_9	T_out	Press_mm_hg	RH_out	Wind speed	Visibility	Tdewpoint	Apliances
0	20.20	37.500000	17.823333	39.300000	20.29	36.560000	18.200000	37.290000	17.926667	47.633333	...	35.663333	18.100000	35.500000	2.800000	744.000000	86.666667	2.666667	28.000000	0.766667	70
1	20.00	42.700000	19.100000	42.466667	20.79	44.500000	17.790000	43.790000	17.594444	54.400000	...	53.450000	16.890000	50.000000	4.633333	754.233333	94.333333	4.000000	26.000000	3.800000	40
2	20.60	36.833333	17.500000	40.223333	21.60	34.863333	20.390000	35.363333	19.290000	47.993333	...	38.466667	19.200000	37.633333	3.033333	750.666667	94.666667	3.166667	60.833333	2.200000	70

	T1	RH_1	T2	RH_2	T3	RH_3	T4	RH_4	T5	RH_5	...	RH_8	T9	RH_9	T_out	Press_mm_hg	RH_out	Windspeed	Visibility	Tdewpoint	Apliances
3	22.39	39.090000	19.890000	41.000000	24.89	37.045000	22.290000	35.652857	20.815000	53.617500	...	46.400000	20.500000	40.466667	7.266667	756.850000	88.333333	4.000000	30.000000	5.466667	60
4	20.20	40.526667	18.390000	41.363333	21.00	39.700000	20.823333	39.500000	17.878889	49.988889	...	43.983333	18.133333	42.433333	2.700000	736.300000	95.000000	2.000000	36.000000	2.000000	110

here there is total 28 features I used only 25 and remaining are light, date because we predict the energy consumption not the category wise energy consumption. And in all of these 25 features 24 are input features and 1 target variable. Here I train the data on the 14,801 instances and testing on the 4,934 instances and therefore 19,735 are the total instances. And here are 0 null values as I got in the project.

All above indicated hourly data climate conditions are collected from the chievres weather station it is the airport whether station.

In my project I evaluate the descriptive statistics as given below: **A.Range**

of the columns I get are pated below:

	T1	T2	T3	T4	T5	T6	T7	T8	T9
count	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000
mean	21.691343	20.344518	22.278802	20.860393	19.604773	7.923216	20.273236	22.028122	19.493479
std	1.615790	2.202481	2.012934	2.048076	1.849641	6.117495	2.118416	1.960985	2.022560
min	16.790000	16.100000	17.200000	15.100000	15.340000	-6.065000	15.390000	16.306667	14.890000

	T1	T2	T3	T4	T5	T6	T7	T8	T9
25%	20.760000	18.790000	20.790000	19.533333	18.290000	3.626667	18.700000	20.790000	18.000000
50%	21.600000	20.000000	22.100000	20.666667	19.390000	7.300000	20.075000	22.111111	19.390000
75%	22.633333	21.500000	23.340000	22.100000	20.653889	11.226667	21.600000	23.390000	20.600000
max	26.260000	29.856667	29.236000	26.200000	25.795000	28.290000	26.000000	27.230000	24.500000

Above code is about the temperature columns.

=>

	RH_1	RH_2	RH_3	RH_4	RH_5	RH_6	RH_7	RH_8	RH_9
count	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000
mean	40.267556	40.434363	39.243995	39.043799	51.014065	54.615000	35.410874	42.948244	41.556594
std	3.974692	4.052420	3.245701	4.333479	9.107390	31.160835	5.097243	5.210450	4.161295
min	27.023333	20.596667	28.766667	27.660000	29.815000	1.000000	23.260000	29.600000	29.166667
25%	37.363333	37.900000	36.900000	35.560000	45.433333	29.996667	31.500000	39.096667	38.530000
50%	39.693333	40.500000	38.560000	38.433333	49.096000	55.267500	34.900000	42.390000	40.900000
75%	43.066667	43.273453	41.730000	42.200000	53.773333	83.226667	39.000000	46.500000	44.326667
max	63.360000	54.766667	50.163333	51.090000	96.321667	99.900000	51.327778	58.780000	53.326667

It is about the pressure

=>

	T_out	Tdewpoint	RH_out	Press_mm_hg	Windspeed	Visibility
count	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000	14801.000000
mean	7.421836	3.782509	79.824197	755.480135	4.029001	38.290284

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

It is about the humidity.

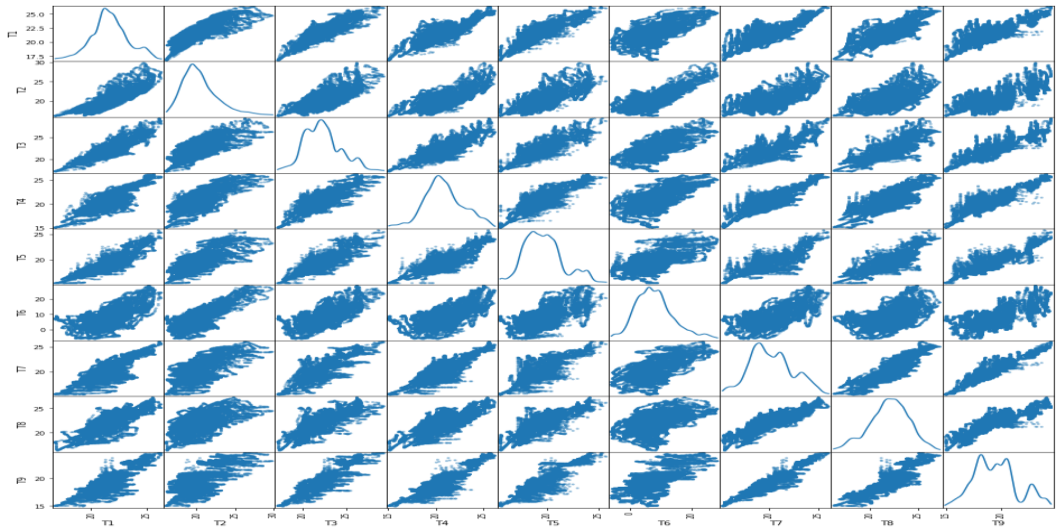
=>

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

It is about the appiances

B.and my scatter plots are:

=>This below plot is about the temperature

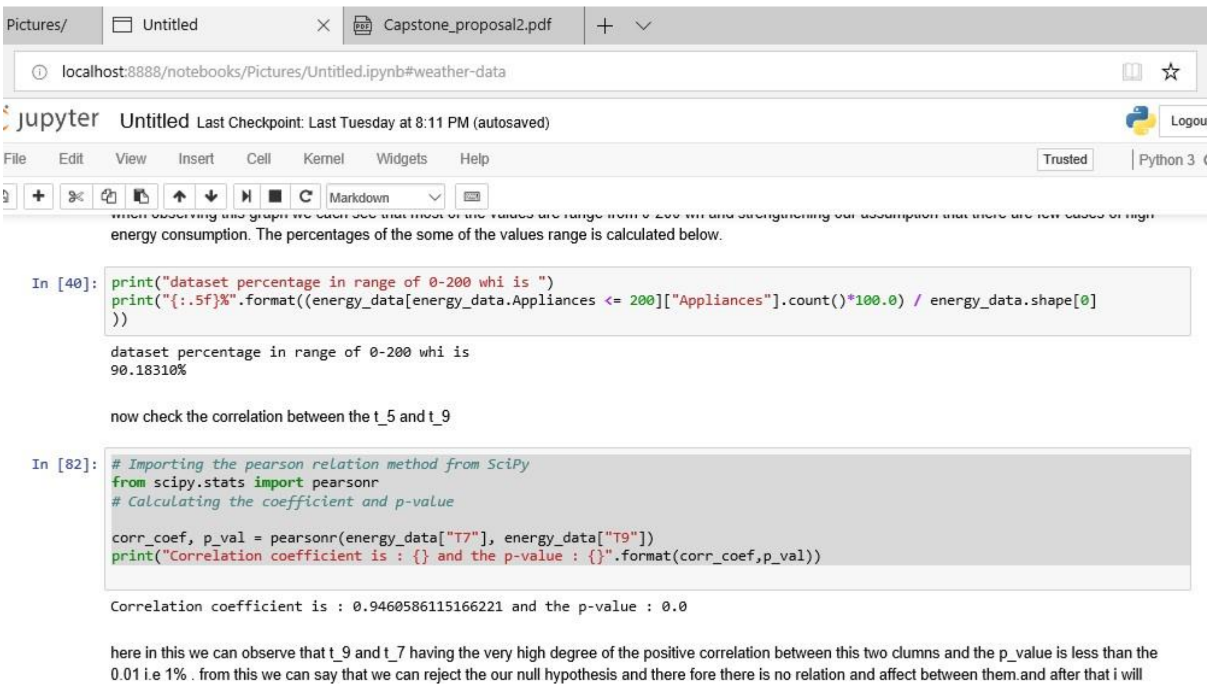


Here by this I evaluated the correlation between the t7 and t9.

by this plots above obtained, We can see that there is a significant correlation between the columns T7 and T9 and this is confirmed the my computing values of there pearson coefficient and it gives the ouput as

```
Correlation coefficient is : 0.9460586115166221 and the p-value : 0.0
```

And my code for this is



energy consumption. The percentages of the some of the values range is calculated below.

In [40]:
```
print("dataset percentage in range of 0-200 whi is ")
print("{:.5f}%".format((energy_data[energy_data.Appliances <= 200]["Appliances"].count()*100.0) / energy_data.shape[0]
))
```

```
dataset percentage in range of 0-200 whi is
90.18310%
```

now check the correlation between the t_5 and t_9

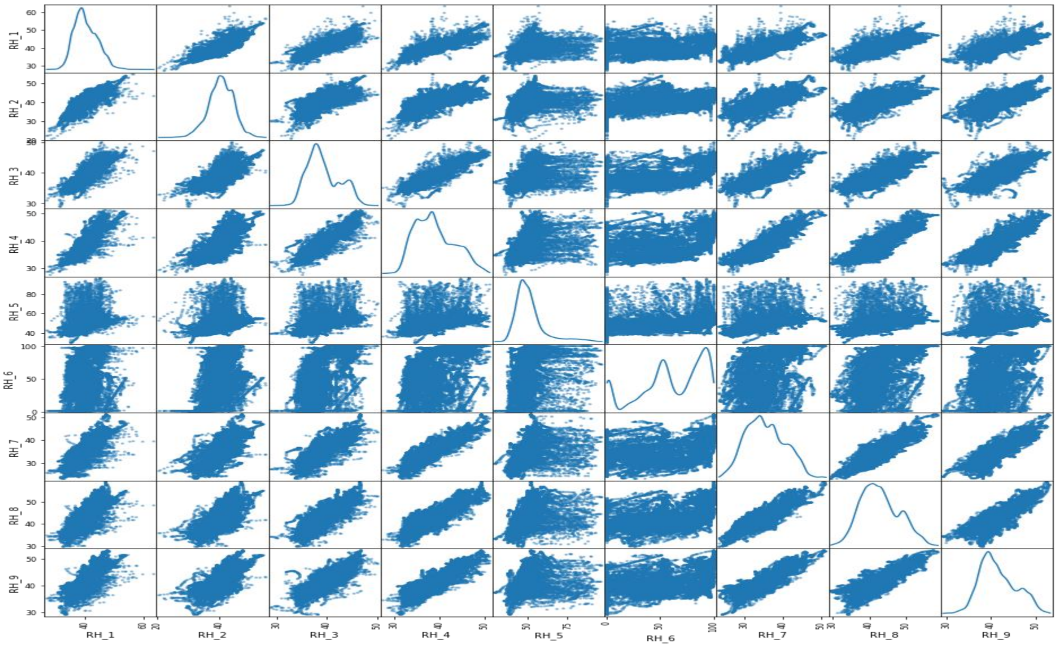
In [82]:
```
# Importing the pearson relation method from SciPy
from scipy.stats import pearsonr
# Calculating the coefficient and p-value

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

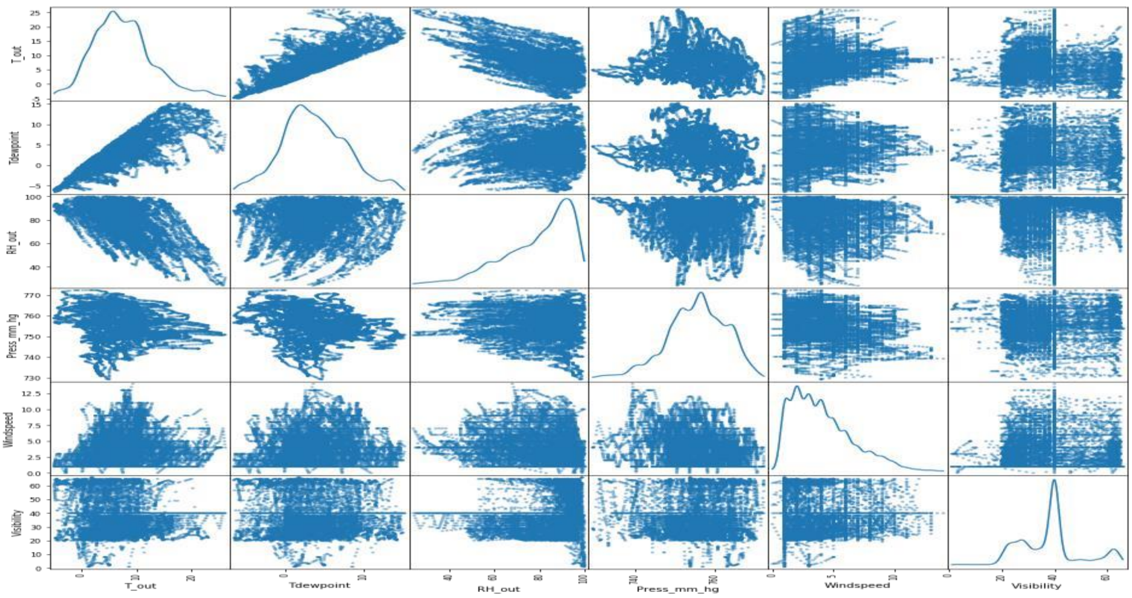
```
Correlation coefficient is : 0.9460586115166221 and the p-value : 0.0
```

here in this we can observe that t_9 and t_7 having the very high degree of the positive correlation between this two clumns and the p_value is less than the 0.01 i.e 1% . from this we can say that we can reject the our null hypothesis and there fore there is no relation and affect between them.and after that i will

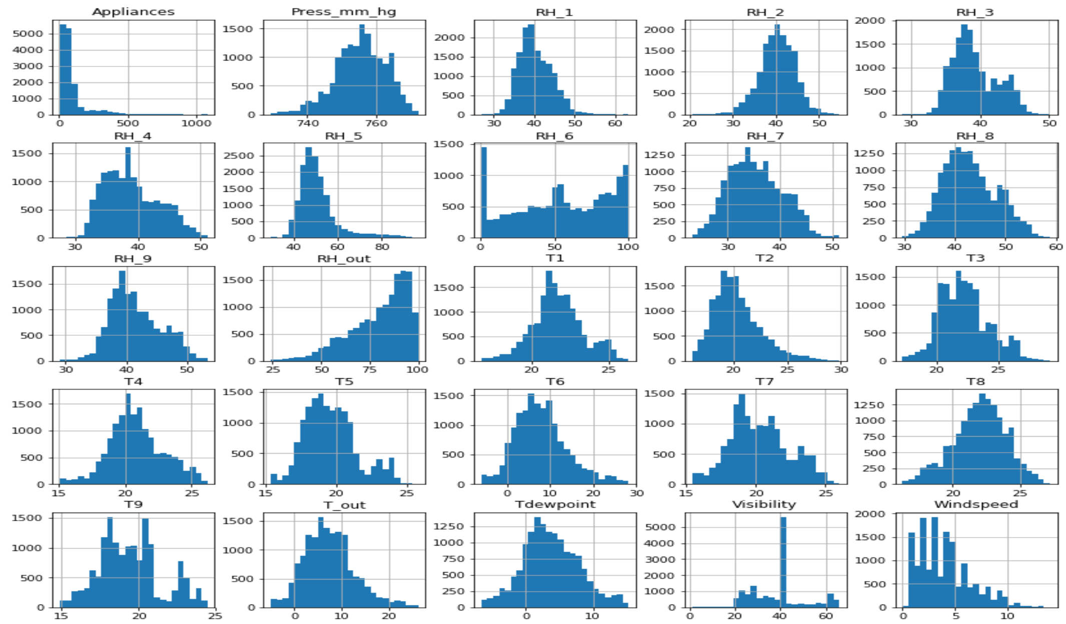
=>This below plot is about the humidit



=> This below plot is about the weather data



c.Districution of all the columns:



by the histograms we observe that:-

=>here in this all the humidity values except rh_out and rh_6 shows the normal distribution.by that all that readings from the sensors are inside the home are form a normal distribution.and remaining are skew ones for the humidity columns. =>in the same way except th t_9 all the temperature readings follow the normal distribution.

=>and for the remaining columns like visibility,appliances and windspeed are follows the skewed distribution.

=>Also, there is no similarity between our target variable, Appliances and the remaining 24 columns.windspeed looks similar but the number of observations are different as seen from the y-axes of both plots.

Let's confirm this by plotting Appliances against the Windspeed. Also, let's plot Appliances histogram separately to get better idea about it's distribution.

```
In [101]: # Importing the pearson relation method from SciPy
          from scipy.stats import pearsonr
          # Calculating the coefficient and p-value

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

          Correlation coefficient is : 0.9460586115166221 and the p-value : 0.0
```

here in this we can observe that t_9 and t_7 having the very high degree of the positive correlation between this two clumns and the p_value is less than the 0.01 i.e 1% . from this we can say that we can reject the our null hypothesis and there fore there is no relation and affect between them.and after that i will check the high degree of correlations greater than the 0.9 i.e >0.9

```
In [102]: # Calculating the coefficient and p-value
          corr_coef, p_val = pearsonr(energy_data["T3"], energy_data["T9"])
          print("Correlation coefficient is : {} and the p-value : {}".format(corr_coef,p_val))

          Correlation coefficient is : 0.9009710955349393 and the p-value : 0.0

In [103]: # Calculating the coefficient and p-value
          corr_coef, p_val = pearsonr(energy_data["T7"], energy_data["T9"])
          print("Correlation coefficient is : {} and the p-value : {}".format(corr_coef,p_val))

          Correlation coefficient is : 0.9460586115166221 and the p-value : 0.0

In [104]: # Calculating the coefficient and p-value
          corr_coef, p_val = pearsonr(energy_data["T6"], energy_data["T9"])
          print("Correlation coefficient is : {} and the p-value : {}".format(corr_coef,p_val))

          Correlation coefficient is : 0.6668305882664114 and the p-value : 0.0
```

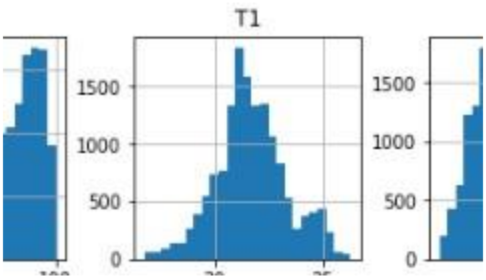
here when we observe thi value we can say that t6 and t9 having the less correlation coefficient

=>by this way we have make lot of pairs so i write an algorithm to find the correlation coefficient >0.9

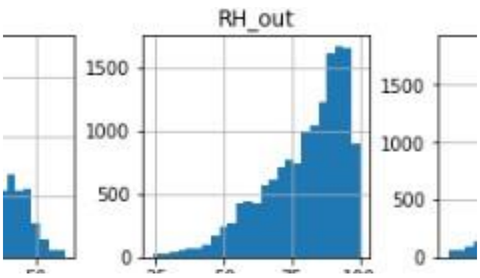
There in the my data set when finding the correlation coefficient of the t7 and t9 having the high degree of the positive correlation between this two columns and the p_values is less than the 0.01 and from this we can say that we can reject the our null hypothesis and there fore there is no linear relation and affect between them. if there is the features having the high degree of the correlation then we can say that those two are doesn't having the linear relation between them so I introduced the scaling in this to make some relation between those.

2.Exploratory visualization:

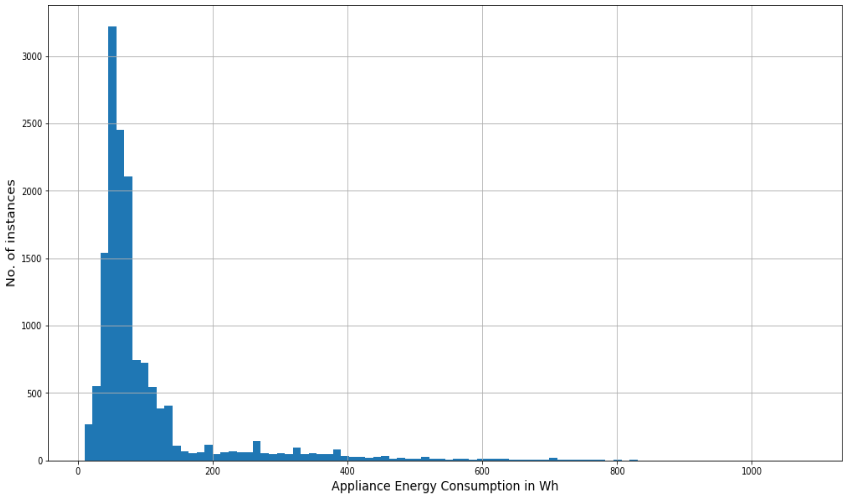
In this data maximum of the features having the normal distribution



And some features having the right or left skew distribution shown as shown below:



=>And the target variable distribution :



when observing this graph we can see that most of the values are range from 0-200 wh and strengthening our assumption that there are few cases of high energy consumption

observations:

here in this most of this features are normally distributed and to the target variable has the high skewed distribution and it is not maintain a linear relation with the any of the other features. And here the t9 is highly correlated with the t3,t5, and t7 and the t6 with t_out.

```
corr_coef, p_val = pearsonr(energy_data["T6"], energy_data["T9"])
print("Correlation coefficient is : {} and the p-value : {}".format(corr_coef,p_val))
```

```
Correlation coefficient is : 0.6668305882664114 and the p-value : 0.0
```

here when we observe thi value we can say that t6 and t9 having the less correlation coefficient

=>by this way we have make lot of pairs so i write an algorithm to find the correlation coefficient >0.9

In [48]:

```
# all pairs for given columns having cc>0.9
from itertools import combinations
for i in combinations(energy_data.columns, 2):
    clm1, clm2 = i
    corr_coef, p_val = pearsonr(energy_data[clm1], energy_data[clm2])
    # Check for high correlation
    if corr_coef > 0.9 or corr_coef < -0.9:
        print("Column pairs which are having cc>0.9: {}, {}".format(*i))
        print("Correlation coefficient : {}".format(corr_coef))
        print("p-value : {}".format(p_val))
```

```
Column pairs which are having cc>0.9: T3, T9
Correlation coefficient : 0.9009710955349393
p-value : 0.0
Column pairs which are having cc>0.9: T5, T9
Correlation coefficient : 0.9101631787384007
p-value : 0.0
Column pairs which are having cc>0.9: T6, T_out
Correlation coefficient : 0.9747835663815296
p-value : 0.0
Column pairs which are having cc>0.9: T7, T9
Correlation coefficient : 0.9460586115166221
p-value : 0.0
```

3.Algorithms and techniques:

For this problem I used the regression algorithm and I take the most basic algorithm in the regression that is the linear regression algorithm of the supervised learning model. If the basic algorithm explains data set well then no need to apply other algorithms. As modifications to original least squares regression ,we can apply the regularization technique on the data to evaluate the coefficient values of the features because higher values generally towards the overfitting and there may be a loss in the generalization. We used the regularization technique because it improves the performance of the linear regression. There are some chances to fit the data by the linear model without the regularization.There some of the linear models are linear,lasso,ridge which are used in the problem .

some of the tree based models are random forest, gradient boosting, extreme randomized trees. After the linear regression I took the tree base regression models and in this model the main advantage is these are more robust to outliers.in the linear regression we can not see the linear relation between the features and the target variable but by the tree based algorithm it provides the better relation between them. After this model i skip the decision trees because when there is substantial number of the features it is evident that the decision tree will leads to overfit the data, so I took the ensemble models, which include building multiple regressors on copies of same training data and combining their output either through mean, median, mode (Bagging) or growing trees sequentially. And in this ensemble, random forest is one of the method and it makes the better performance on the high dimensional data like mine. And we also go with extreme trees regression because It will leads to further by making the split random gradient boosting machine(boosting mrthod)

And finally the one more algorithm in the supervised algorithm is neural networks and it is the non linear hypothesis and make best work when there is no relation between the inputs and the outputs. We cant prefer for the small sized data sets because it takes more time to train the data I used the mlpregression (multi layer regressor) for the choice of the neural networks. Here the error function is squared.

Here is the description of the some methods which are used by me are:

Ridge regressor:

class *sklearn.linear_model.**Ridge**(*alpha=1.0*,* fit_intercept=True*,* normalize=False*,* copy_X=True*,* max_iter=None*,* tol=0.001*,* solver='auto'*,* random_state=None*)*

This model solves a regression model where the loss function is the linear least squares function and regularization is given by the l2-norm. Also known as Ridge Regression or Tikhonov regularization. This estimator has built-in support for multi-variate regression (i.e., when y is a 2d-array of shape [n_samples, n_targets]).

Link: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

Lasso regressor:

The algorithm used to fit the model is coordinate descent.

To avoid unnecessary memory duplication the X argument of the fit method should be directly passed as a Fortran-contiguous numpy array.

Link: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html

Random forest:

The default values for the parameters controlling the size of the trees (e.g. max_depth, min_samples_leaf, etc.) lead to fully grown and unpruned trees which can potentially be very large on some data sets. To reduce memory consumption, the complexity and size of the trees should be controlled by setting those parameter values.

The features are always randomly permuted at each split. Therefore, the best found split may vary, even with the same training data, max_features=n_features and bootstrap=False, if the improvement of the criterion is identical for several splits enumerated during the search of the best split. To obtain a deterministic behaviour during fitting, random_state has to be fixed

Link: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Extra tree regressor:

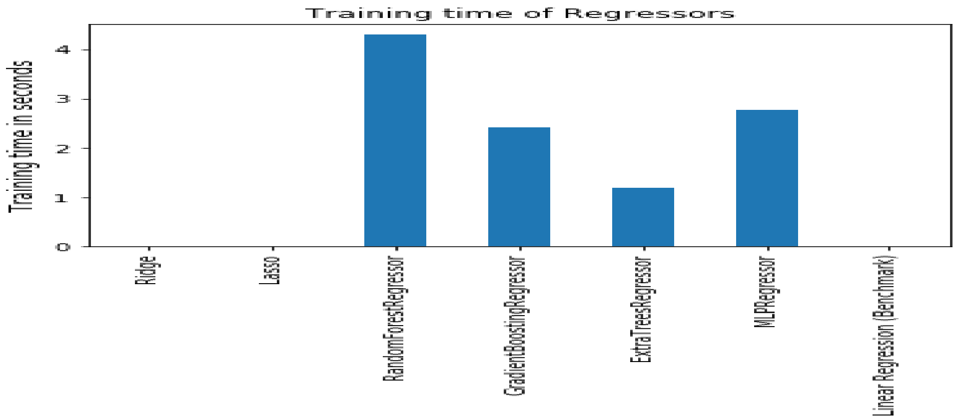
This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Link: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html

And I trained my data by taking the some features as inputs to the problem and after some operations like fitting and tuning data to our model finally it will give the output. And in the

testing also we take some inputs and it will give the outputs as we train the data, as we considered the some set of the data as training and testing set.

Every time for the different models it will train the data by taking the some training time and when comparing our bench mark model linear regression with others in the training time it takes less time but it is not giving the high score than the others which are taking the train time more than this. In the linear regression as I declared if there is no linear relation between the features it cannot improve the output and then come to the ridge it will handle it by the regularization of i2_norm and it wil train the data more efficient than the linear and with the lasso it uses the algorithm used to fit the models is coordinate descent and it is useful to unnecessary memory duplication the X arguments and directly passed to the anumpy arrays. . After this model i skip the decision trees because when there is substantial number of the features it is evident that the decision tree will leads to overfit the data, so I took the ensemble models, which include building multiple regressors on copies of same training data and combining their output either through mean, median, mode (Bagging) or growing trees sequentially. And in this ensemble, random forest is one of the method and it makes the better performance on the high dimensional data like mine and finally at the extra tree regressor give the best score than the others with the all aspects and we can evident this by the below plot.



by this graph we can say that ridge, lasso and linear are the regressor models to train the data fast and random and mlpregressors are slow regressors to train.

4.Bench mark:

As I declared in the proposal I choose my bench mark model as the linear regression and on the all features without scaling the data. And in this model I make some obseravations are: r2_score on the training data, testing data sets is 14.68%,14.25 respectively and rmse on the test data is 0.92 and time taken to the fitting is 0.032 sec.

Methodologies :

Note: as u reviewed that a small error in my code I change it I learn a new thing that we use only the fit.transform() on the train not on the test data set.

1.Data preprocessing:

Here are the features ranges

Temperature ->-6 to 30

Humidity->1 to 100

Windspeed ->0 to 14

Visibility=> 1 to 66

Pressure=>729 to 772

Appliances energy usage->10 to 1080

As we observe there are different ranges of the features it is leads to thr domination in the regession and to avoid the situation all the features to be scaled as in the bench mark model we train the data with out the scaling. The scaling is based on the 0 mean and unit variance.

Link to the scale: http://scikit-learn.org/stable/modules/preprocessing.html

And after the scaling I removed the some of the clumns t6 and t9 and these are having the significant correlation along with the t_out and t3,t5,t7 respectively. And finally there are 22 features are in the training set.

2.Implementation:

Here the model implementation has done in the three stages they are:

a. Record the each metrics and execute the each regressor by using the pipline() function .

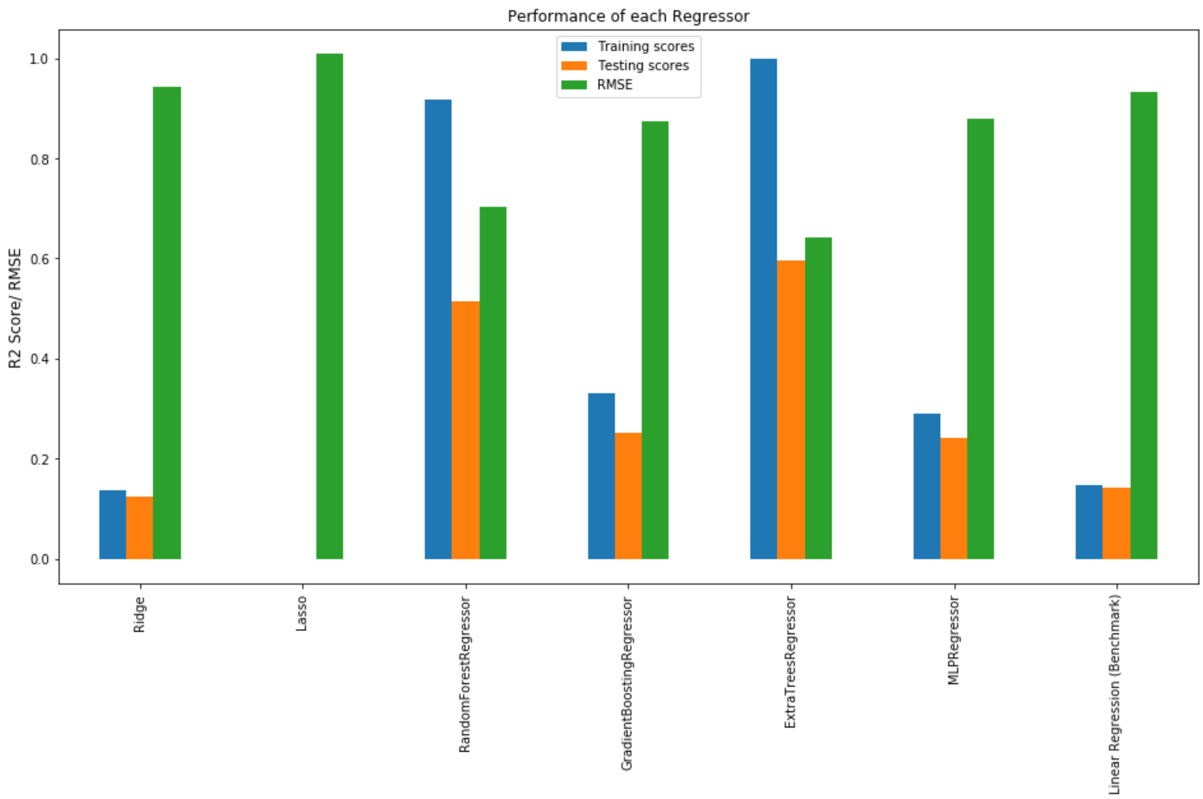
Link: http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html#building-apipeline

b. And anfter that I pass the each regressor to above pipeline() from the execute_pipeline().
c. And finally take the data into the dataframes using the get_prorperties and plot the metrics .
For the choosing the best model I tested below algorithms when compared to the linear regressor they are
Ridge regressor, lasso, random forest, gradient boosting, extra tree regressor,mlpregrssor. And here are the links to them Here
=>while doing the project I faced some of the difficulties are

When comparing the training time of the models, linear regression is the best and in the efficiency extra tree regressor make a mark to it. And why it take that much time too train our data and when dropping of the some features I expected that ,is It make some affect on the score of the model or not . and I didn't expect that these too are make less r2 score than the other features (it is just my opinion) and I faced main problem here is choosing the best model to our data and in the below plot I observed that intuition that lasso is the least performing regressor and the extra tree regressor give the best perform on the our data.as extra tree regressor having the r2_score of 1.0 on the training dataset which leads to overfitting but in this it has the highest score on the testing of the data set. and its rmse value is also less. then i conclude that extra tree regressor is the best model out of the all them. Finally it is little bit difficult ranking the best features in the correct way but here are the some features I ranked

Most important feature = RH_1
Least important feature = Visibility
Top 5 most important features:- RH_1,RH_out,RH_8,T3,RH_3

Top 5 least important features:- Visibility,T4,T1,Tdewpoint,Windspeed



http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html
http://scikit-learn.org/stable/modules/neural_networks_supervised.html

and on all these methods I used the r2 score and rmse as I described above. Results :

	RMSE	Testing scores	Training scores	Training times
Ridge	0.936121	0.123677	0.137409	0.015626
Lasso	1	0	0	0.0156186
RandomForestRegressor	0.72144	0.479525	0.916954	4.30872
GradientBoostingRegressor	0.86829	0.246073	0.331539	2.43047
ExtraTreesRegressor	0.661027	0.563044	1	1.20285

MLPRegressor	0.870117	0.242897	0.290518	2.76574
Linear Regression (Benchmark)	0.926026	0.142476	0.146873	0.0156221

By observing the above data I decided that extra tree regressor is better than others and we can consider this as better method only without considering the training times. when comparing to the times of training in the linear training time is less but it doesn't perform well than the extra tree regressor.

3.Refinement:

For this model I.e extra tree regressor

I use some of the functions which consisted by the extra tree regressor for my convinience

Link: http://scikit-learn.org/stable/modules/generated/sklearn.tree.ExtraTreeRegressor.html#sklearn.tree.ExtraTreeRegressor They are , max_features :it will say about the number of features to be considered max_depth: to find the depth of the tree n_estimators: the number of the trees used

and after that tune the model with our data on the test set r2 score is 0.635 and before tuning it is 0.56

=>R2 score improvement from Benchmark model = 0.473.

=>RMSE improvement from Benchmark model = 0.312.

=>R2 score improvement from Untuned model = 0.058.

=>RMSE improvement from Untuned model = 0.041

Result:

1.Model evalution and validation :

Features of the untuned model

n_estimators=1

max_features=22

max_depth=none

features of the best model after the hyper parameters

n_estimators=200

max_features：log2(n_features)

max_depth=200

Robustness of the checking:

Here from all of the 22 features we have to predict the best features and give the ranking to to them among their importance by this model by the rmse and r2 scores.

And after that I checked for the best features among the 22

R2 score on test data = 0.491

R2 score of untuned model = 0.528

Difference = 0.059or 5.9%.

RMSE on test data = 0.719

RMSE of untuned model = 0.665

Difference = 0.343

2.Justification:

Benchmarks values

Classifier fitted with in the 1.65641 seconds

Score on training data : 0.146873

Score on testing data :0.1425787

Rmse on test data:0.92

And for the best model:

Score on training data : 1.0

Score on testing data : 0.635

Rmse on test data:0.609

Difference in Score on training data :0.8532

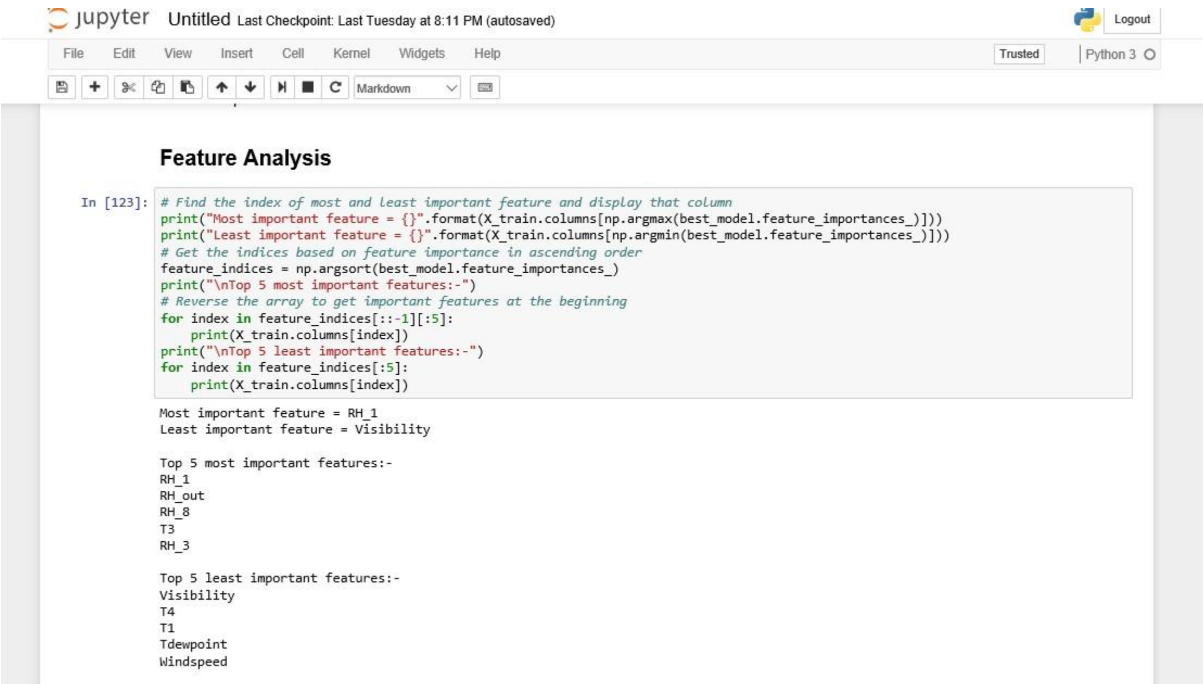
Difference in Score on testing data :0.4925

Difference in Score on training data :0.3

Conclusions:

1.Free from visualization:

As I declared the best ranked important features are

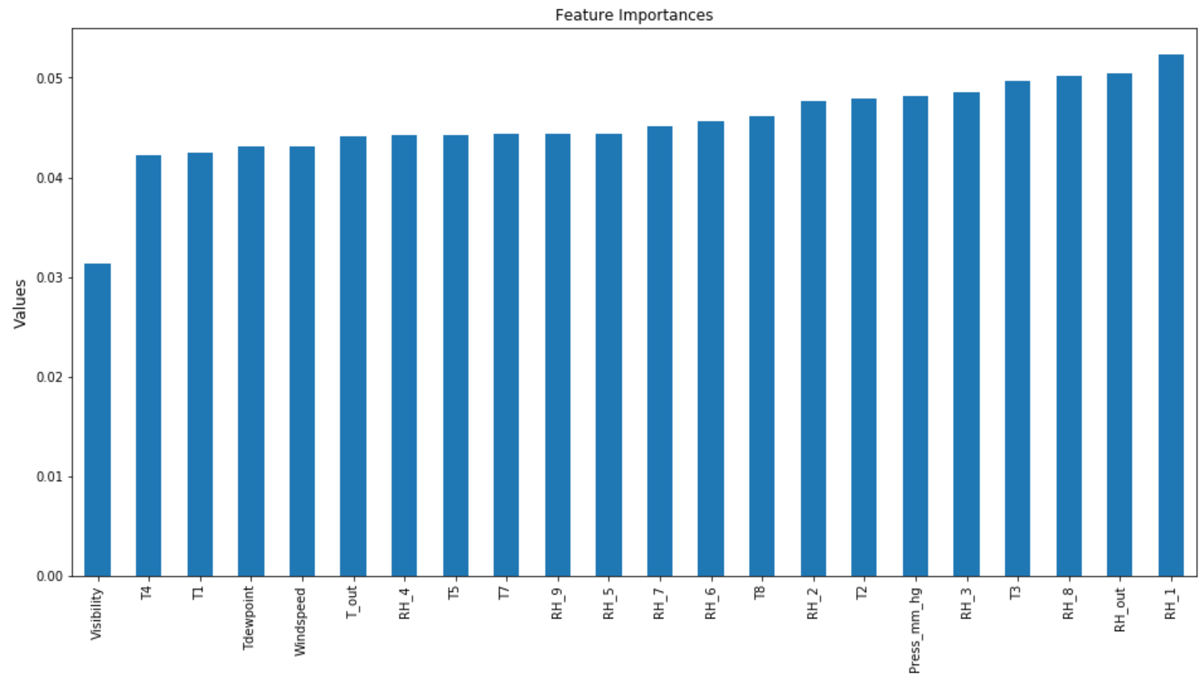



```
# Find the index of most and least important feature and display that column
print("Most important feature = {}".format(X_train.columns[np.argmax(best_model.feature_importances_)]))
print("Least important feature = {}".format(X_train.columns[np.argmin(best_model.feature_importances_)]))
# Get the indices based on feature importance in ascending order
feature_indices = np.argsort(best_model.feature_importances_)
print("\nTop 5 most important features:-")
# Reverse the array to get important features at the beginning
for index in feature_indices[::-1][:5]:
    print(X_train.columns[index])
print("\nTop 5 least important features:-")
for index in feature_indices[:5]:
    print(X_train.columns[index])
```

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

Top 5 most important features:-
RH_1
RH_out
RH_8
T3
RH_3

Top 5 least important features:-
Visibility
T4
T1
Tdewpoint
Windspeed
```



By this we can say the best features and I came to an intuition that in this data humidity effects energy power conjumption more than the temperature and I make this observation because more of the features of humid is towards the higher end than the temperature and out of the weather parameters humid and pressure are affect power consumption more than other.

2.Reflections:

My project is summarized as the following steps;

1. At 1st I search for the data set for my project in the Kaggle and in the UCI machine learning repositories
2. And then I decide that is my problem is belongs to the classification or the regression.
3. I visualized the different aspects of the dataset.
4. After done with the data preprocessing and the selection of the features
5. We have choose the algorithms to be used for the project
6. Choosing the benchmark model to solve the problem
7. Noe apply the selected algorithm and then visualize the result
8. Tuning of the hyper parameters to train the model in the getting of better score
9. And finally decide the best features to this problem and then check the robustness of the model.

Among the above steps 1,3,6 are the little bit interesting steps because they can decide the problem is under the regressor or the classification , and In this I decide the regressor model. After that we have to visualize the data and in the preprocessing if there are any null values or outliers we will drop those ones. Here we train the data based on the some certain inputs and give the output by the testing it will evaluated and find the scores for the models which is the best method in that model. In this project, I had initially decided to create two benchmark models, one that would always return the mean of the target variable and one which would return the median. But, after visualizing the data and concluding that there are no Linear relationships of any feature with the target variable, and after that I realized that linear regression is better to choose as the benchmark model.

3.mprovements:

By all the above things I think that there some improvements are

- ⇨ In this we have to drop the in wanted and irrelevant data features like windspeed and visibility
- ⇨ By using the grid search instead of the randomized one to determine the best solution to search the parameters.
- ⇨ And performing the more aggressive feature engineering.