Appliances Energy Prediction Dataset

This assignment is about implementation of linear and logistic regression on Appliances Energy Prediction dataset with some visualizations of the respective regression models. In this report, for linear regression we implemented the gradient descent algorithm to minimize the error and predict the target variable. The link to the Appliances Energy Prediction dataset is available for download here.

Our main goal for this assignment is to implement a linear regression model on the dataset to predict the energy usage of appliances. Implemented a gradient descent algorithm with batch update to reduce the cost function as much as possible with the help of the learning rate (α). Used the sum of squared error normalized as the cost function and error measure. The number of features used are 15 and some additional experimentation and exploratory analysis has been done. This is a Multiple Linear Regression problem since it has multiple features.

Model Representation

In Simple Regression Model, we have one input variable (X) and one output variable (Y). Similarly, in Multiple Linear Regression Model the input variable (X) has n features. Therefore, we can represent this linear model in general form as:

$$Y = \beta 0x0 + \beta 1x1 + \beta 2x2 + \cdots + \beta nxn$$

Where xn is the ith feature in input variable. By introducing $x0 = 1$, we can write this equation.

$$Y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots + \beta nxn$$

And converting it to matrix form has its own benefits in Python.

$$Y = \beta TX$$

We must define the cost of the model. The cost function basically gives the error in our model. Y in above equation is our hypothesis(approximation) and we are going to define it as our hypothesis function as :

$$h\beta(x) = \beta TX$$

And the cost is,

$$J(\beta) = 1/2m \sum(h_{\beta}(x(i)) - y(i))$$

By minimizing this cost function, we can find β. We use Gradient Descent for this.

Gradient Descent

Gradient Descent is an optimization algorithm. We will optimize our cost function using Gradient Descent Algorithm becomes small enough (i.e. convergence takes place).

Step 1

Initialize values $\beta 0, \beta 1, \ldots, \beta n$ with some value. I have initialized them with 0 initially.

Step 2

Iteratively update until it converges,

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta)$$

Here α is the learning rate and $\partial\ \partial \beta j$

$J(\beta)$ means we are finding partial derivate of cost w.r.t each βj. In step 2 we are changing the values of βj in a direction in which it reduces our cost function. And Gradient gives the direction in which we want to move. Finally, we will reach the minima of our cost function. But we don't want to change values of βj drastically, because we might miss the minima. That's why we need learning rate.

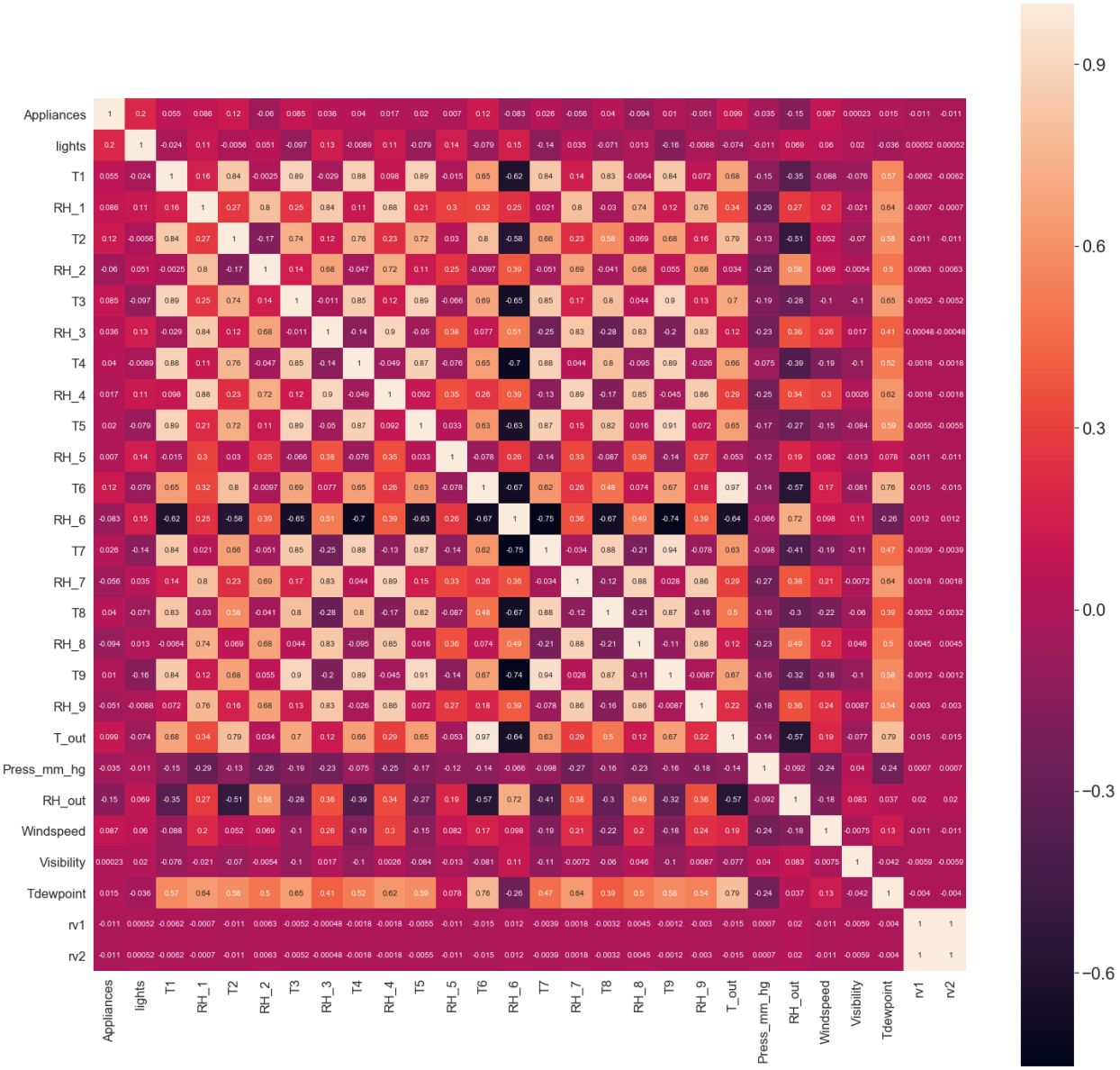
Implementation

I have used python from a notebook called Jupyter notebook to write the code. Though it looks lengthy, I have made use of the arrays and vectors to make the process easy and simple.

Exploratory Analysis – EDA

The main object I've of performing EDA is to get a gist of your data before-hand. For this I have plotted heatmaps and pair plots to understand the overall relationship between variables/features with the Target variable. Above all, I have done the necessary imports of the python packages that will help us perform various EDA techniques.

The below heat map is shown because it had most of the variation in the data.

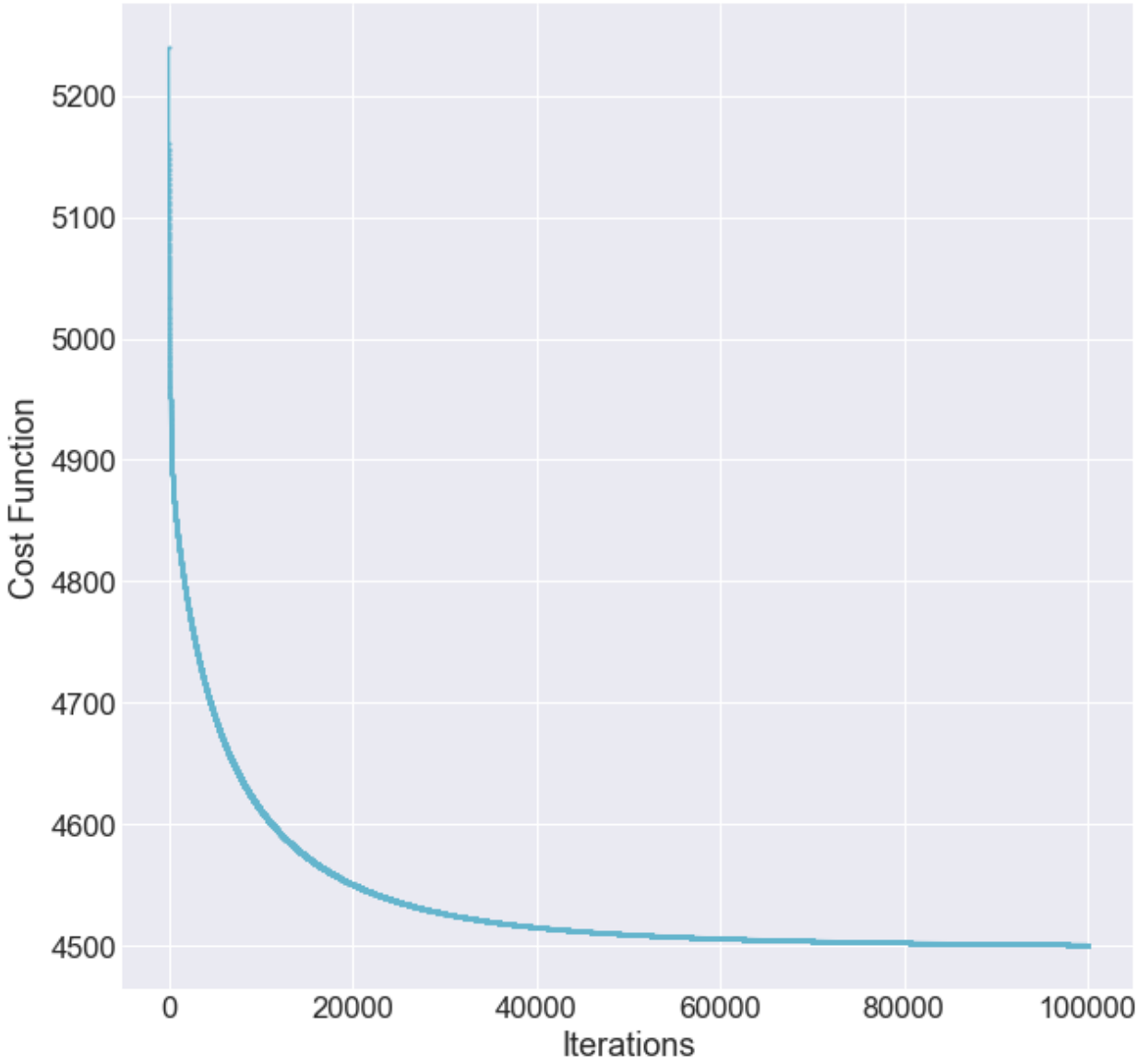


Procedure

- I began with importing the dataset into the workspace by using the pandas library which is pandas.read_csv().
- Then I viewed the whole data by using the print function using data.head(), data.shape, and so on.
- Dropped some unnecessary variable based on the decision made from the heat map.
- Split the data into 70:30 train and test respectively.
- Separated the independent and response variables into separate arrays.
- Generated the beta with initial values 0 as an array.

- Implemented the cost and gradient function under a class function called LinearRegressionModel.
- Iterated the beta values for the number of iterations to achieve convergence.

Plotted a graph between the gradient descent function by decreasing the cost function over 100000 iterations.



The cost function reduced with the help of gradient descent using batch update rule over a number of iterations. It reduced from 9894.72636 to 4500.679244723637.

The New beta values that we got are:

[2.57566421 2.44451405 1.86318234 -8.36485188 20.49026434

-11.12194277 -4.50117999 1.54946911 1.80848612 8.18820248

-9.33557493 14.93957193 -10.88064266 3.74928544 -7.29438176

0.14257553]

These values are the new coefficients which are estimated using the function.

Describe your interpretation of the results. What do you think matters the most for predicting the energy usage? What other steps you could have taken with regards to modeling to get better results?

From the results of the model, we can say that I feel the energy appliances has a significant effect on the rest of the variables as the cost function reduces as the number of iterations increases when the gradient descent function is used.