BUAN 6341 Applied Machine Learning
Assignment – 1
Linear regression: Seoul Bike-sharing Demand Data Set

Introduction:

The given data set pertains to the historical demand for bike-sharing in Seoul. The data contains the count of rented bikes on every hour of a given day from December 2017 to November 2018. This data also provides different predictors that could explain the variance in demand of bikes for rent such as temperature, humidity, wind speed, Visibility, Dew point, Solar Radiation, Rainfall, snowfall, season, and whether the day is a holiday or functioning day. Although these predictors seem to be an exhaustive list of variables that could influence the dependent variable such as the number of bikes rented, there can be many other macros independent variables such as a change in income levels in the region, Fuel prices etc, which are present in any practical data set.

Hence while trying to predict the functional relationship between the given dependent variable and independent variable, we need to factor in an error rate that is systematic and cannot be reduced.

Since our objective is to predict the outcome rather than inferring the relation between all the variables, we can use a multiple linear regression model. This linear regression model works on a 2 step basis. First, a shape or form of the function needs to be assumed and next, we can develop an algorithm to reduce the error between the predicted dependent variable and the actual independent variable.

In this case, we are going to use the most frequently and widely used concept – Gradient descent for building our linear regression model.

Algorithm:

The gradient descent model is implemented using Python programming language with NumPy and pandas packages. The batch update rule is used for the algorithm since we have a number of observations and predicters. The algorithm is implemented to give flexibility in choosing the hyperparameters that influence the outcomes and enables manipulation of the parameters to get the best results.

Below cost function is used to determine the quality of the fit

$$J(\beta_0, \beta_1) = (1/2m)[\sum(y\wedge(i) - y(i))^2$$

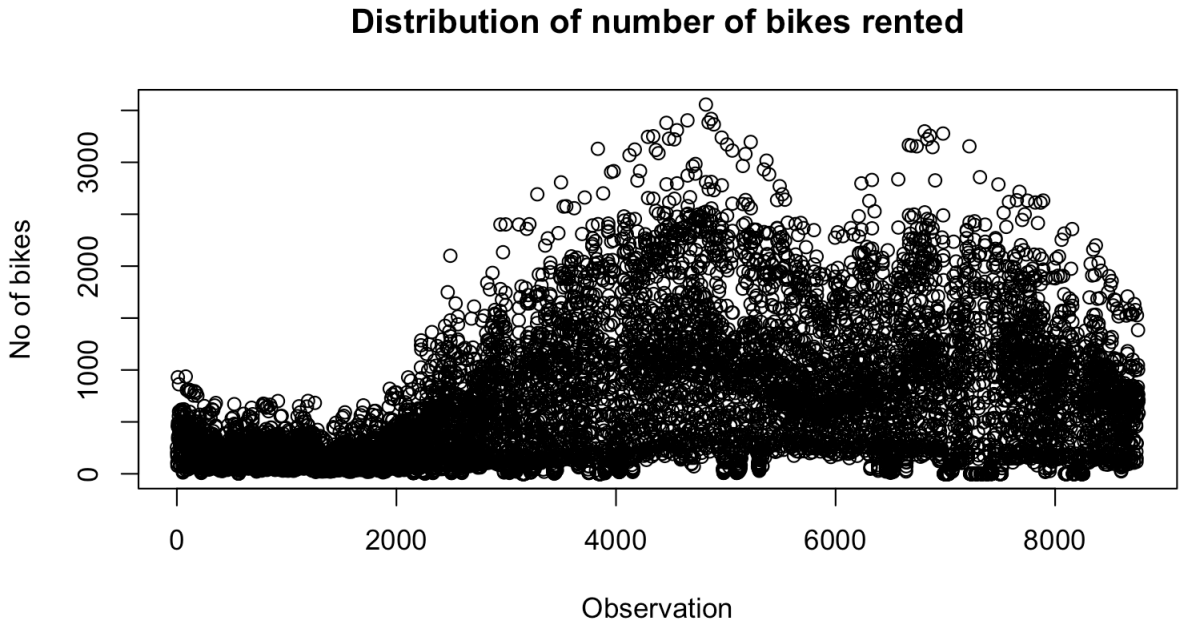
The below update rule is used to update the betas until convergence:

$$\beta_0 = \beta_0 - \alpha * 1/m \, [\sum(y\wedge(i) - y(i)]$$
$$\beta_j = \beta_j - \alpha * 1/m \, [\sum(y\wedge(i) - y(i)] \, x(i)(j)$$

Data Preparation and Pre-processing:

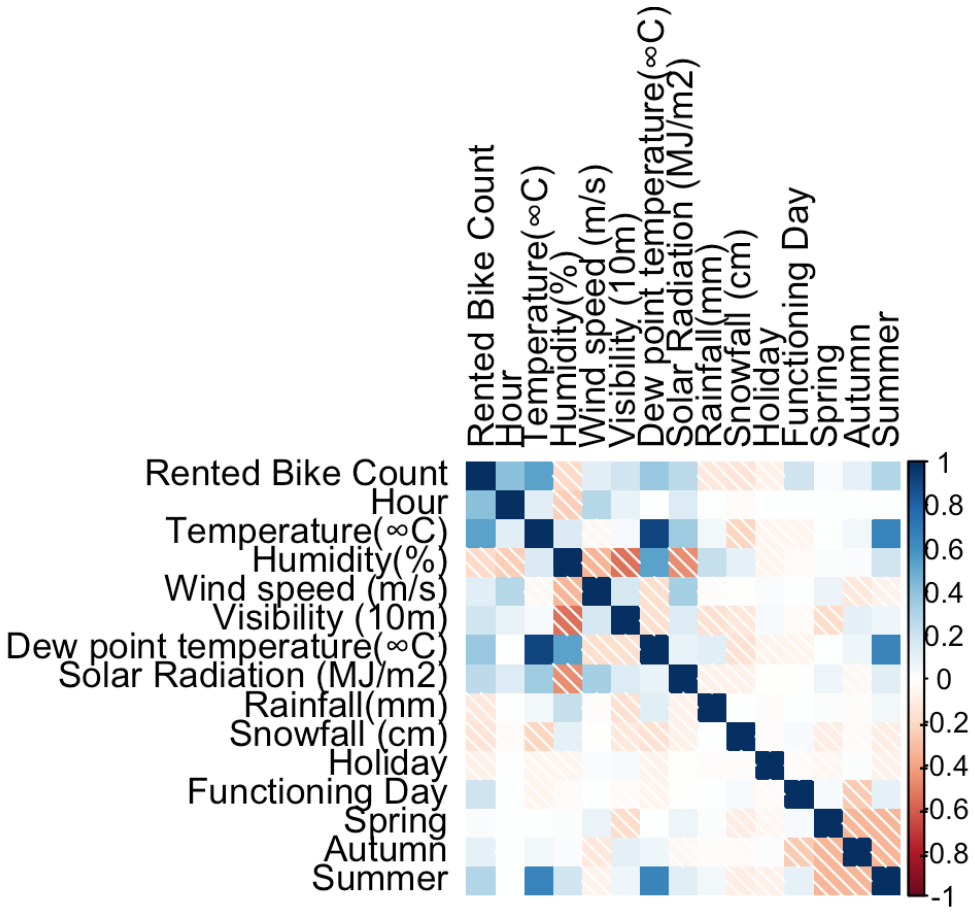
The objective of the model is to accurately predict the number of bikes that are going to be rented given the independent variables such as temperature, humidity, wind, radiation, snowfall, holiday etc.,

Firstly lets take a look at the below graph depicting the distribution of the dependent variable ie., the number of bikes rented.



Apparently, the data is almost symmetric with two peaks and a little bit left-skewed. This tells us that we can use the data as it is without any major transformations.

Also by looking at the summary of the data, the scales are not the same for each variable. For example, the time has 0-24 range and humidity has a range of 0-98. Hence all the feature variables have been scaled using the standardization method with a mean of 0 and an SD of 1.



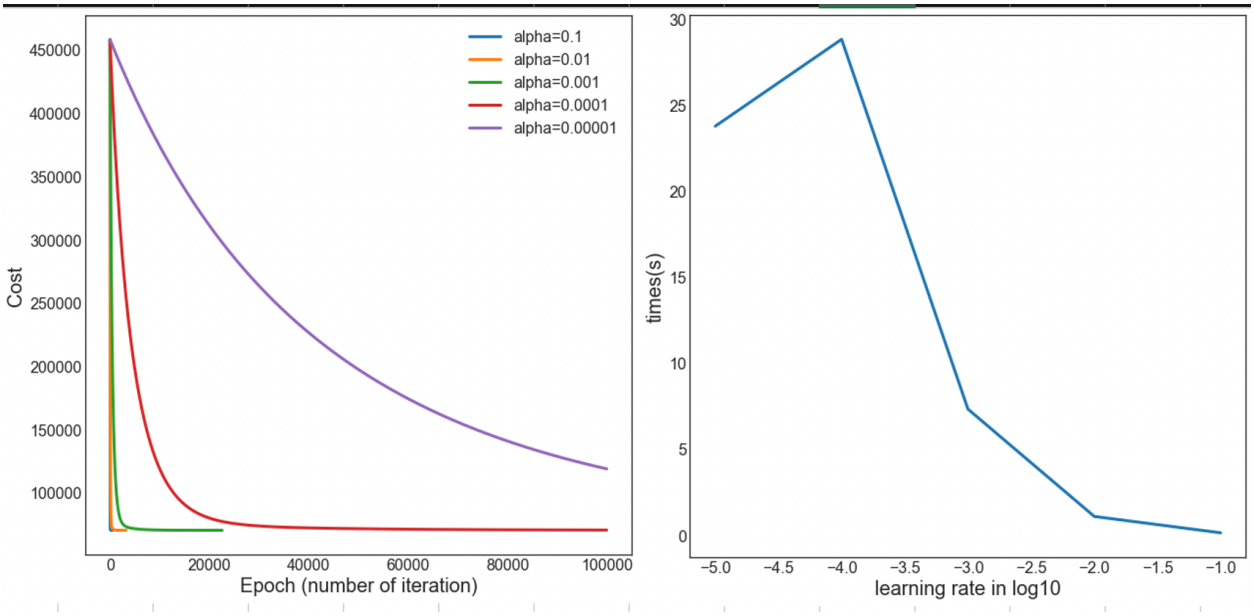
Apparently, there is a significant correlation between temperature and Dew point temperature. This correlation is understandable as temperatures will be usually high in summer and when the temperature is high, dew point temperature will also be high in general. Hence to avoid multicollinearity, dew temperature is dropped.

Also, since we have categorical variables for more than 2 levels in the season and hour variable, new dummy variables are created with one group removed as a reference group to avoid multicollinearity. And hot encoding is done for features – holidays & functioning day features since they have 2 levels.

Linear Regression funciton:

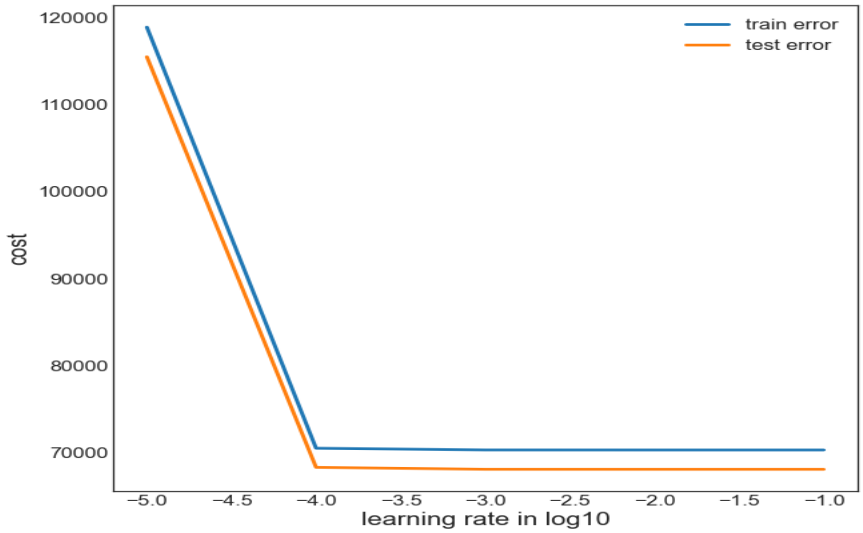
Yhat = B0 + B1(temperature) + B2(Humidity)+B3(Wind speed)+ B4(Visibility) + B5(Solar radiation)+B6(Rainfall)+B7(Snowfall)+B8(Holiday)+B9(FunctioningDay)+B10(Year)+B11(Month)+B12(Day)+B13(WeekDay Encoding)+B14(Seasons_spring)+B15(Seasons_Summer)+B16(Seasons_Winter)+B17(Hour_1) +B18(Hour_2) +B19(Hour_3) +B20(Hour_4) +B21(Hour_5) +B22(Hour_6) +B23(Hour_7) +B24(Hour_8) +B25(Hour_9) +B26(Hour_10) +B27(Hour_11) +B28(Hour_12) +B29(Hour_13) +B30(Hour_14) +B31(Hour_15) +B32(Hour_16) +B33(Hour_17) +B34(Hour_18) +B35(Hour_19) +B36(Hour_20) +B37(Hour_21) +38(Hour_22) +B39(Hour_23).

Experimentation-1:



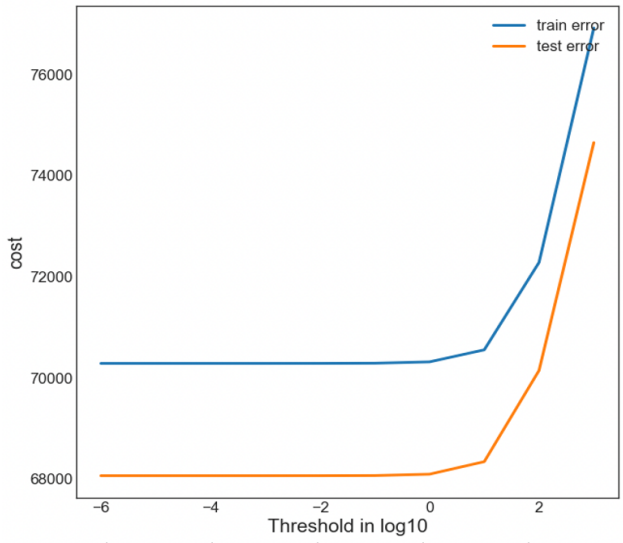
Keeping everything else constant, 5 alphas are considered for experimentation – 0.1, 0.01, 0.001, 0.0001, 0.00001 to see which of them are fast and efficient.

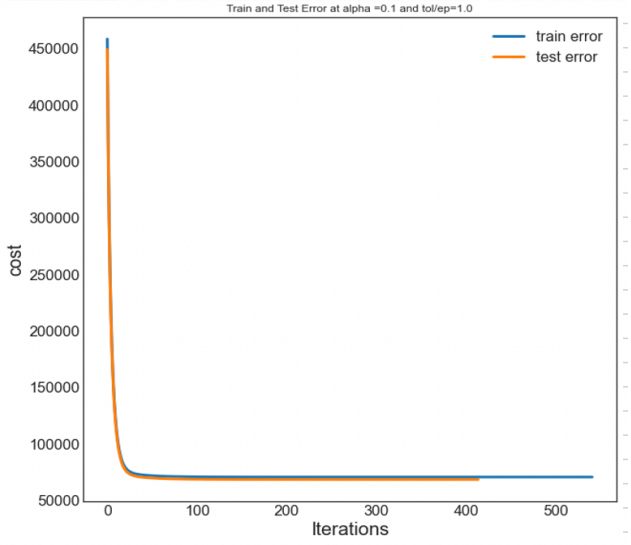
Above graph shows how the cost goes down with increment in iterations at different alpha levels. Clearly choosing alpha s of 0.1 or 0.01 or 0.001 is better as the convergence happen at less than 10,000 iterations. Choosing more than 0.001 slows down the model.



The above graph shows how the cost is reduced with an increase in alpha. At very low alpha, the cost is high due to slow learning and convergence didn't happen even within 100000 iterations.

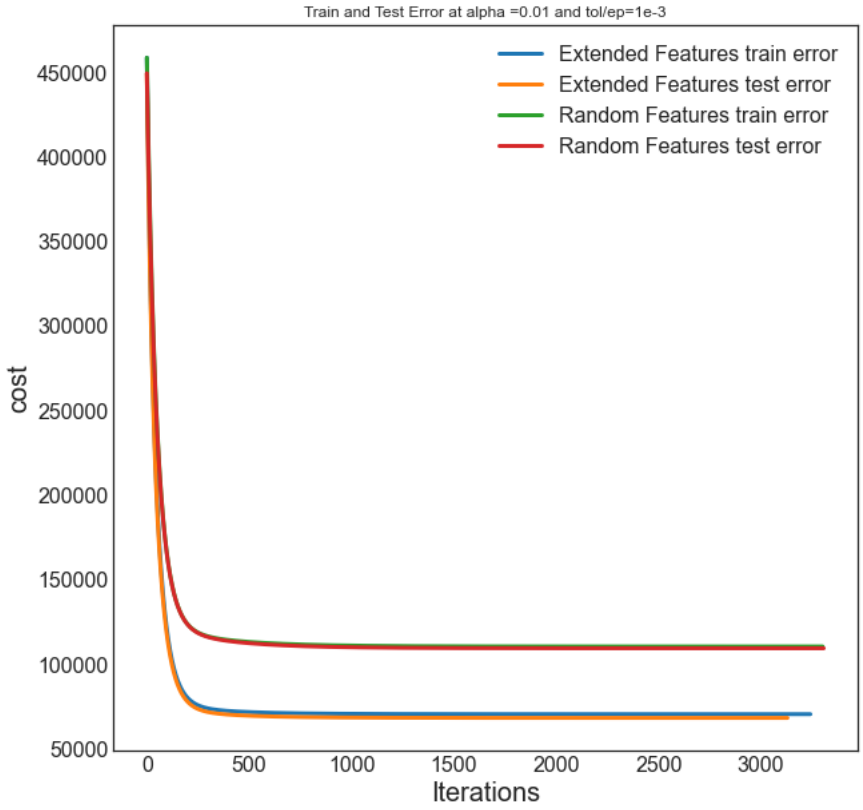
Experimentation-2: Changing Convergence threshold-





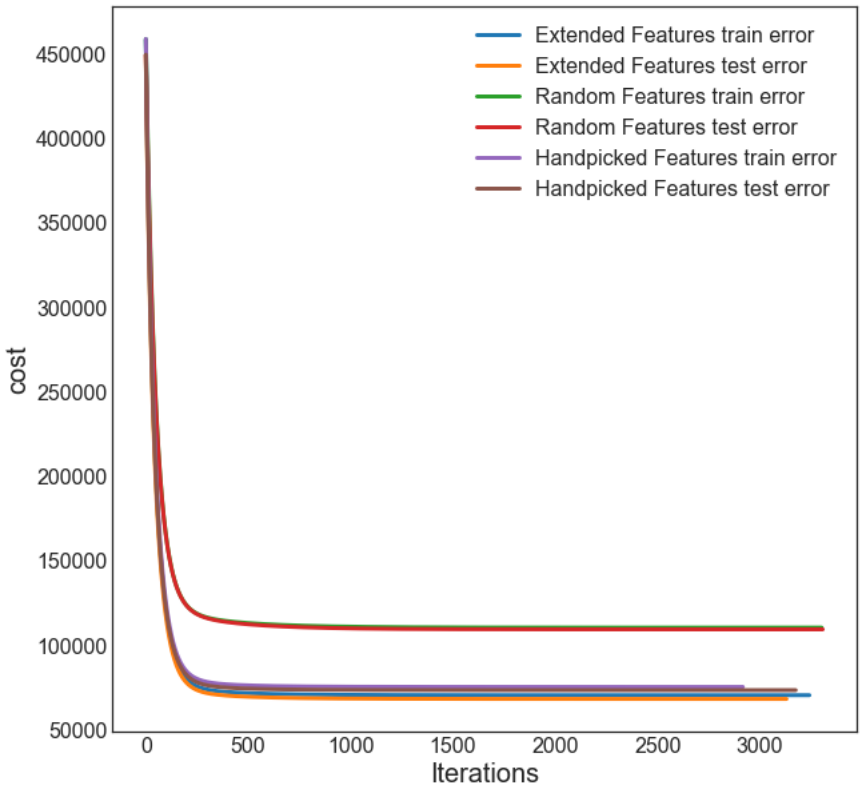
The algorithm converges when the change in cost is less than the convergence threshold. Hence the convergence would be faster at higher thresholds but too high a threshold can increase the cost. This can be seen in the graph. Apparently, conversion threshold of 1 is a better tradeoff between cost and speed of running the algorithm. Right side graph explains the cost and number of iterations at best threshold.

Experimentation-3: 8 Random features-



Above graph shows cost at different iterations for algorithm run on 8 randomly selected features(Visibility, Rainfall, Wind speed, Solar Radiation, Hour, Humidity, Dew point temperature, Holiday). Apparently, random features are having higher costs even though the time taken in a number of iterations is almost similar. Hence taking all features is better than randomly selecting.

Experimentation-4: 8 Selected features:-



The above graph shows that randomly selected features are doing worse with a higher cost than selecting all features or selecting features based on choice. This might be because the original data is well structured and captures all the information. Even if we are cautiously selecting the lower number of features the error is not significantly going down.