Rob Lindsay RXL200006

BUAN 6341

APPLIED MACHINE LEARNING

ASSIGNMENT 1

LINEAR REGRESSION

THROUGH GRADIENT DESCENT

EXECUTIVE SUMMARY:

- **Smaller Alphas require significantly more iterations which can be problematic on large datasets with many observations or variables**
- **Larger Alphas can result in cost functions that are relatively close to smaller alpha cost functions**
- **The convergence threshold will require more iterations but not necessarily contribute to significant added value**
- **Improved Training results do not necessarily mean improved Test results.**
- **Reduced variables can still provide good results provided they are chosen with proper analysis**

Objective: Design a Linear Regression Model utilizing Gradient Descent to see how data manipulation, categorical encoding, and hyperparameter settings can impact the outcome.

Data Source: Utilized the Seoul Bike sharing Demand Data Set available for download at https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand

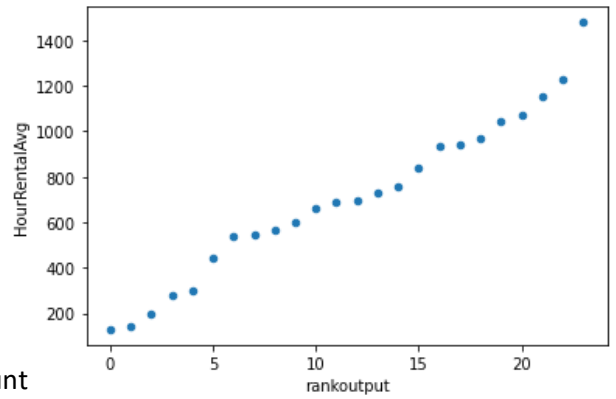
Data Issues: The header contained special characters (the degree symbol) which I altered before loading into Python. Upon reading the file in, the first header name was missing as well so I had to adjust that.

Data Structure: The data contained 14 columns of data with 8760 observations. The variables originally consisted of the following:

Date	Rented Bike Count	Hour	Temperature(dC)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(dC)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day

Data Transformation: After creating an initial train/test split, I was able to utilize mean values from the training set to transform my data. I first approached this by identifying several variables that were categorical in nature.

- *Seasons* was clearly categorical with its classification of the 4 seasons. It was an ideal candidate for dummy encoding as three seasons would be capable of encoding the 4th, but I decided to instead employ Targeted Encoding in which I assigned each season it's mean value for Rented Bike Count.
- I also took a categorical view point of the *Date* and *Hour* variables. I converted *Date* to a Day of The Week indication with the thought that there might be correlations in weekend or weekday rentals.
- For *Hour*, I ranked the hours based on their average Rented Bike Count. After plotting the ranked output, it was clear that there was a linear relationship, with the main exception being in the peak hour. Wanting to account for this, I created a *Peak Hour* variable with one hot encoding for the peak hour. One downfall to this is that the peak hour variable was dependent upon the train test split. If I ran different iterations, I would sometimes see two peak hour trends. I chose to not account for this particular anomaly and instead encoded my approach through a single peak hour variable designation.



Scaling of the Data: After the transformation, I had 14 independent variables to aid me in determining the predicted values of *Rented Bike Count.* To ensure that the regression would perform without bias for any particular column, I scaled the columns that did not already appear to have a distribution approximately between 0 and 1. I choose the following approach for scaling: $\frac{(x_i^C - \mu_{TS}^C)}{MAX_{TS}^C - MIN_{TS}^C}$ where TS represents the training set and C is the column.

Correlation Inspection:

	Rented Bike Count	DOWAdjust	HourRank	PeakHours	Temperature(dC)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(dC)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	SeasonalMeanAdj	Holiday	Functioning Day
Rented Bike Count															
DOWAdjust	0.06														
HourRank	**0.53**	0.00													
PeakHours	0.26	-0.00	**0.35**												
Temperature(dC)	0.54	0.02	0.14	0.04											
Humidity(%)	-0.20	0.02	-0.28	-0.07	0.16										
Wind speed (m/s)	0.12	0.02	**0.34**	0.10	-0.04	**-0.34**									
Visibility (10m)	0.20	-0.00	0.11	0.04	0.04	-0.54	0.17								
Dew point temperature(dC)	**0.38**	0.02	-0.00	0.00	**0.91**	0.54	-0.18	-0.18							
Solar Radiation (MJ/m2)	0.26	0.01	0.18	-0.04	0.35	**-0.46**	**0.33**	0.15	0.09						
Rainfall(mm)	-0.12	-0.01	0.01	0.02	0.05	0.24	-0.02	-0.17	0.13	-0.07					
Snowfall (cm)	-0.14	0.01	-0.02	-0.01	-0.22	0.11	-0.00	-0.12	-0.15	-0.07	0.01				
SeasonalMeanAdj	**0.46**	0.04	0.00	-0.00	**0.85**	0.27	-0.13	0.04	**0.83**	0.18	0.07	-0.21			
Holiday	-0.07	-0.01	-0.00	0.00	-0.06	-0.05	0.02	0.03	-0.07	-0.01	-0.01	-0.01	-0.11		
Functioning Day	0.20	0.08	0.01	0.00	-0.05	-0.02	0.01	-0.03	-0.05	-0.01	0.00	0.03	-0.07	-0.03	

Final Correlation Matrix

It is not surprising to see the high correlation between SeasonalMeanAdj and Temperature nor was it surprising to see strong correlations for the Temperature and SeasonMeanAdj with the Rented Bike Count. You would also expect to see high correlation with HourRank and Rented Bike Count.

I was surprised to see that there was very little correlation when I added the DOWAdjust to take into account the Day of the Week. I also thought I would see a stronger negative correlation between Rainfall and Rented Bike Count. Trying to rationalize this, I hypothesize that this could be the result of people who would normally walk choosing to rent a bike and therefore offsetting people who normally ride but choose not too because of the rain.

IMPLEMENTAION OF GRADIENT DESCENT

The gradient descent function was implemented at a batch level, meaning all Beta values were updated simultaneously. Cost was driven by the Sum of Squares of the residual errors.

Cost Function: $J(\beta) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2$

Beta Update $\beta_j = \beta_j - \frac{\alpha}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 x_j^{(i)}$

Experiment 1 – Learning Rate and Iteration Length

Purpose: Identify learning rate and iteration hyperparameter values.

Takeaways: After spending a significant amount of time initially trying to fine tune my learning rate over a large period of iterations, I realized the following:

- Relatively large variances in the rate that I choose would still lead to relatively similar cost function results for both the Train and Test Data set
- Just because you were reducing your Cost Function didn't mean that your Test results would get better
- Iterations longer than 5000-10000 could take significantly more time if convergence wasn't met.
- Narrowing in by using 50 iterations to start seemed to provide much faster hyperparameter tuning for this data set

The charts on the following page provide a snapshot of my approach to hyperparameter tuning with some variations included to acknowledge the learning aspect that too large of a value will cause divergence. The alpha values progress from left to right as (.85 , .825 , .8 , .65 , .35) and iterations work from top to bottom as (30000 , 10000 , 1000 , 50).

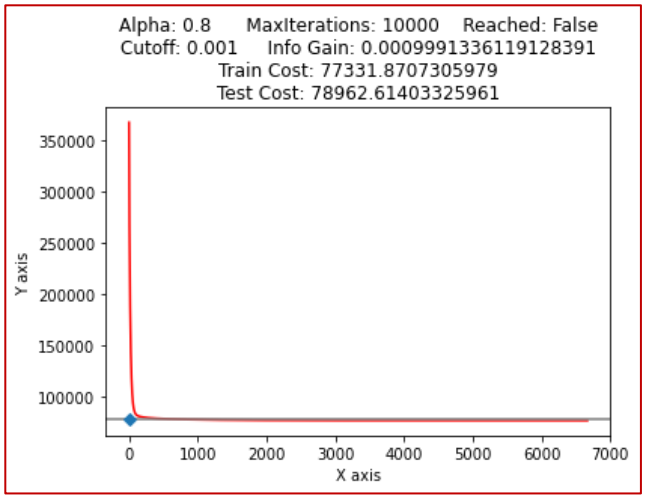
On my computer, for this data set, the 10000 iteration setting for a learning rate of .80 does not take more than 15-20 seconds before it reached it's .001 convergence threshold between 6000 and 7000 iterations. Satisfied with it's Training Cost Convergence and the Test Cost Comparison amongst its competitors, I am comfortable selecting this pair as my candidate for regression analysis. The resulting Beta Equation takes the following form:

Rented Bike Count $= \beta_0 X0 + \beta_1$ DOWAdjust$+ \beta_2$ HourRank$+ \beta_3$ PeakHours$+ \beta_4$ Temperature(dC) $+ \beta_5$ Humidity(%)$+ \beta_6$ Wind speed (m/s)
$+ \beta_7$ Visibility (10m) $+ \beta_8$ Dew point temperature(dC) $+ \beta_9$ Solar Radiation (MJ/m2) $+ \beta_{10}$ Rainfall(mm) $+ \beta_{11}$ Snowfall (cm)
$+ \beta_{12}$ SeasonalMeanAdj$+ \beta_{13}$ Holiday$+ \beta 1_4$ Functioning Day

$= -530.2688 + 24.0667 *$ DOWAdjust $+ 462.5577 *$ HourRank $+ 269.1883 *$ PeakHours $+ 77.71069 *$ Temperature(dC) $- 382.9884 *$ Humidity(%)
$- 123.4211 *$ Wind speed (m/s) $+ 12.8274 *$ Visibility (10m) $+ 335.9771 *$ Dew point temperature(dC) $+ 104.4145 *$ Solar Radiation (MJ/m2)
$- 1885.2996 *$ Rainfall(mm) $+ 61.0107 *$ Snowfall (cm) $+ 607.4729 *$ SeasonalMeanAdj $- 96.9215 *$ Holiday $+ 844.7965 *$ Functioning Day

Since we scaled our variables, we can easily see which variables have the largest effect on our predicted Rented Bike Count. Hour Rank, Peak Hours, Humidity, Dew point, Rainfall, Season, and Functioning Day appear to be the largest contributors to our predicted count with both positive and negative influences on the outcome. However, we should keep in mind that Seasonality and Dew Point are highly correlated. Temperature, Wind speed, Solar Radiation, Holiday and Snowfall have a medium effect on the predicted outcome with Day of the Week and Visibility having very little effect.

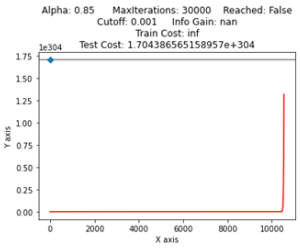
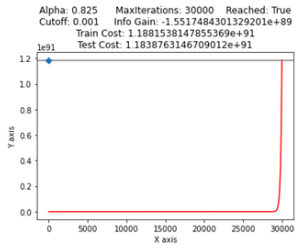
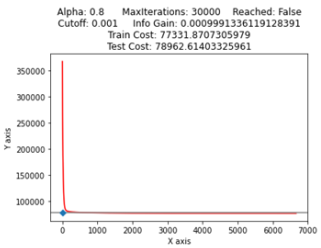
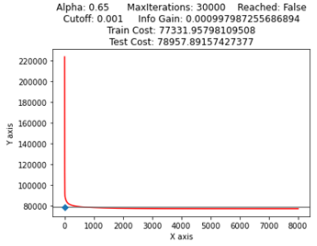
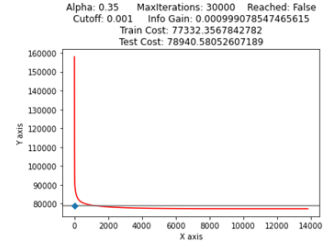
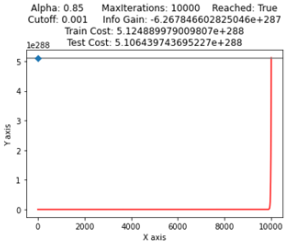
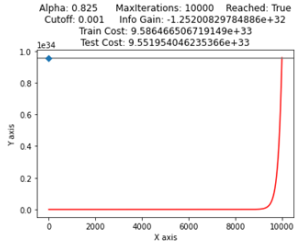
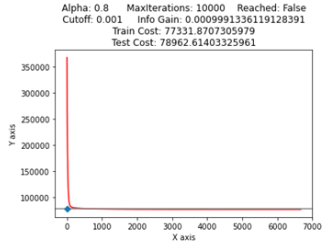
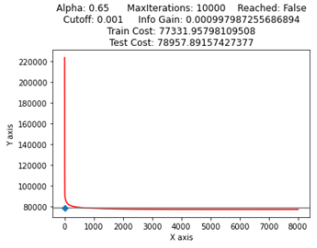
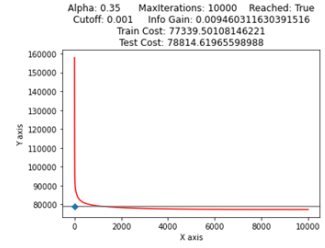
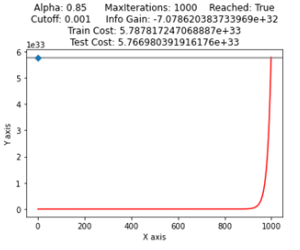
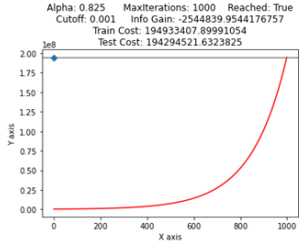
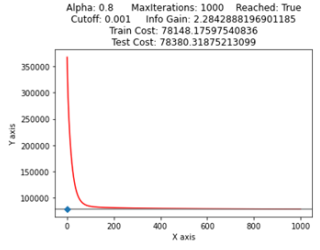
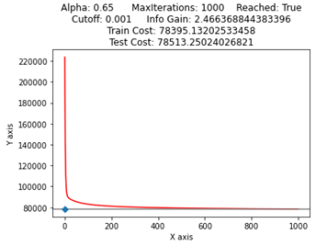
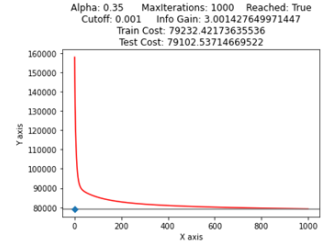
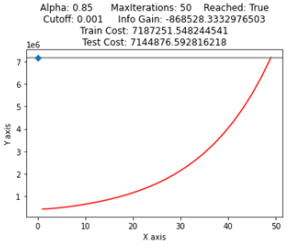
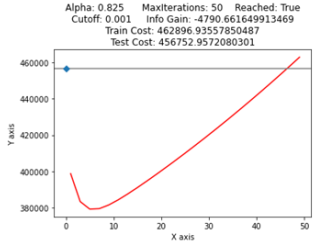
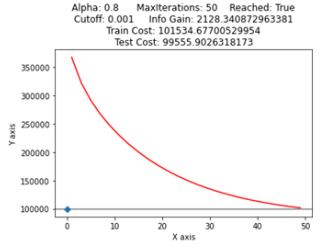
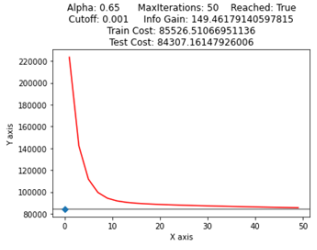
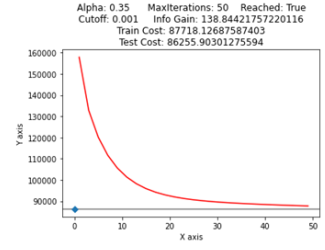
Note: For the plots, the red line indicates the Cost Function as the iterations increase. The blue dot indicates the Test Set Cost Value with the grey line extending across to provide a visual comparison of how close the Test Cost Value is to the Train Cost function.



For my experiment, the best Test Cost Value is found at alpha .8. The best Training Cost Function was found by both alpha = .8 and alpha = .65 at 77331 with minor variations in the remaining decimal value. Alpha = .35 would provide better results with a decreased convergence threshold but would require siginificantly more iterations. At the .001 threshold, it is already taking about twice the amount of iterations as the .8 and .65 alphas.

Alpha

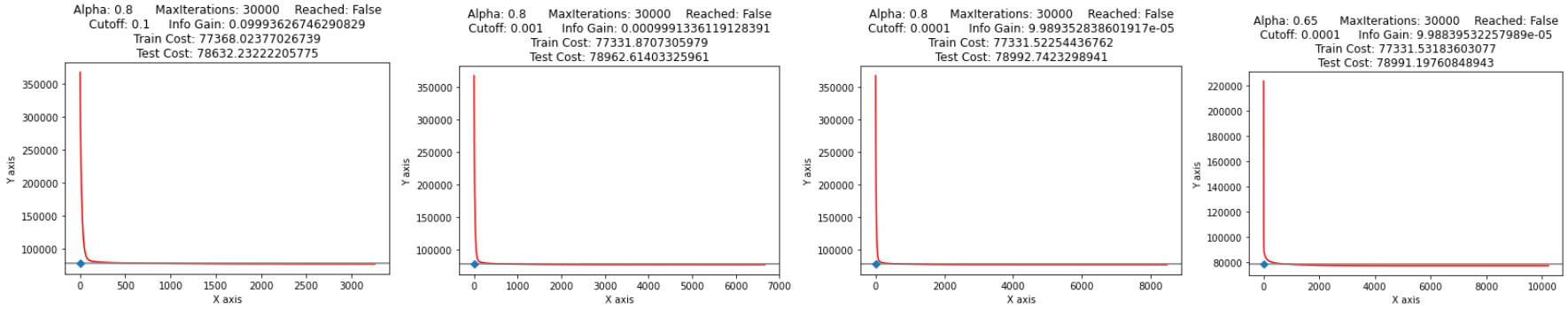
MAX ITERATIONS

	.85	.825	.8	.65	.35
30000					
10000					
1000					
50					

Experiment 2 – Convergence Threshold

Having settled on an alpha of .8 and knowing the convergence threshold of .001 was met in under 10000 iterations, any increases in the threshold will result in even faster convergence threshold being met, but this should result in diminished training cost function results. Increasing the threshold will reduce the time that it takes to meet the convergence threshold. For our data set and the time it takes to run the .001 threshold, there isn't much advantage in time savings. However, if this were a significantly larger data set, we might consider time savings in the Convergence Threshold analysis. So the primary focus for this data set is comparing Cost Functions across different thresholds while ensuring that smaller thresholds don't result in cumbersome learning time increases. Since we are going to reduce the threshold, we will push the max iteration setting back to 30000 to allow for convergence within a reasonable amount of time (approximately 1 – 2 minutes for this data set on my computer)

As expected, setting the convergence threshold to .1 (left) resulted in the largest shift of the Cost Function for the training set. However, it had the best Cost Function results for the Test set. This could be the result of overfitting or simply do to the fact that there are other factors in nature that we are not capturing in our data set. Looking at the remaining data sets, we actually see this trend continue. The smaller the convergence threshold, the larger the Cost Function of the Test Set. Looking at our time increases, despite lowering the convergence threshold all the way to .0001(right center), convergence was still met in under 10,000 iterations. Had we chosen a smaller alpha from experiment 1, i.e. .65 (right), the convergence would have been met shortly after 10,000 iterations, but even this should not be considered unreasonable.



For the purpose of this assignment, I will keep my previously established convergence threshold of .001(left center). However, if I truly wanted to improve upon this, I would take random samplings to form my Training and Test sets to see if the .1 (or some other higher number than .001) convergence threshold consistently outperformed for the Cost Function on the Test sets. You could extend this practice to create new Beta Equations as well and compare both the resulting Training Cost Function results as well as the Test Cost Functions. You would then be able to determine an alpha and convergence threshold pair that fit into your iteration time tolerances and provided robust results across most Training and Testing Data Sets.

Experiment 3 – Random Variable Reduction to 8

Using a random.sample function, the following variables were dropped:

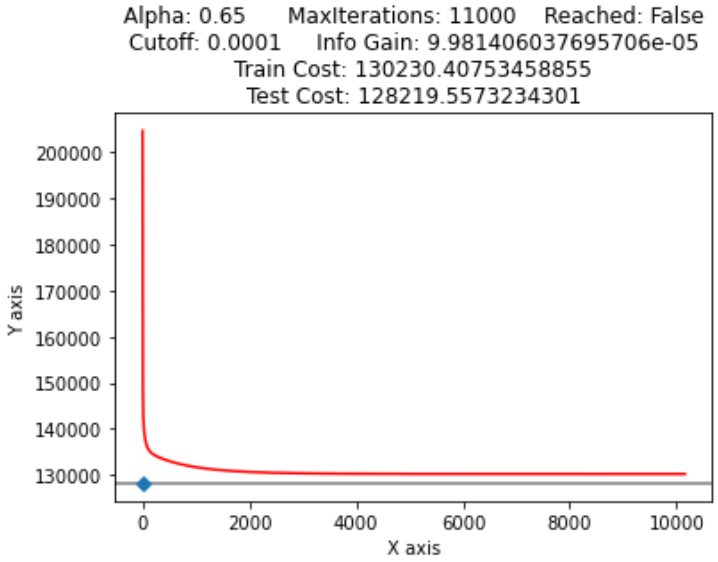
```
['Functioning Day', 'PeakHours', 'Temperature(dC)', 'Dew point temperature(dC)', 'Holiday', 'SeasonalMeanAdj']
```

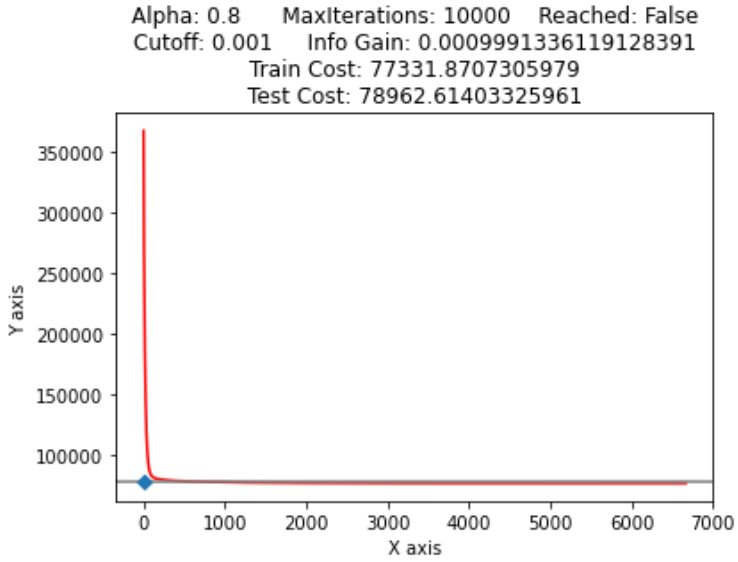
Notice that Functioning Day, Peak Hours, Dew point, and Season all played a significant role in our initial Beta Values. We would expect to see very different Beta Values in our new Beta Equation as well as new alpha, convergence threshold and iteration requirements.

Using similar techniques as described in the previous two sections, I have come to a determination that alpha = .65, Convergence threshold of .0001, and max iterations of 11000 are sufficient to produce the following Beta Equation:

Rented Bike Count = 706.635 + 46.9266 * DOWAdjust + 592.0051 * HourRank + 189.2093 * Humidity(%) + -413.7901 * Wind speed (m/s)
+ 104.6116 * Visibility (10m) + 488.166 * Solar Radiation (MJ/m2) + -1904.2219 * Rainfall(mm) + -1354.8525 * Snowfall (cm)

When comparing the Cost Function for our randomly chosen 8 variables (left) to that of our original Model (right) , there is a significant increase in the Cost as we reduced our variables.



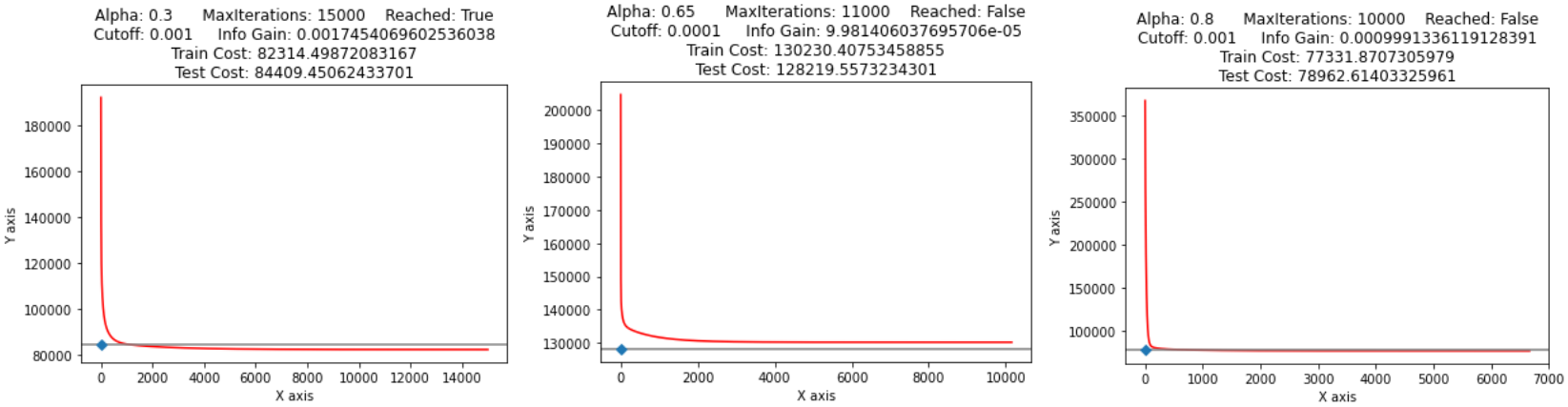


Experiment 4 – Custom Variable Reduction to 8

As noted in Experiment 1, Hour Rank, Peak Hours, Humidity, Dew point, Rainfall, Season, and Functioning Day appear to be the largest contributors to our predicted count. Removing Dew Point, we have 6 variables identified. From our previously identified medium effect variables, we know that Temperature is highly correlated with Season, so we can remove Temperature for consideration. With Wind speed, Solar Radiation, Holiday and Snowfall remaining, I will choose to go with Holiday and Snowfall as the remaining two variable as they have the least amount of correlation between our other variables. The variables we are dropping are:

```
['DOWAdjust', 'Wind speed (m/s)', 'Visibility (10m)', 'Dew point temperature(dC)', 'Solar Radiation (MJ/m2)', 'Tem
perature(dC)']
```

Following similar procedures as before, we come up with an alpha of .3, Max Iterations of 15000 and convergence threshold of .001.



As you can see, both the Train Cost and Test Cost are significantly improved on our new model (left) when compared to the random variable experiment(center). This is likely a result of using analytical insights from previous steps and visualization techniques to determine the most likely contributors to the best fitting solution.

When comparing to the original model (right), there isn't a huge drop-off in the Cost Functions. If this database were huge, the time savings and memory savings of the variable reduction could be significant in determining if the abridged model defined in this experiment was good enough for our purposes. I am not surprised that there was a significant increase from the random variable selection. I wasn't surprised by the small drop in the performance of the custom variable selection when compared to our original estimation model. Had we had significantly more variables, I may have expected a slight increase in performance here as we may have been capturing too much noise, but just dropping a few variables which in general did not have a huge impact on the calculations shouldn't result in too much of a change in either direction.