BUAN 6341 Applied Machine Learning

ASSIGNMENT NO 1

Appliances energy prediction

Executive Summary

- **Optimizing Hyperparameters like learning rate and convergence threshold improves accuracy and time taken to converge.**
- **Sensible feature selection and preparation improve prediction accuracy.**
- **Temperature, Functioning Day and Hour are the best parameters to predict the bike rental count .**
- **Prediction accuracy can be further improved by including time variables.**

Introduction

In this project, the objectives were to implement a gradient descent algorithm and utilize it to predict the rental bikes demand in urban cities for mobility comfort. This report details the various experiments conducted using the dataset to understand the effect of tuning hyperparameters of the gradient descent algorithm. Also, the effect of the use of various independent variables in prediction is evaluated and the best model was selected through experimentation.

About the Data

The dataset consists of 14 features and 8760 records. The data is the rented bike count captured at each hour and weather conditions (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), along with date information. To remove the serial correlation with time, the hours variable is converted to dummies to predict.

Project Outline

The Project is outlined to have 3 parts:
Part 1: Data Preparation and Exploratory Data Analysis

Part 2: Use Linear Regression Model for Rental Bike count Prediction and Report equation

Part 3: Implement the gradient descent algorithm with batch update rule and carry out experiments with different values of

i) Learning rate,

ii) Threshold and report the best fit values,

iii) Pick random 8 features and train the model with the best hyper parameters,

iv) Hand pick the best 8 features based on our assumption and train the model with the best hyper parameters.

Algorithm Implementation

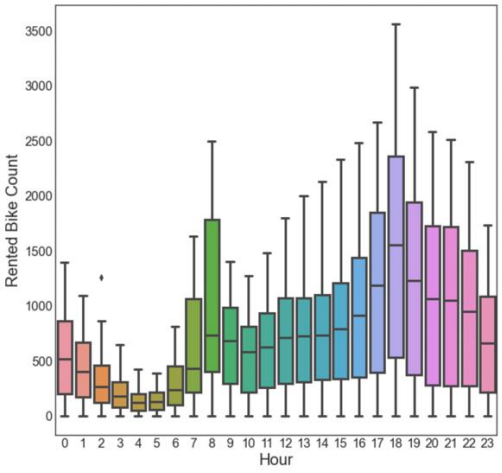
The gradient descent algorithm is implemented using the python numerical computation package “numPy”. The NumPy package provides new homogenous array and matrix data structures to python which is immensely beneficial to implement vectorized implementation of the gradient descent.

The algorithm is implemented for one machine learning technique, namely, Linear regression. The algorithm is implemented with options to change the hyperparameters: Learning rate, convergence threshold, and max epoch.

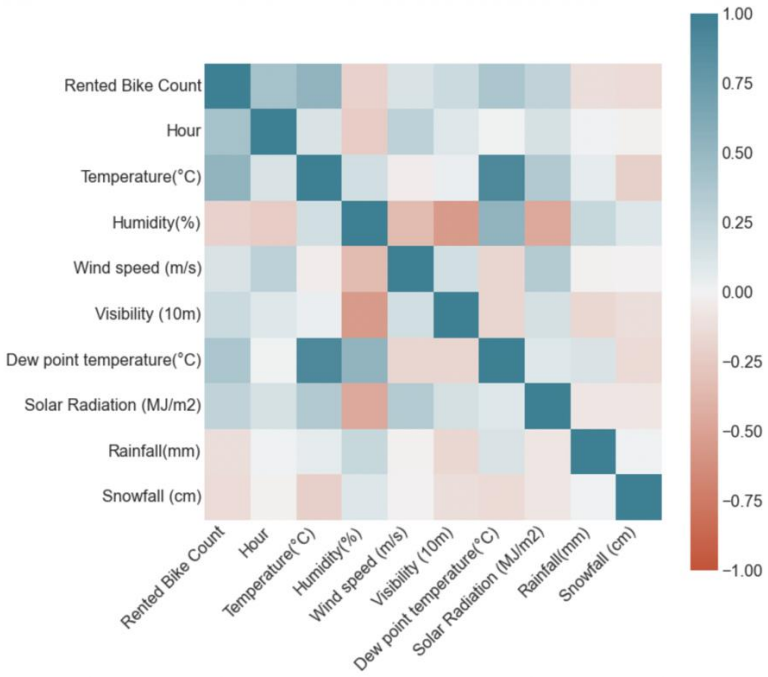
PART 1:

Data Preparation and Exploratory Data Analysis

The objective of the linear regression model is to accurately predict the bike count required at each hour for the stable supply of rental bikes.



To remove the serial correlation of errors, the hour feature converted to dummies.



The correlation plot shows a high correlation between Temperature and Dew point temperature, and a negative correlation between Visibility and Humidity which was expected. This can be due to some calibration issues of the device. In further analysis dropping one of the variables from each can increase the predictivity.

We will drop the Dew point temperature(C) feature, on the other hand, the Seasons feature is converted to dummies using hot encoding. Holiday and Functioning Day are ordered data converted to numeric values by replacing yes and no with 1 and 0 respectively. The date is also converted to extended features as separated by Day, Month, Year, etc.

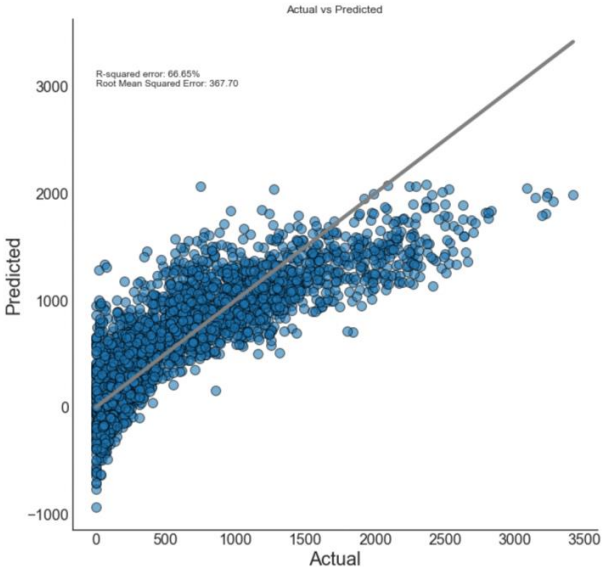
The gradient descent algorithm performs much better when all features have a similar scale. So, all features are standardized demeaning and dividing standard deviation. This makes all variables have 0 mean and 1 SD.

PART 2:

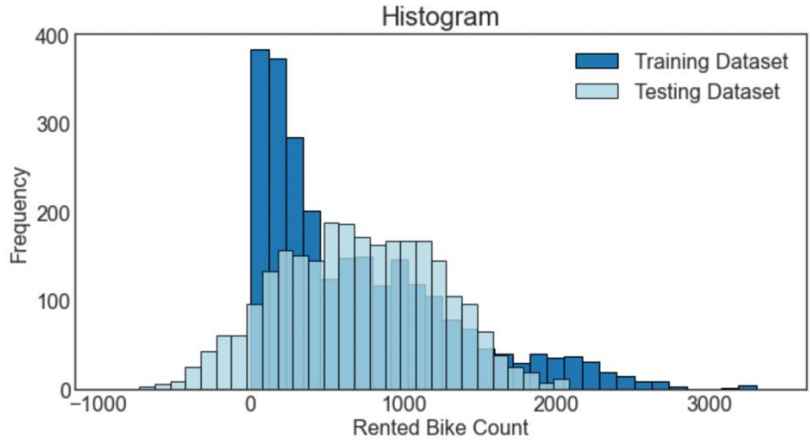
Linear Regression general equation:

Rented Bike Count = B0+ B1*Temperature(°C) + B2*Humidity(%) + B3*Wind speed (m/s) + B4*Visibility (10m) + B5*Solar Radiation (MJ/m2) + B6*Rainfall(mm) + B7*Snowfall (cm)+ B8*Holiday + B9*Functioning Day + B10*Year + B11*Month + B12*Day +B13*Autumn + B14*Spring + B15*Summer + B16*Hour_0 + 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 + B38*Hour_22

Rented Bike Count = 702.58+ 276.60*Temperature(°C) -141.96*Humidity(%) - 4.90*Wind speed (m/s) + 3.60*Visibility (10m) + 61.95*Solar Radiation (MJ/m2) - 63.50*Rainfall(mm) + 9.05*Snowfall (cm) - 30.02*Holiday + 166.09*Functioning Day - 156.67*Year - 149.76*Month - 4.88*Day + 332.17*Autumn + 154.09*Spring + 212.14*Summer – 20.61*Hour_0 – 42.99*Hour_1 - 64.01*Hour_2 – 82.92*Hour_3 – 88.23*Hour_4 – 89.58*Hour_5 – 55.77*Hour_6 + 4.07*Hour_7 + 75.18*Hour_8 - 15.75*Hour_9 – 63.94*Hour_10 - 61.25*Hour_11 – 60.47*Hour_12 – 59.60*Hour_13 – 57.25*Hour_14 – 41.79*Hour_15 – 13.99*Hour_16 + 35.90*Hour_17 + 128.03*Hour_18 + 86.30*Hour_19 + 62.30*Hour_20 + 66.99*Hour_21 + 43.91*Hour_22



The model score on the test dataset is 66.67% and the mean square error is 135206.245 and the curve indicates that errors are non-linear and indicate heteroskedasticity.



From the distribution plot, we can infer that the Rented Bike Count in the training dataset is right-skewed whereas in the testing dataset is almost normally distributed.

Experimentation and Results

Mainly four experimentations were undertaken using the dataset in which we explored the effect of changing the hyperparameters of the algorithm and discuss the effectiveness of feature selection in predictive accuracy. These experiments were repeated for linear regression. Finally, the results of the experiments were discussed in the results section.

Experimentation

1. Hyperparameter tuning

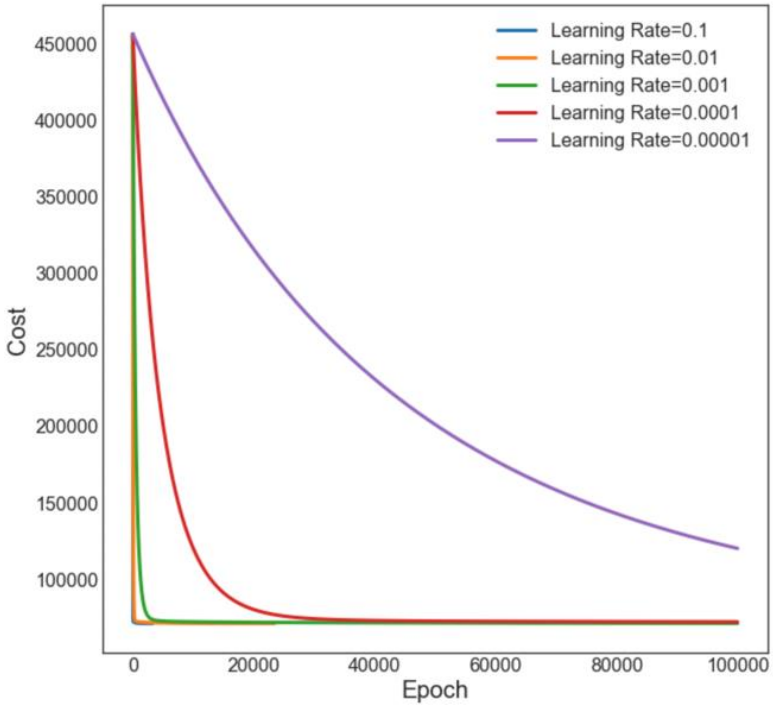
In linear regression implemented using a gradient descent algorithm, the hyperparameter we can tune is the learning rate ($\propto$). The value of the learning rate must be tuned for optimal convergence to estimates (β values).

Using Gradient Descent Implementation batch rule at learning rate = 0.1:

Rented Bike Count = B0+ B1*Temperature(°C) + B2*Humidity(%) + B3*Wind speed (m/s) + B4*Visibility (10m) + B5*Solar Radiation (MJ/m2) + B6*Rainfall(mm) + B7*Snowfall (cm)+ B8*Holiday + B9*Functioning Day + B10*Year + B11*Month + B12*Day +B13*Autumn + B14*Spring + B15*Summer

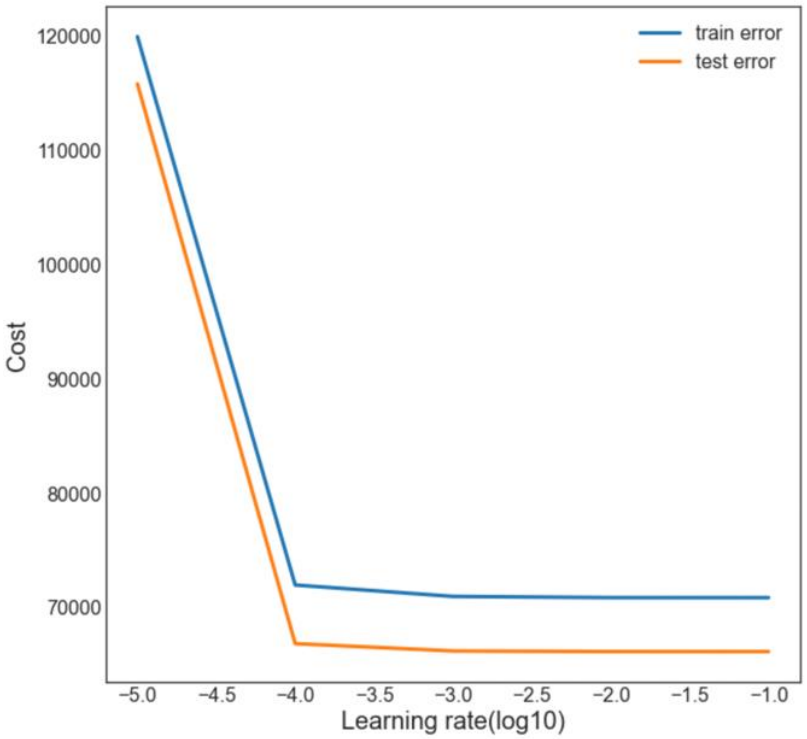
+ B16*Hour_0 + B17*Hour_1 + B18*Hour_2 + B19*Hour_3 + B20*Hour_4
+ B21*Hour_5 + B22*Hour_6 + B23*Hour_7 + B24*Hour_8 + B25*Hour_9
+ B26*Hour_10 + B27*Hour_11 + B28*Hour_12 + B29*Hour_13 + B30*Hour_14
+ B31*Hour_15 + B32*Hour_16 + B33*Hour_17 + B34*Hour_18 + B35*Hour_19
+ B36*Hour_20 + B37*Hour_21 + B38*Hour_22

Rented Bike Count = 702.58+ 276.90*Temperature(°C) -141.82*Humidity(%) - 4.91*Wind speed (m/s) + 3.66*Visibility (10m) + 62.06*Solar Radiation (MJ/m2) - 63.52*Rainfall(mm) + 9.02*Snowfall (cm) - 30.03*Holiday + 166.08*Functioning Day - 153.03*Year – 145.70*Month - 4.86*Day + 327.39*Autumn + 152.40*Spring + 208.69*Summer – 20.60*Hour_0 – 42.97*Hour_1 - 64.00*Hour_2 – 82.89*Hour_3 – 88.21*Hour_4 – 89.56*Hour_5 – 55.76*Hour_6 + 4.08*Hour_7 + 75.18*Hour_8 - 15.76*Hour_9 – 63.95*Hour_10 - 61.28*Hour_11 – 60.51*Hour_12 – 59.64*Hour_13 – 57.29*Hour_14 – 41.82*Hour_15 – 14.02*Hour_16 + 35.88*Hour_17 + 128.02*Hour_18 + 86.29*Hour_19 + 62.31*Hour_20 + 66.99*Hour_21 + 43.91*Hour_22

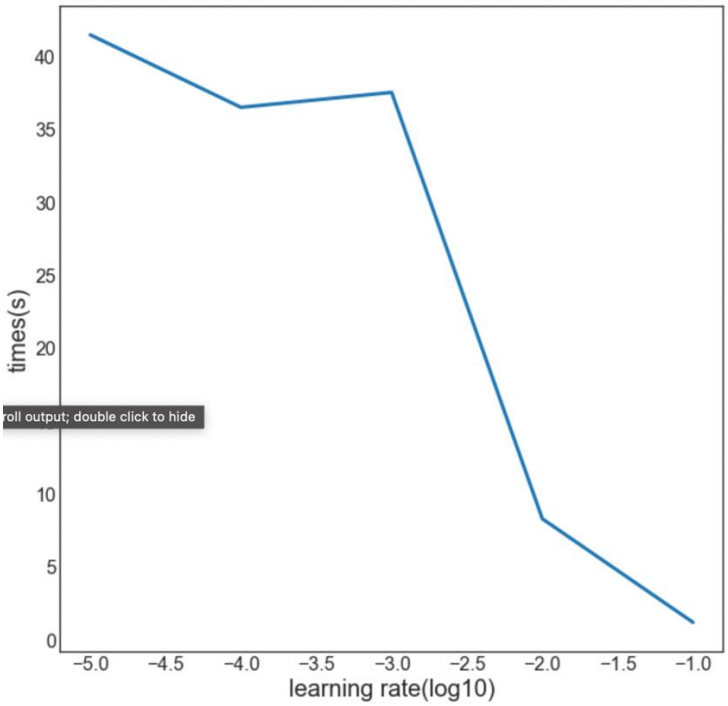


With each epoch, the cost is seen continuously decreasing and the algorithm is said to have reached convergence when the decrease in cost is within a defined threshold. When the learning rate is low, the algorithm takes many iterations to converge while the estimates can be more precise. When the learning rate is very high, the algorithm might

diverge and never reach the minimal point. At Alpha 0.0001 and at 0.00001 algorithms never reaches a global minimum point even at 100000 iterations



The figure shows train & test Costs as a function of learning rate(log). As we can observe, when the learning rate is very low ($\propto$ = 0.00001), the cost is higher. This is because of slow learning, even at 100000 iterations, the algorithm did not converge within a tolerance value. The effect of changing tolerance values can be seen in a later experiment. At the learning rate (($\propto$ = 0.0001) even though the algorithm didn't converge to the global minimum after 100000 iterations, it is close to a global minimum. We can see that from 0.1 to 0.0001 the global minimum for cost function is almost the same (very less value of change is observed) but nr of iterations are more. We can also say that algorithm for learning rate between range 0.7~0.0001 the cost function change is very minimal and after 0.7 the learning rate fails to reach the global minimum.



Here, we explore the time taken for the algorithm to converge as a function of the learning rate. At a learning rate = 0.001, the time is taken to complete the run is higher than 0.00001. In general, we can see a linear increase in the time taken when the learning rate is decreased.

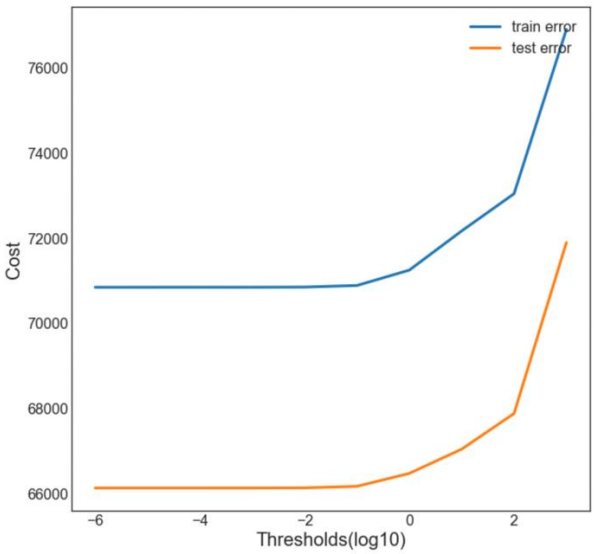
Learning Rate	Train Error	Test Error
0.1	70840.1044	66120.73972
0.01	70843.8137	66124.35576
0.001	70947.1438	66175.92321
0.0001	71946.7264	66818.15493
0.00001	119925.777	115788.6078

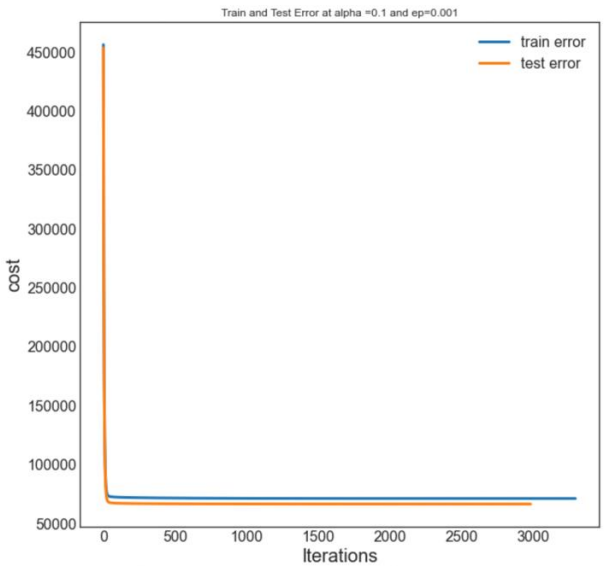
Points observed:

- **For the ideal learning rate, the algorithm converges fast to the minimal point.**
- **When the learning rate is shallow (~0.00001), the cost decreases nearly linear and takes more than 100000 iterations to converge.**
- **The best value of alpha for linear regression is found to be in the ranges of 0.001 and 0.1.**

2. Changing convergence threshold

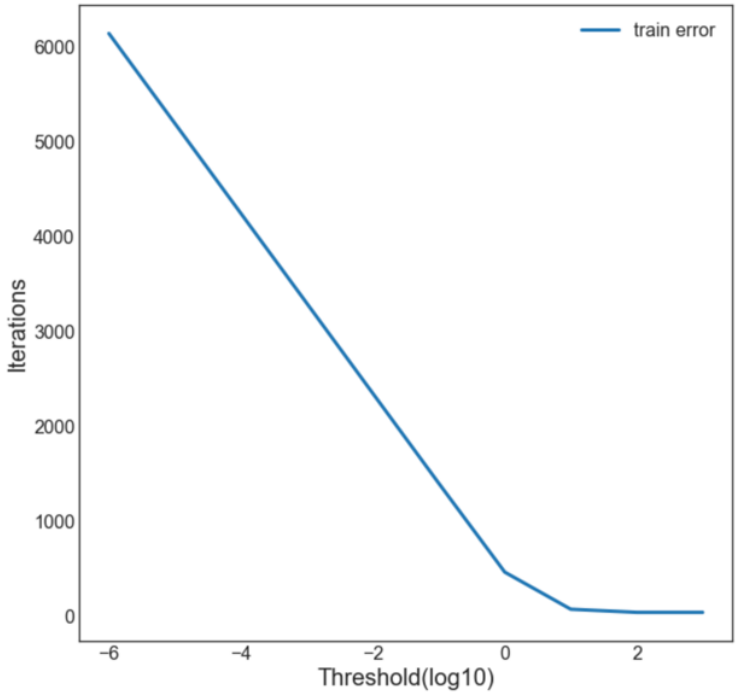
When the Change in Cost is within the Convergence threshold, the gradient descent algorithm is said to be converged. So, the algorithm converges faster at a higher threshold value but at a certain point increasing the threshold is found to increase the cost in both train and test datasets. We find the optimal value of the threshold by plotting cost as a function of the convergence threshold and find the point at which the test error is minimum.





Threshold	Train Error	Test Error
1000	77748.94536	72770.37201
100	73129.91114	67967.59627
10	72173.21276	67043.75288
1	71243.11578	66465.3668
0.1	70880.89217	66160.46621
0.01	70843.81495	66124.35525
0.001	70840.1044	66120.73972
0.000001	70839.6934	66120.33885

From the above, we can say that the ideal threshold value is 0.001 after that the error remains almost constant for both train and test.



We can see from the above graph that when we decrease the threshold it takes more iterations to reach convergence.

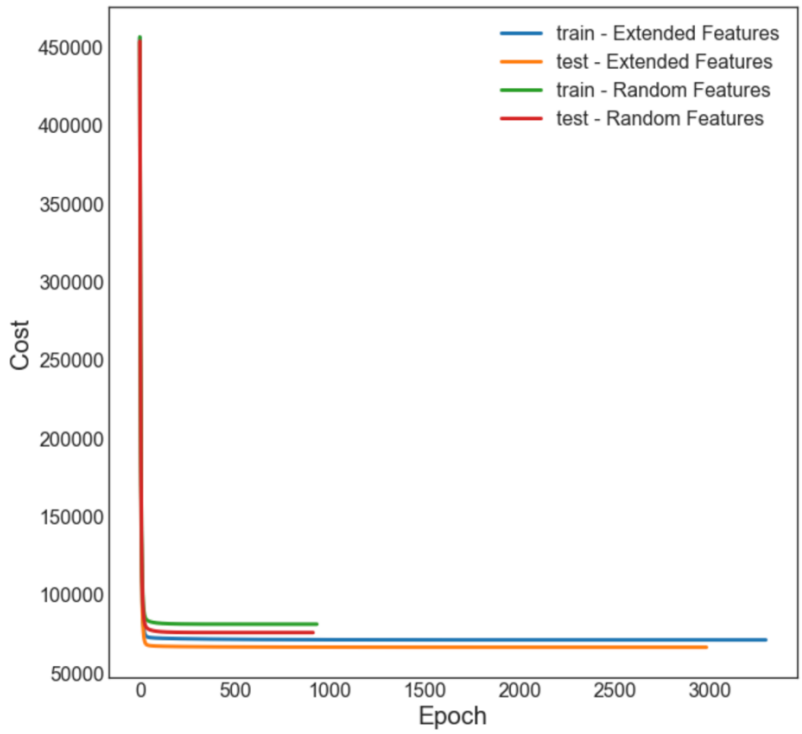
Points observed:

- **Tuning convergence threshold improves the accuracy of prediction and the convergence threshold of minimum error for both train and test datasets can be different.**
- **The cost and learning of the algorithm exponentially decrease with iteration and therefore trade of time and accuracy can be an important consideration.**

3. Random Feature Selection

In this experiment, 8 features are selected at random using the DataFrame.sample function, and the train and test errors are compared to the errors from the original set of features. The errors of random feature selection are found to be more than the respective errors from the original set of 14 features.

Rented Bike Count = B0+ B1* Dew point temperature(°C) + B2*Holiday
+ B3*Rainfall(mm) + B4*Snowfall (cm) + B5*Solar Radiation (MJ/m2) +
+ B6*Functioning Day + B7*Autumn + B8*Spring + B9*Summer
+ B10*Hour_0 + B11*Hour_1 + B12*Hour_2 + B13*Hour_3 + B14*Hour_4
+ B15*Hour_5 + B16*Hour_6 + B17*Hour_7 + B18*Hour_8 + B19*Hour_9
+ B20*Hour_10 + B21*Hour_11 + B22*Hour_12 + B23*Hour_13 + B24*Hour_14
+ B25*Hour_15 + B26*Hour_16 + B27*Hour_17 + B28*Hour_18 + B29*Hour_19
+ B30*Hour_20 + B31*Hour_21 + B32*Hour_22



	Extended	Random	Per Change
Train Error	70840.1	80893.4	14.191492
Test Error	66120.7	75521.8	14.2180609

The train and test error have been increased by 14.2% and 14.2% respectively. Here we can also observe that the percentage increase in both errors is equal when variables are sampled randomly.

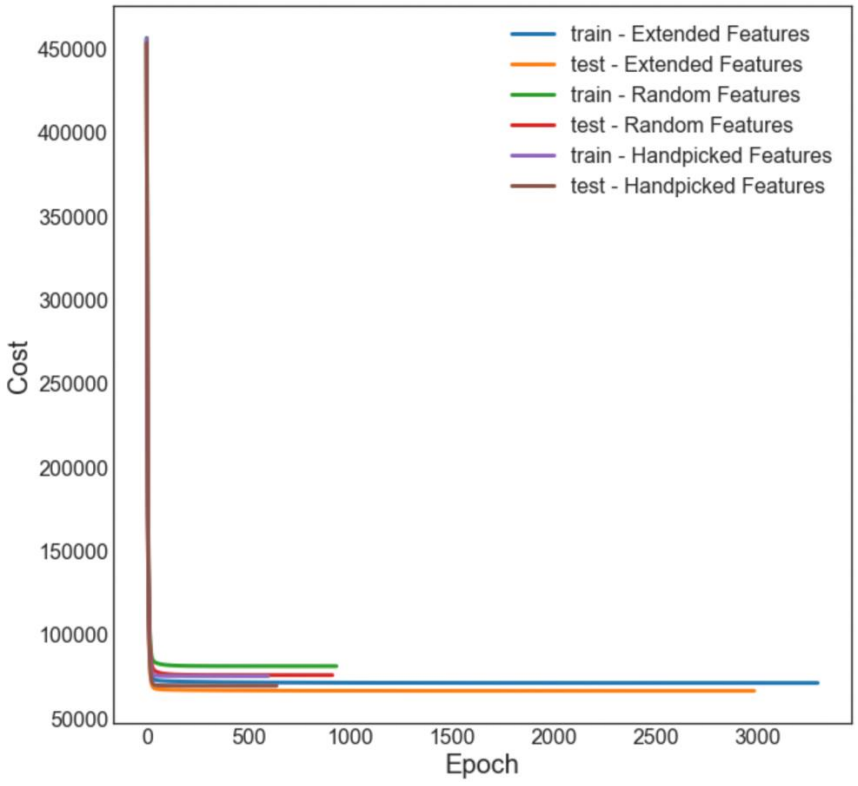
Points observed:

- **Random feature selection errors are mostly found to be more than intuition-based feature selection**
- **We need a clear knowledge of business understanding.**

4. Manual Feature Selection

In this experiment, 8 features are handpicked from the total set of 14 features.

Rented Bike Count = B0+ B1*Temperature(°C) + B2*Visibility (10m) + B3*Solar Radiation (MJ/m2) + B4*Rainfall(mm) + B5*Holiday + B6*Functioning Day
+ B7*Autumn + B8*Spring
+ B9*Summer + B10*Hour_0 + B11*Hour_1 + B12*Hour_2 + B13*Hour_3
+ B14*Hour_4 + B15*Hour_5 + B16*Hour_6 + B17*Hour_7 + B18*Hour_8
+ B19*Hour_9 + B20*Hour_10 + B21*Hour_11 + B22*Hour_12 + B23*Hour_13
+ B24*Hour_14 + B25*Hour_15 + B26*Hour_16 + B27*Hour_17 + B28*Hour_18
+ B29*Hour_19 + B30*Hour_20 + B31*Hour_21 + B32*Hour_22



Column1	Original	Random	Change1(%)	Hand-picked	Change2(%)
Train Dataset	70840.1	80893.4	14.191492	74956.5	5.81088976
Test Dataset	66120.7	75521.8	14.2180609	69121.2	4.53788143

The train and test errors are found to increase by 5.8% and 4.5% respectively from the original set.

Points Observed:

- **The randomly picked features performed worse compared to handpicked and original set of features because random selection doesn't incorporate the domain knowledge of the user.**
- **The handpicked features performed worse than the original features because we are restricting important features from the model. Restricting important features causes effect the performance of the algorithm.**
- **Using domain knowledge to understand which features are important for prediction to increase accuracy.**

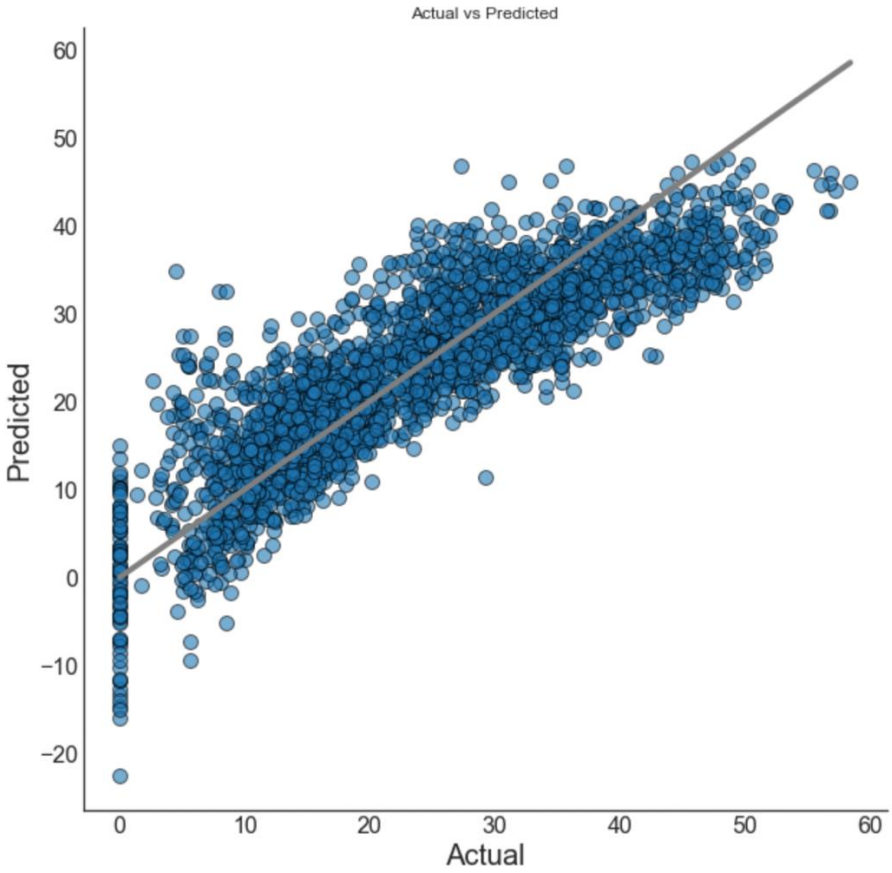
Results

Through experimentation, we found that **optimizing hyperparameters** like learning rate and convergence threshold are effective for better accuracy of the prediction. But some accuracy-improving measures like low alpha settings or very low convergence thresholds increase the time taken for fitting the ML algorithm with the training set. Therefore, machine learning involves balancing accuracy and computational limitations.

Also, we saw the importance of feature selection and engineering to improve the performance of the ML algorithm. The models incorporating some of domain knowledge will improves the acquired knowledge from data.

Features	Variable importance
Temperature	276.5991
Humidity	-141.95556
Wind speed	-4.90374
Visibility	3.59927
Solar Radiation	61.94704
Rainfall	-63.50185
Snowfall	9.04723
Holiday	-30.01851
Functioning Day	166.08951
Year	-156.67139
Month	-149.76465
Day	-4.87931
Autumn	332.16643
Spring	154.08936
Summer	212.1382
Hour_0	-20.61151
Hour_1	-42.98575
Hour_2	-64.00966
Hour_3	-82.91784
Hour_4	-88.22775
Hour_5	-89.57906
Hour_6	-55.77453
Hour_7	4.06589
Hour_8	75.17502
Hour_9	-15.75115
Hour_10	-63.93598
Hour_11	-61.24584
Hour_12	-60.47138
Hour_13	-59.59885
Hour_14	-57.24857
Hour_15	-41.79216
Hour_16	-13.987
Hour_17	35.89716
Hour_18	128.02666
Hour_19	86.29927
Hour_20	62.29645
Hour_21	66.98999
Hour_22	43.90886

From the variable importance table, we can infer that Temperature, Seasons, and Functioning Day are high importance features to predict the bike rental count.
We can try the functional form of Y to make errors linear, we applied square root transformation on Y and ran the algorithm.



From the above result, we observe the linearity in errors and the value has also been reduced where Train Error: 20.607462758329167

Test Error: 18.88241483364576.