Context:

In this project, we will be implementing linear and logistic regression on the SGEMM GPU kernel performance dataset to predict the average GPU runtime. Using gradient descent algorithm with batch update rule, we construct both linear and logistic regression models. We will perform experiments to find the best values for the parameters like learning rate and threshold, making sure that the models are efficient in the prediction of the target variable.

About the Data:

Dataset consists of 18 features and 241600 records, out of which 4 features represent the target variable. The dataset has no missing values and is confirmed after reviewing the data. Out of the 14 input features, 10 are ordinal and the rest being categorical with levels 0 and 1. All the features of type ordinal have values equal to power of 2.

Tasks 1,2:

Load the data set after downloading it from the UCI ML data repository. Rescale all the features to the range [0, 1] by subtracting each feature with its min and divide with the range of respective feature. Randomly split the dataset into train and test data using a split percentage of 0.7 for test.

Regression equation to find the average run time is:

$$AvgRunTime = \beta_0 + \beta_{MWG} * MWG + \beta_{NWG} * NWG + \beta_{KWG} * KWG + \beta_{MDMIC} * MDIMC + \beta_{NDIMC} * NDIMC + \beta_{MDIMA} * MDIMA + \beta_{NDIMB} * NDIMB + \beta_{KWI} * KWI + \beta_{VWM} * VWM + \beta_{VWN} * VWN + \beta_{STRM} * STRM + \beta_{STRM} * STRN + \beta_{SA} * SA + \beta_{SB} * SB$$

Task 3:

Initial parameter values used to implement gradient descent batch rule are:

β_0	β_{MWG}	β_{NWG}	β_{KWG}	β_{MDMIC}	β_{NDIMC}	β_{MDIMA}	β_{NDIMB}	β_{KWI}	β_{VWM}	β_{VWN}	β_{STRM}	β_{STRN}	β_{SA}	β_{SB}
0.21	0.25	0.19	0.20	0.22	0.24	0.26	0.18	0.27	0.28	0.29	0.31	0.18	0.32	0.20

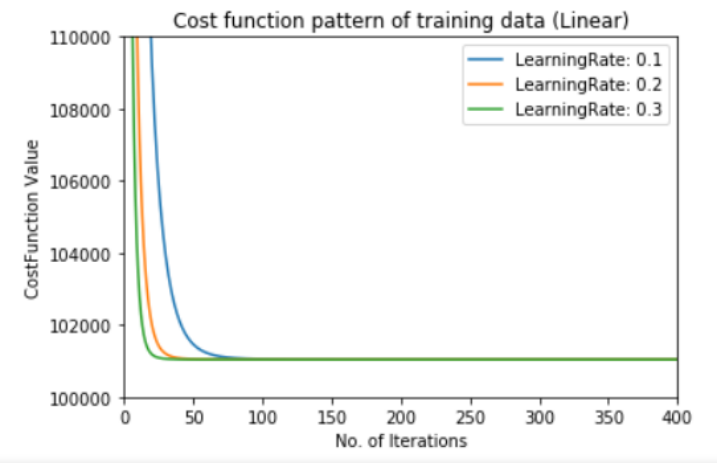
Errors with the given parameters are:
Squared error for train-data : 101041.6699
Squared error for test-data : 42103.212

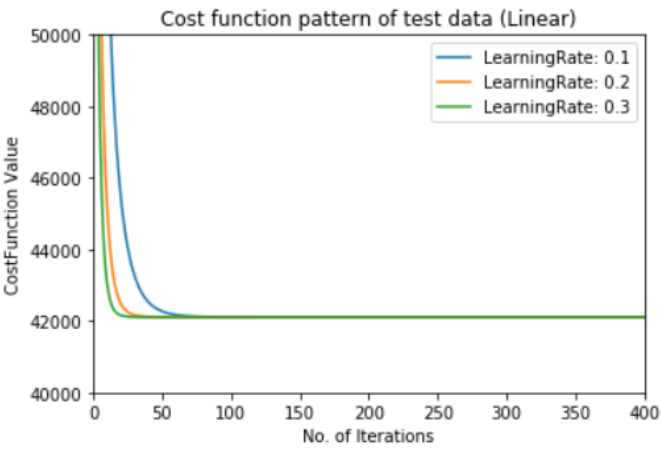
Experimentation 1:

In this experiment, we will be finding the best value of the learning rate for both linear and logit regression. The number of iterations is fixed at 700. Learning rate values that are used in this experiment are 0.10, 0.20, 0.30. After the experiment is conducted with linear regression, the number of iterations for the cost function, train error and test error are as below.

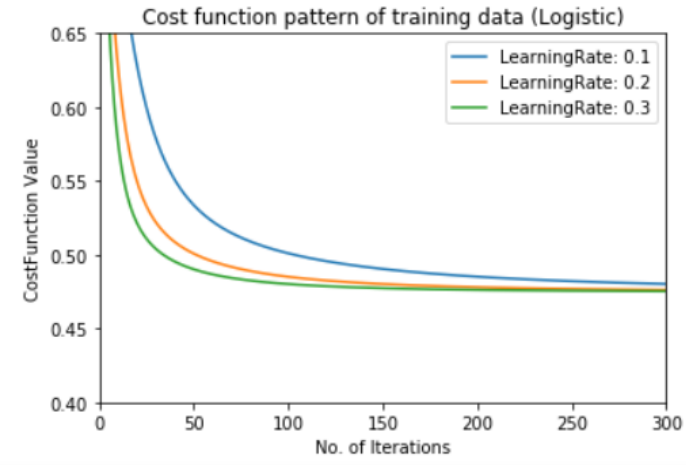
Learning Rate (Alpha)	0.10	0.20	0.30
Iterations (to converge)	350	189	131
Train Error	101041.6740	101041.6718	101041.6711
Test Error	42103.2168	42103.2147	42103.2141

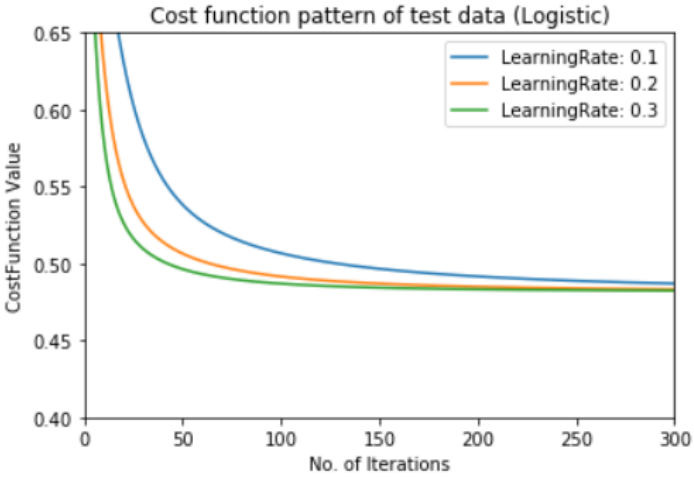
The variations in train and test error are plotted below:





From the graph, we can infer that the cost function of the gradient descent algorithm converges at different iterations levels for different alpha. With an increasing alpha value, it is evident that the cost function converges in lesser iterations. For instance, the learning rate 0.1 took 350 iterations to converge whereas with the value of 0.3 for learning rate the algorithm took 62% fewer iterations.
This pattern is similar in case of logistic regression as well. It's clear that the cost function takes fewer iterations to converge with increase in alpha value Variations in train and test error are plotted below.





After experimenting using logit regression, values of train and test errors with learning rate are as below:

Learning Rate	0.10	0.20	0.30
Iterations (to converge)	178	122	97
Train error	0.4870	0.4823	0.4804
Test error	0.4938	0.4892	0.4874

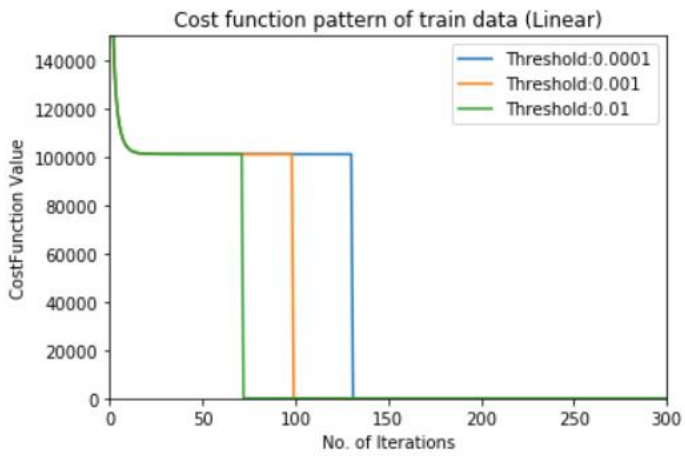
From the experiments conducted using linear and logistic regression, we can conclude that an alpha value of 0.30 is the best value to choose for this particular data set as the cost function converges with fewer iterations and have a lesser error term.

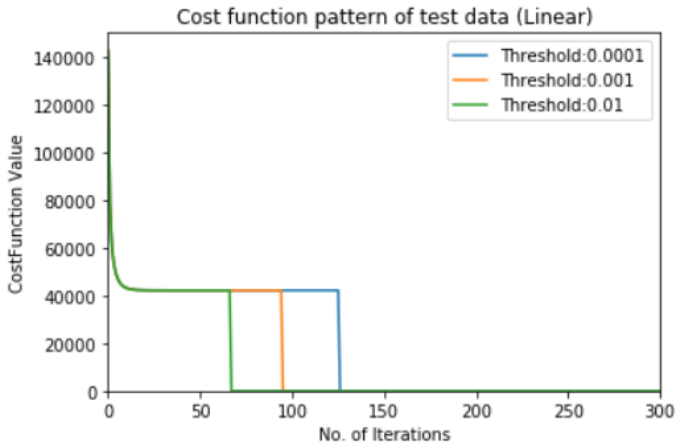
Experimentation 2:

As part of this experiment, we will be finding the best threshold value for which we attain the minimum cost function value. The threshold values chosen for this experiment are 0.0001, 0.001, 0.01 and the number of iterations is fixed at 500. Learning rate is fixed at 0.30. After the experiment is conducted with linear regression, the number of iterations for the cost function, train error and test error are as below.

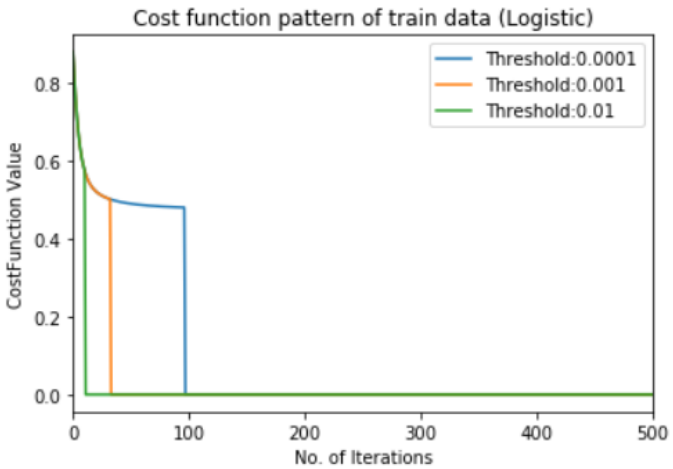
Threshold	0.0001	0.001	0.01
Train Error	101041.67117	101041.68352	101041.7828
Test Error	42103.2141	42103.2259	42103.3271
Iteration to converge	131	99	72

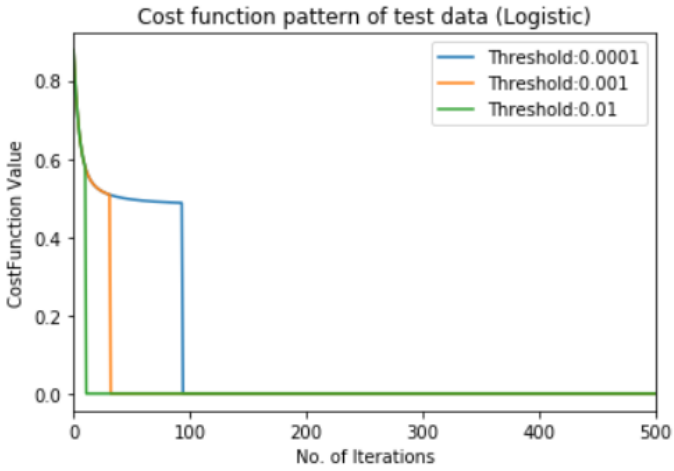
The variations in train and test error are plotted below:





Variation in train and test error for logistic regression are plotted below:



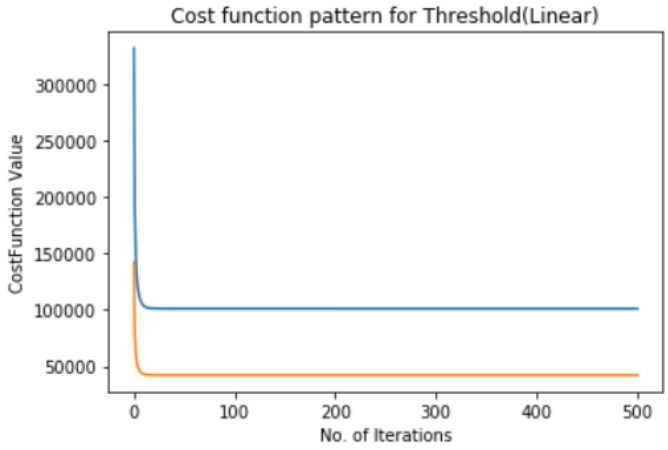


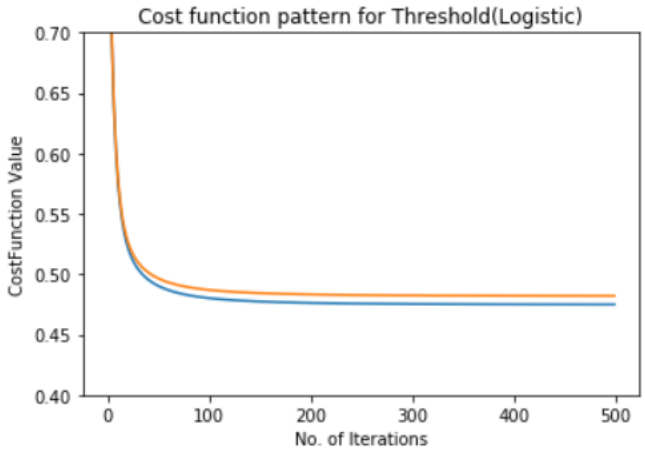
Threshold and cost function values for logistic regression are as below:

Threshold	0.0001	0.001	0.01
Train error	0.480438	0.50087	0.56601
Test error	0.48747	0.50750	0.56988
Iterations to converge	97	33	11

As seen from the plots, smaller threshold values take more iterations for the cost function to converge. A lower value of threshold will lead to a smaller cost function value. A threshold value has to be chosen which will give min value and also the number of iterations, the cost function will need to converge, is reasonable. In this case the best threshold value is 0.0001. Logistic regression also has similar relation between threshold, number of iterations and cost function value.

Test and train data error plot as a function of gradient descent with chosen learning rate and threshold values is as follows:





Linear regression Errors	
Train Error	101041.67
Test Error	42103.214

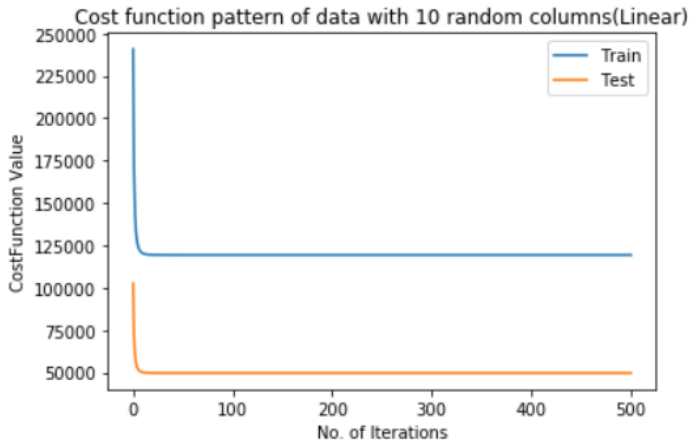
Logistic regression Errors	
Train Error	0.48043
Test Error	0.48747

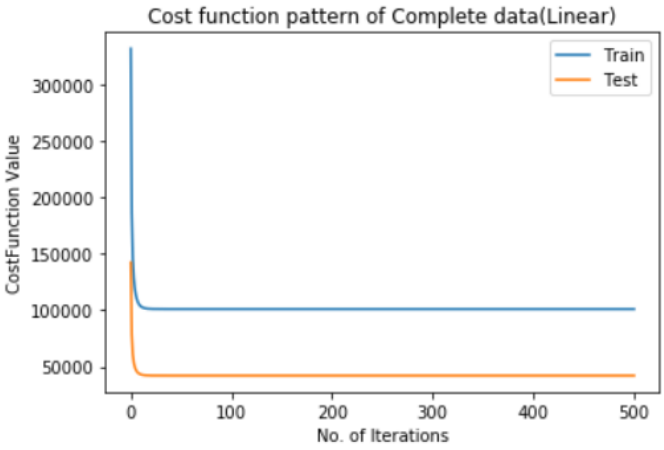
With the selected threshold value (0.0001), Train error and test error for linear regression are 101041.67 and 42103.214 respectively. In case of logistic regression, train error is 0.48043 and test error is 0.48747.

Experimentation 3:

In this experiment, we build linear and logistic models with 10 features selected using a random generator. Using random generator, we select 10 random numbers from a range of 14 numbers. Mapping the results with the columns gives us 10 features selected at random.

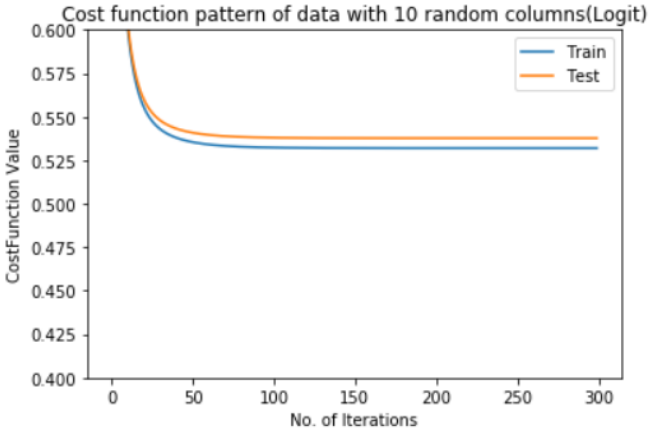
Train and test Errors of the linear regression for all features vs 10 random features are as below:

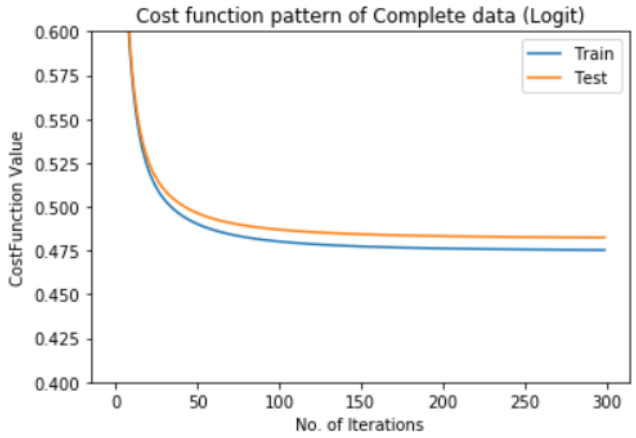




Model Type	Train Error	Test error
10 random features	119329.6469	49642.827
Complete Data	101041.6711	42103.21
	18287.9758	**7539.617**

Train and test Errors of the logistic regression for all features vs 10 random features are as below:



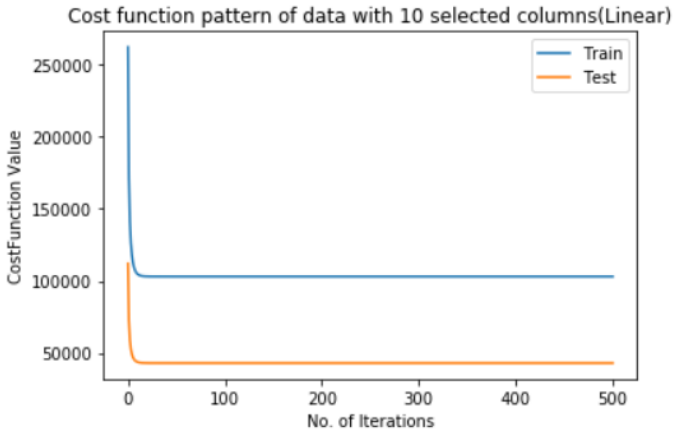


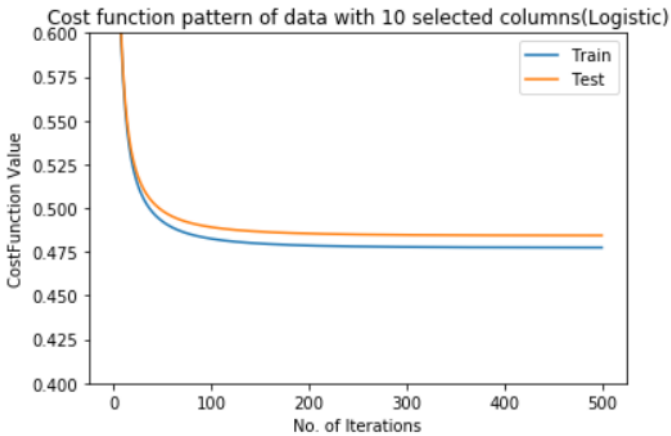
Model Type	Train Error	Test error
10 random features	0.53384	0.53956
Complete Data	0.48286	0.48966
	0.05098	**0.04990**

From the above results, it is clear that the linear and logistic models built with 10 random features significantly have more train and test error when compared to the models built using all features.
10 features used in this experiment are **MWG, NWG, MDIMC, MDIMA, NDIMB, KWI, VWM, VWN, STRN, SB.**

Experimentation 4:

As part of this experiment we find the 10 best features to build linear and logistic models which will be efficient in finding the target variable. To find the best and relevant features, I am considering the correlation between the target variable and Input Features. Train and test error pattern and details of the models built using the selected 10 features are as below.





Summary of Linear model	
Features	10
Train Error	103048.601
Test Error	42963.71

Summary of Logistic model	
Features	10
Train Error	0.48286
Test Error	0.48966

Eliminating the features KWG, MDIMA, NDIMB, STRN will give us the best 10 features to build the regression models because the features which I mentioned have very weak correlation with the target variable and hence will not contribute much in estimating the average runtime(target).

Train and test error values of this model are less by a significant margin when compared to the model built using 10 random features. Choosing features like MDIMA, NDIMB, STRN over other features, on which the target variable depends more, will degrade the prediction of the average runtime as features, with good correlation, contribute better in predicting the target variable when compared to the one chosen.

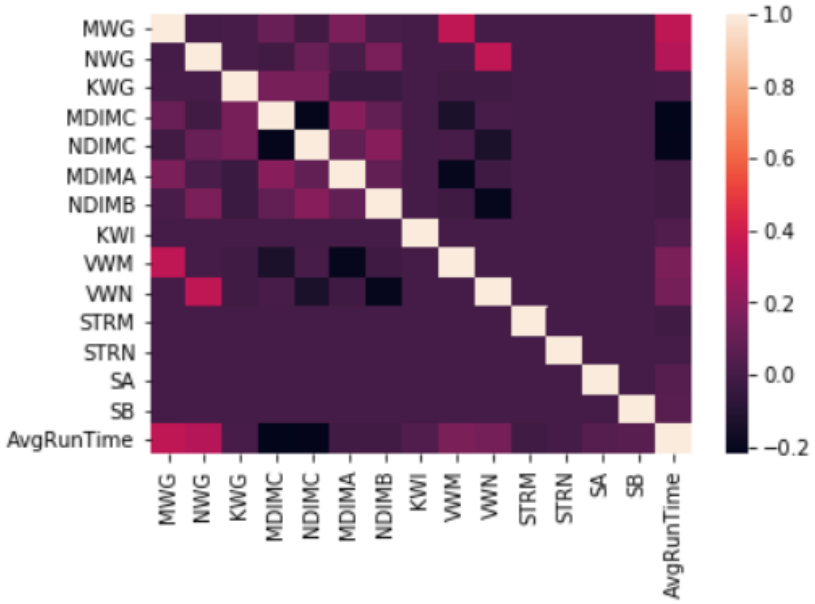
In other scenario, when we compare the errors from the model built using 10 selected features and all input features there is a small margin of difference in the values. Even though we exclude the columns with a weak correlation in Experiment 4, as the variables excluded are not independent of the target variable, we miss on the minor dependency that the features provide enabling extra error while prediction. Hence the error term in the model with all features is less compared to the model built with 10 best features.

Discussion:

From the experiments, the conclusion is that a learning rate of 0.30 and a threshold of 0.001 works well for the data set. In the process of finding the optimum values, it was evident that changing the parameter such as learning rate and threshold makes the algorithm more efficient in terms of time taken or finding a good cost function value. We trained the algorithm with the random train split and then applied it on the test data. Initially, we just chose one kind of split which will not cover all combinations of data on which we are supposed to build the model. So, one of those things to improve the model is by building the model using different combinations of train and test data.

During the process, we did find the learning rate and threshold values that gave us the best results. Analyzing the results and how the iterations, the model took, varied with different data inputs and parameters. The time taken by the models to find the optimum cost function value can be reduced further by finding more precise values of parameters like learning rate and threshold, which would work well for all cases. As part of the algorithm implementation, we were looking for the convergence of cost function value and achieving the min value at the earliest. When we think of the cost function value, it is significantly large and models with such values are not reliable in practice. We might need to work on the relation between the data and train the model so that the magnitude of cost function value comes down.

The selection of input features will affect the efficiency of the model significantly. For instance,s during experiment 3, when we chose random features and built the model, we ended with cost function values which are relatively higher when compared with the cost function values obtained from the model that used all features. So, sometimes, it is suggested to use all features instead of choosing the wrong ones. Now consider the correlation between the target variable and the input features.



From Experiment 3 and 4, it was clear that choosing the features affects the model on a significant note. When we chose 10 features in experiment 4 based on the correlation between the target and the input variables, the results got better rather than choosing 10 random features. Similarly we might get better results when we use other feature selection techniques like univariate selection which might give more relevant metrics that helps in choosing features that builds the best model.