ASSIGNMENT 1 REPORT – MANIKANDAN LALITPRASAD (mxl190001)

Dataset- SGEMM GPU Kernel Performance

This data set measures the running time of different parameter combinations of parameterizable SGEMM GPU kernel. The dataset contains 241600 records of these feasible combinations. For each tested combination, 4 runs were performed and recorded (measured in milliseconds.

There are 14 parameters, the first 10 are ordinal and can only take up to 4 different powers of two values, and the 4 last variables are binary.

Attribute Information:

Independent variables:
1-2. MWG, NWG: per-matrix 2D tiling at workgroup level: {16, 32, 64, 128} (integer)
3. KWG: inner dimension of 2D tiling at workgroup level: {16, 32} (integer)
4-5. MDIMC, NDIMC: local workgroup size: {8, 16, 32} (integer)
6-7. MDIMA, NDIMB: local memory shape: {8, 16, 32} (integer)
8. KWI: kernel loop unrolling factor: {2, 8} (integer)
9-10. VWM, VWN: per-matrix vector widths for loading and storing: {1, 2, 4, 8} (integer)
11-12. STRM, STRN: enable stride for accessing off-chip memory within a single thread: {0, 1} (categorical)
13-14. SA, SB: per-matrix manual caching of the 2D workgroup tile: {0, 1} (categorical)

Output:
15-18. Run1, Run2, Run3, Run4: performance times in milliseconds for 4 independent runs using the same parameters. They range between 13.25 and 3397.08.

We take the average of these four runs as our target variable and added as a column in our data frame

OBJECTIVE-

Our Goal is to predict the run time GPU Kernel by generating Linear and Logistic Regression models by implementing the Gradient Descent Algorithm

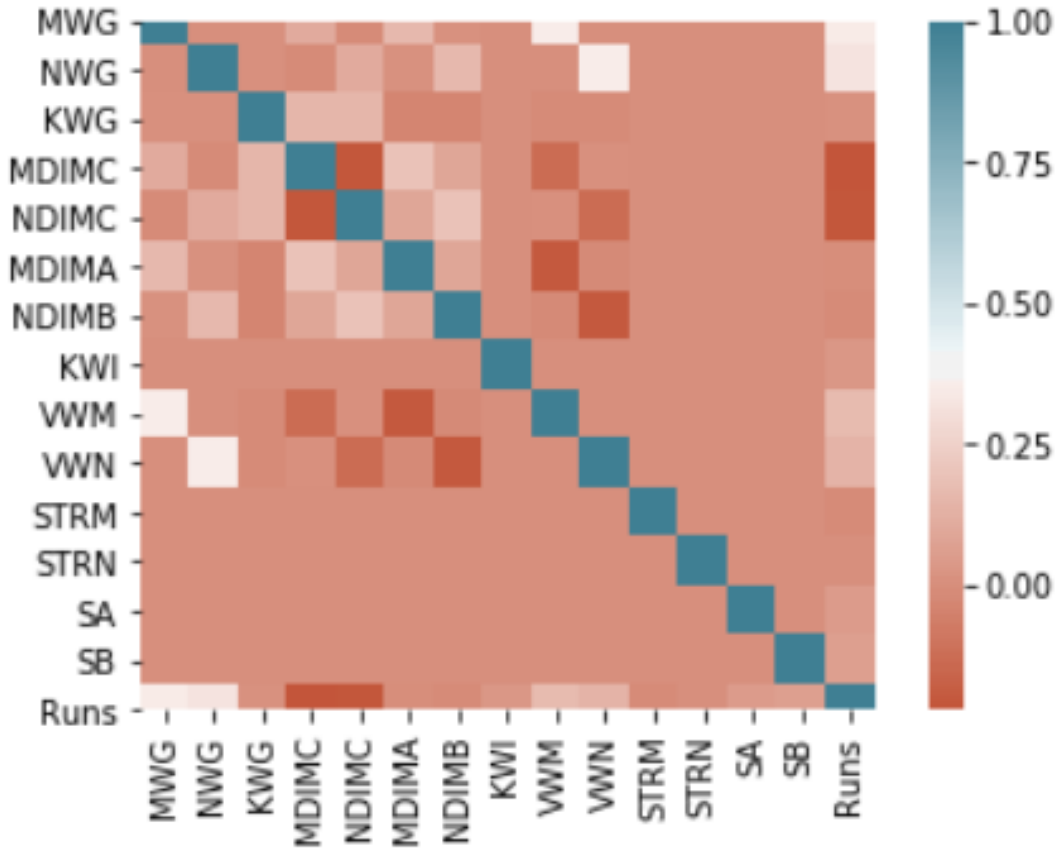
Target Variable: Runs

Correlation Plot:

	MWG	NWG	KWG	MDIMC	NDIMC	MDIMA	NDIMB	KWI	VWM	VWN	STRM	STRN	SA	SB	Runs
MWG	1	0.0006	0.009296	0.105791	-0.00859	0.158772	0.014898	1.05E-19	0.353763	-0.00084	-7.35E-21	0	0	0	0.35181
NWG	0.0006	1	0.009296	-0.00859	0.105791	0.014898	0.158772	2.94E-20	-0.00084	0.353763	1.84E-21	1.84E-21	0	0	0.32046
KWG	0.009296	0.009296	1	0.148348	0.148348	-0.03456	-0.03456	-6.56E-19	-0.01199	-0.01199	6.43E-21	2.76E-21	0	0	0.01123
MDIMC	0.105791	-0.00859	0.148348	1	-0.20956	0.197433	0.084606	2.54E-18	-0.13391	0.010531	-3.49E-19	-3.68E-21	0	0	-0.2211
NDIMC	-0.00859	0.105791	0.148348	-0.20956	1	0.084606	0.197433	-7.35E-20	0.010531	-0.13391	1.10E-20	3.68E-21	0	0	-0.21459
MDIMA	0.158772	0.014898	-0.03456	0.197433	0.084606	1	0.088096	-1.08E-18	-0.20271	-0.01903	5.51E-21	1.84E-21	0	0	-0.00704
NDIMB	0.014898	0.158772	-0.03456	0.084606	0.197433	0.088096	1	3.10E-18	-0.01903	-0.20271	-1.84E-21	-1.84E-21	0	0	-0.00871
KWI	1.05E-19	2.94E-20	-6.56E-19	2.54E-18	-7.35E-20	-1.08E-18	3.10E-18	1	9.41E-17	-1.44E-17	0	0	0	0	0.032571
VWM	0.353763	-0.00084	-0.01199	-0.13391	0.010531	-0.20271	-0.01903	9.41E-17	1	0.001165	-1.19E-20	-4.60E-21	0	0	0.164273
VWN	-0.00084	0.353763	-0.01199	0.010531	-0.13391	-0.01903	-0.20271	-1.44E-17	0.001165	1	3.68E-21	3.68E-21	0	0	0.144745
STRM	-7.35E-21	1.84E-21	6.43E-21	-3.49E-19	1.10E-20	5.51E-21	-1.84E-21	0	-1.19E-20	3.68E-21	1	0	0	0	-0.01259
STRN	0	1.84E-21	2.76E-21	-3.68E-21	3.68E-21	1.84E-21	-1.84E-21	0	-4.60E-21	3.68E-21	0	1	0	0	-0.00011
SA	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0.051975
SB	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.063963
Runs	0.35181	0.32046	0.01123	-0.2211	-0.21459	-0.00704	-0.00871	0.032571	0.164273	0.144745	-0.01259	-0.00011	0.051975	0.063963	1

We can see the above plot and the correlation between the variables.

Heat Map-



TASKS PERFORMED:

PART 1:

The dataset has been partitioned into train and test sets in the ratio 70:30 which is the ideal ratio to divide the dataset.

PART 2:

For the regression model, I have chosen 14 variables after analyzing the correlation and choosing the ones that are at least weakly correlated with the target variable. Below is the regression model which I have used in the model:

The Regression Equation is given as-

Runs = β0+ β1 *MWG +β2*NWG+ β3*KWG+ β4*MDIMC+ β5*NDIMC_2+ β6*MDIMA+ β7*NDIMB+ β8*KWI+ β9*VWM+ β10*VWN+ β11*STRM+β12*STRN + β13*SA+ β14*SB

PART 3: The initial parameter values for the model are as follows:

Bo	0.21
B1	0.25
B2	0.19
B3	0.20
B4	0.22
B5	0.24
B6	0.26
B7	0.18
B8	0.27
B9	0.28
B10	0.29
B11	0.31
B12	0.18
B13	0.32
B14	0.20

Training Set Error: `101041.66997619592`
Testing Set Error: `42103.21286365832`

The linear regression was implemented with batch update rule as follows:

$j = 0, 1, \ldots, n$

Simultaneous update

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

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

The Gradient Descent Cost Function is defined as:

$$J(\beta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

PART 4:

The problem is converted into a binary classification problem. The Logistic Regression function is implemented to carry out classification on this dataset.

EXPERIMENTATION 1:

For this experiment, the maximum number of iterations for gradient descent is fixed at 700 iterations. The learning rate experimented with are 0.1,0.2 and 0.3.

For Linear Regression:

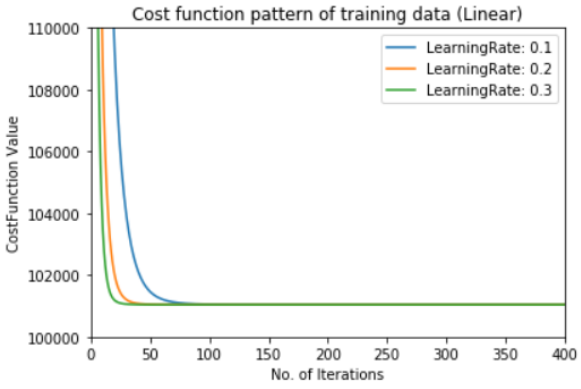
The variations of train and test error for different values of alpha (learning rate) is plotted below:

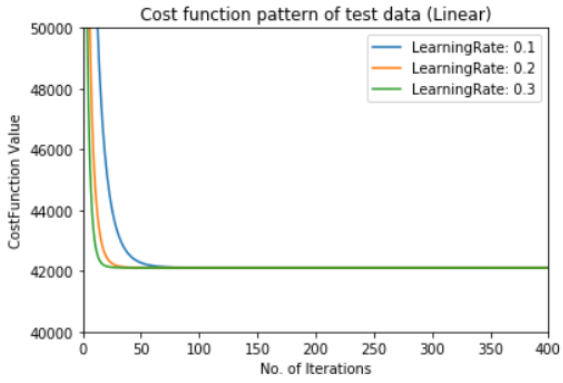
Learning Rate	0.1	0.2	0.3
Iterations to converge	350	189	131
Train Error	101041.674027198	101041.6718682865	101041.67117193801
Iterations to converge	337	182	126
Test Error	42103.2168500458	42103.21477147242	42103.21410259092

From the table, we can see that the training error and testing error is the lowest for learning rate (a=0.3). Therefore, we fix a=0.3 as our best learning rate for this model.

As evident from the train and test plots, we can see that the number of iterations taken by the gradient descent algorithm to converge is less for higher learning rates. For smaller learning rate, the number of iterations required is more.

The training and test errors at which at curve reaches a minimum for each learning rate is given below-





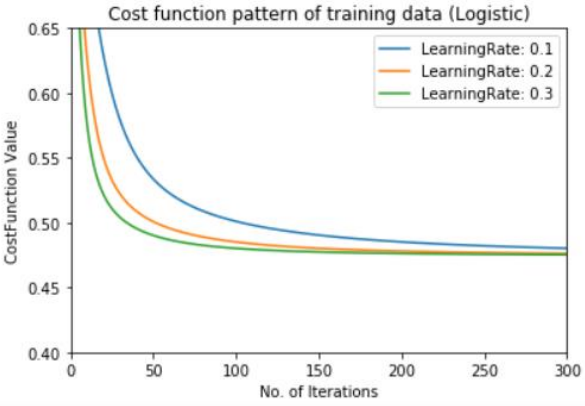
For Logistic Regression-

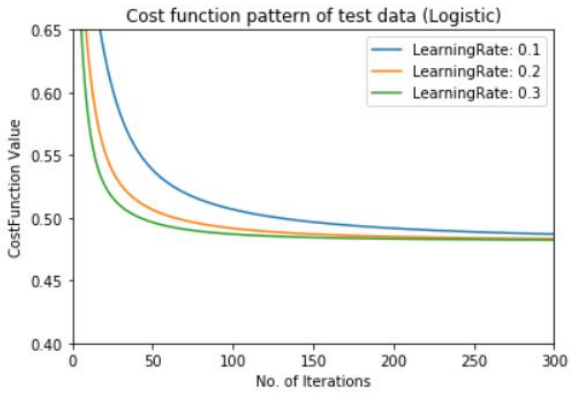
From the table, we can see that the training error and testing error is the lowest for learning rate (a=0.3). Therefore, we fix a=0.3 as our best learning rate for this model.

As evident from the train and test plots, we can see that the number of iterations taken by the gradient descent algorithm to converge is less for higher learning rates. For smaller learning rate, the number of iterations required is more.

Learning Rate	0.1	0.2	0.3
Iterations to converge	178	122	97
Train Error	0.4870705320680	0.48235939471321	0.480438743709100
Iterations to converge	174	119	94
Test Error	0.49382461253189	0.487477887668888	0.487477887668888

The training and test errors at which at curve reaches a minimum for each learning rate is given below-





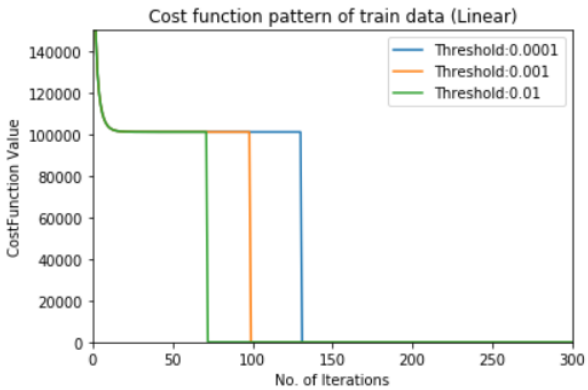
EXPERIMENTATION 2:

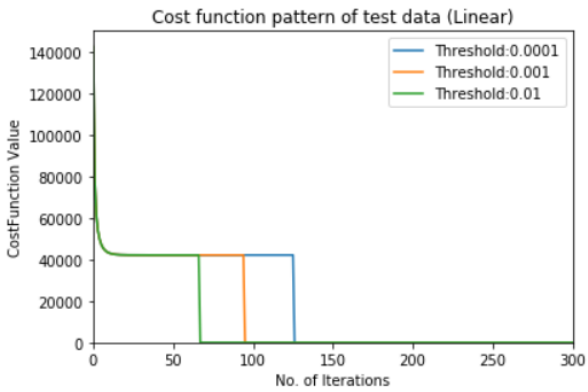
The threshold values chosen for this experimentation are 0.0001, 0.001, 0.01 at the fixed learning rate of alpha 0.3.

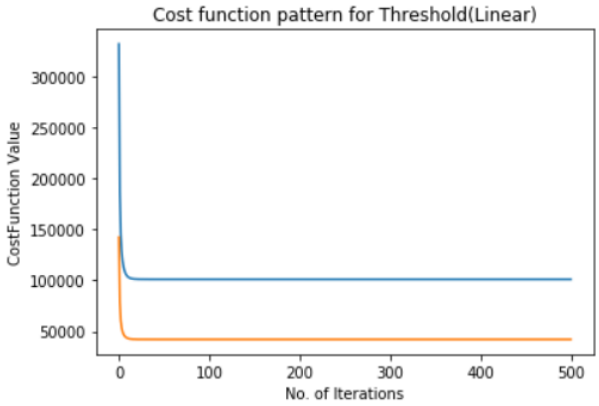
Linear Regression-

The plot of train and test error as function of threshold are given at the bottom. As seen from the plot of training cost and test cost, the lower value of threshold, the error seems to be minimum and maximum when the threshold value is higher. The best threshold for this dataset seems to be 0.0001

Threshold Value	0.0001	0.001	0.01
Iterations to converge	131	99	72
Training Cost Error	101041.671171938	101041.6835289406	101041.78288952922
Iterations to converge	126	95	67
Test Error	42103.2141025909	42103.22594307602	42103.327113944426







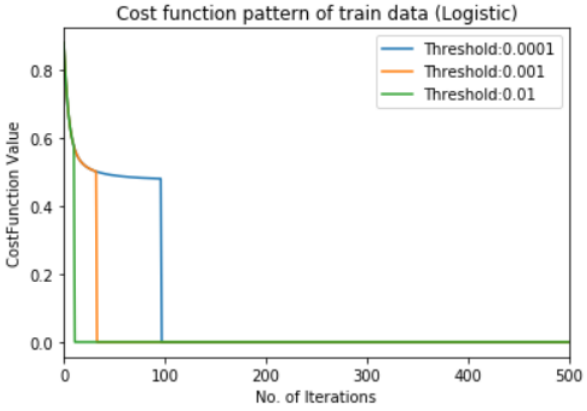
In the plot, the best threshold value of 0.0001 is chosen and train and test error is plotted (in one figure) as a function of number of gradient descent iterations.

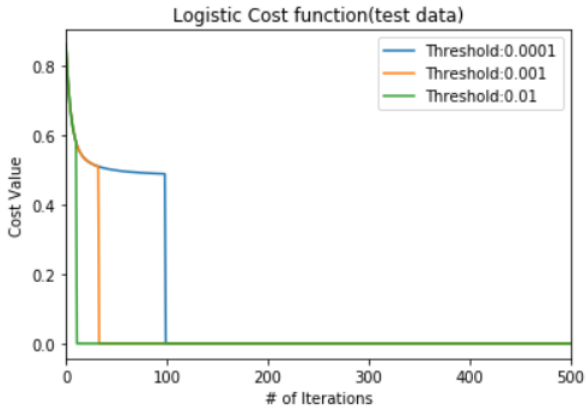
Here, the blue line is for the training set and the orange line is for the test set.

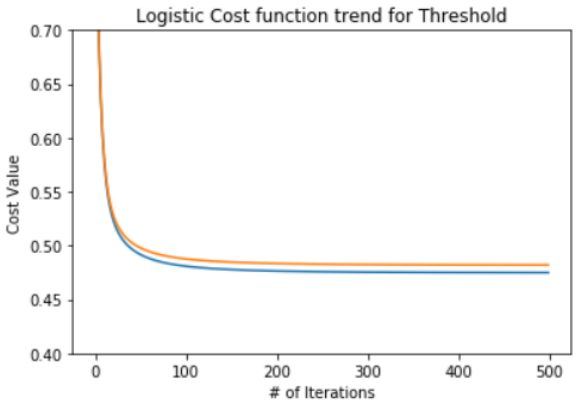
Logistic Regression-

The plot of train and test error as function of threshold are given at the bottom. As seen from the plot of training cost and test cost, the lower value of threshold, the error seems to be minimum. The best threshold for this dataset seems to be 0.0001

Threshold Value	0.0001	0.001	0.01
Iterations to converge	97	33	11
Training Cost Error	0.48043874370910	0.500870921762877	0.566012554709457
Iterations to converge	99	33	11
Test Error	0.487698440627565	0.508211753884678	0.570241455443700







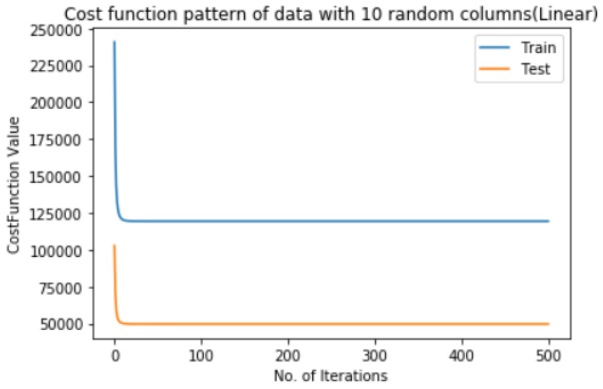
In the plot, the best threshold value of 0.0001 is chosen and train and test error is plotted (in one figure) as a function of number of gradient descent iterations.

Here, the blue line is for the training set and the orange line is for the test set.

EXPERIMENT 3:

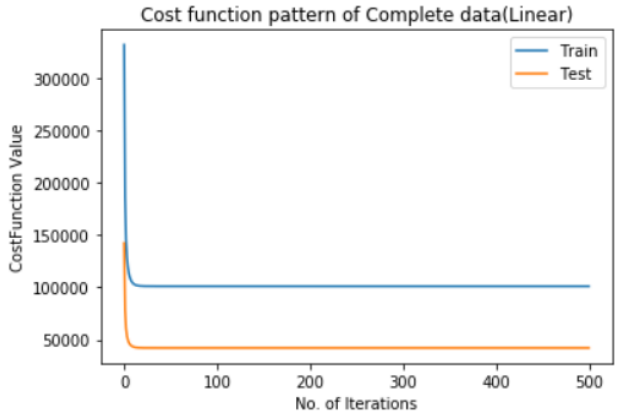
Using a random number generator to pick 10 random features ['MWG', 'NWG', 'MDIMC', 'MDIMA', 'NDIMB', 'KWI', 'VWM', 'VWN', 'STRN', 'SB'].

Linear Regression-



The comparison between the train and the test error between model containing 10 random features and the model containing all the features is given below:

Model Type	10 Feature Model	Full Feature Model
Iterations to converge	131	118
Training Cost Error	101041.67117193801	119329.64694100169
Iterations to converge	126	114
Test Error	42103.21410259092	49642.82734612157

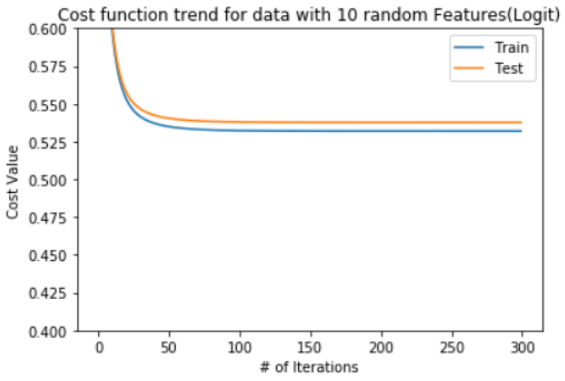


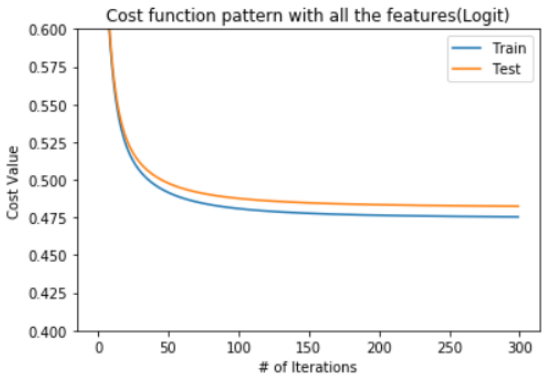
The 10 randomly selected features seems to have lower training and test cost error than Full feature model

Logistic Regression-

Model Type	10 Feature Model	Full Feature Model
Iterations to converge	58	101
Training Cost Error	0.533950231837862	0.48074843818522756
Iterations to converge	57	99
Test Error	0.539571828993999	0.48769844062756523

The comparison between the train and the test error between model containing 10 random features and the model containing all the features is given



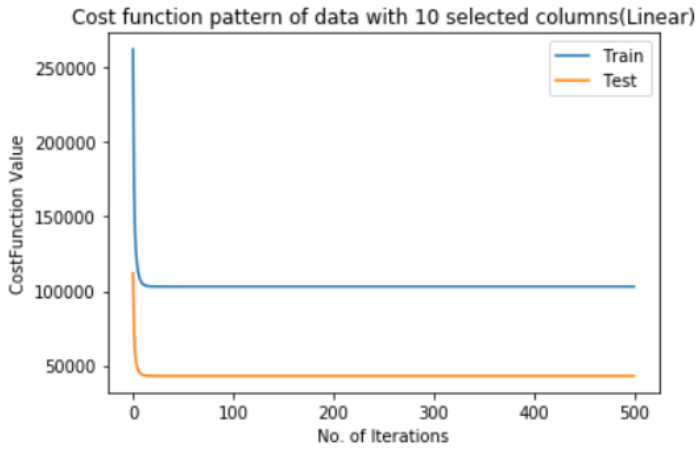


The 10 randomly selected features seems to have higher training cost error and test cost error than Full feature model

EXPERIMENT 4

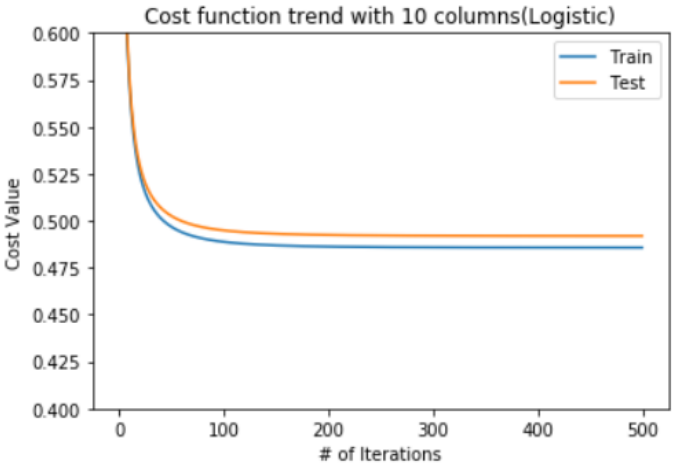
In this experiment, we have excluded 4 features – KWG, MDIMA,NDIMB, STRM and retrained the models using these ten features – MWG, NWG, MDIMC, NDIMC, KWI, VWM, VWN, STRN, SA, SB to be best suited to predict the output.

Linear Regression-



On comparison with the original set of features and to random features, these 10 features selected has a lower train error of 103048.60124190032 and test error of 42963.71300305453 to that of full feature model, but has a higher error rate on comparison with the random feature model

Logistic Regression-



On comparison with the original set of features and to random features, these 10 features selected has a higher train error of o.4897832784909297 and test error of 0.49604377631966856 to that of full feature model, but has a lower error rate on comparison with the random feature model

Now the training and test error are certainly lower than that of the randomly selected features, but still is not as good as the original model with 14 features. It performs better than the random model because in this model, the variables are carefully selected with common sense knowledge and hence the error is a bit lower. But this model has a higher error rate than the original model because the original model has a lot of features (14) and hence can explain better. Also, the correlation plays a factor in the first model since it had a lot of correlated variables. I didn't account for correlation in this experiment and hence might have resulted in a higher error. Also, a nonlinear model with polynomial features might have given a lower error rate.

INTERPRETATION AND SUMMARY-

- We can clearly see that learning rate is very critical in this model as a lower learning rate can result in the number of iterations to increase to a large extent. Hence finding the correct learning rate and threshold value can work a long way in reducing the training and test error rate as much as possible.
- Also as seen from the experiment 4, choosing the most relevant variables is the most important aspect of a model. Feature selection must be done so that it reduces the bias – variance tradeoff and does not cause underfitting or overfitting. The model also must not be made complex by adding too many features.
- In conclusion, I would say that choosing the correct learning rate and feature selection are the most important aspects of prediction.

OTHER STEPS WHICH CAN BE TAKEN TO IMPROVE THE MODEL-

- Cross validation can be performed to further fine tune the model before testing the model with the test data so that the mean squared error can be decreased further.
- For feature selection, various techniques such as forward, backward and hybrid methods can be used to select the most important parameters for the model.
- Also, the outliers have to be removed so that the variables are not skewed and biased.
- If we feel that a linear model is not sufficient, we can move to a nonlinear model such as a polynomial model or a log model to increase the performance of the model. Sometimes these models can help reduce overfitting and underfitting.
