Prepared by: Eeshan Phatak **Date: 02/13/2020**

BUAN 6341
APPLIED MACHINE LEARNING
ASSIGNMENT 1
SGEMM GPU KERNEL PERFORMANCE DATASET

Report Summary

- The report begins with description about the dataset used for the project and the algorithms used for implementation of the project
- Later in this report we will be talking about the steps of Project Execution which includes Data Preparation, Algorithm Implementation and Experimentation using different variables

Dataset Information

- The Dataset used for this assignment is the SGEMM GPU kernel performance dataset which is available on the UCI Machine Learning Repository.
- The dataset consists of 18 columns and 241600 rows. The last 4 columns of the dataset are the 4 tested run times of the GPU.
- The Dependent Variable is obtained by calculating the mean of these 4 columns and storing into a new column 'Run_Avg'

Project Execution Steps

Data Preparation

1. **Linear Regression**

- The values of the Dependent Variable (Run_Avg) are skewed are so we have taken the log of the dependent variable so that the skewed values would not affect the predictions.
- We then regularize the whole dataset so that all the values are in the range of -1 to 1
- Now the data is ready to use for Linear Regression

MWG	NWG	KWG	MDIMC	NDIMC	MDIMA	NDIMB	KWI	VWM	VWN	STRM	STRN	SA	SB	Run_Av
516754	-1.516754	-1.210995	-0.753892	-0.753892	-0.998052	-0.998052	-0.999998	-0.741447	-0.741447	-0.999998	-0.999998	-0.999998	-0.999998	0.11847
516754	-1.516754	-1.210995	-0.753892	-0.753892	-0.998052	-0.998052	-0.999998	-0.741447	-0.741447	-0.999998	-0.999998	-0.999998	0.999998	-0.22805
516754	-1.516754	-1.210995	-0.753892	-0.753892	-0.998052	-0.998052	-0.999998	-0.741447	-0.741447	-0.999998	-0.999998	0.999998	-0.999998	-0.20735
516754	-1.516754	-1.210995	-0.753892	-0.753892	-0.998052	-0.998052	-0.999998	-0.741447	-0.741447	-0.999998	-0.999998	0.999998	0.999998	-0.14296
516754	-1.516754	-1.210995	-0.753892	-0.753892	-0.998052	-0.998052	-0.999998	-0.741447	-0.741447	-0.999998	0.999998	-0.999998	-0.999998	0.13576
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
120450	1.120450	0.825764	2.294244	2.294244	1.558017	1.558017	0.999998	0.794054	0.794054	0.999998	-0.999998	0.999998	0.999998	-1.54438
120450	1.120450	0.825764	2.294244	2.294244	1.558017	1.558017	0.999998	0.794054	0.794054	0.999998	0.999998	-0.999998	-0.999998	-0.92029
120450	1.120450	0.825764	2.294244	2.294244	1.558017	1.558017	0.999998	0.794054	0.794054	0.999998	0.999998	-0.999998	0.999998	-0.94207
120450	1.120450	0.825764	2.294244	2.294244	1.558017	1.558017	0.999998	0.794054	0.794054	0.999998	0.999998	0.999998	-0.999998	-1.12963
120450	1.120450	0.825764	2.294244	2.294244	1.558017	1.558017	0.999998	0.794054	0.794054	0.999998	0.999998	0.999998	0.999998	-1.54438

s × 15 columns

2. Logistic Regression

- For Logistic Regression we need to classify the Dependent Variable values into 0 and 1.
- We take the median of the dependent variable and classify all values equal or above median as 1 and below median as 0.
- Then the Independent Variables are regularized, and the data is ready to use for Logistic Regression

MWG	NWG	KWG	MDIMC	NDIMC	MDIMA	NDIMB	KWI	VWM	VWN	STRM	STRN	SA	SB	Run_Avg
3754	-1.516754	-1.210995	-0.753892	-0.753892	-0.998052	-0.998052	-0.999998	-0.741447	-0.741447	-0.999998	-0.999998	-0.999998	-0.999998	1
3754	-1.516754	-1.210995	-0.753892	-0.753892	-0.998052	-0.998052	-0.999998	-0.741447	-0.741447	-0.999998	-0.999998	-0.999998	0.999998	1
3754	-1.516754	-1.210995	-0.753892	-0.753892	-0.998052	-0.998052	-0.999998	-0.741447	-0.741447	-0.999998	-0.999998	0.999998	-0.999998	1
3754	-1.516754	-1.210995	-0.753892	-0.753892	-0.998052	-0.998052	-0.999998	-0.741447	-0.741447	-0.999998	-0.999998	0.999998	0.999998	1
3754	-1.516754	-1.210995	-0.753892	-0.753892	-0.998052	-0.998052	-0.999998	-0.741447	-0.741447	-0.999998	0.999998	-0.999998	-0.999998	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0450	1.120450	0.825764	2.294244	2.294244	1.558017	1.558017	0.999998	0.794054	0.794054	0.999998	-0.999998	0.999998	0.999998	0
0450	1.120450	0.825764	2.294244	2.294244	1.558017	1.558017	0.999998	0.794054	0.794054	0.999998	0.999998	-0.999998	-0.999998	0
0450	1.120450	0.825764	2.294244	2.294244	1.558017	1.558017	0.999998	0.794054	0.794054	0.999998	0.999998	-0.999998	0.999998	0
0450	1.120450	0.825764	2.294244	2.294244	1.558017	1.558017	0.999998	0.794054	0.794054	0.999998	0.999998	0.999998	-0.999998	0
0450	1.120450	0.825764	2.294244	2.294244	1.558017	1.558017	0.999998	0.794054	0.794054	0.999998	0.999998	0.999998	0.999998	0

: 15 columns

Algorithm Implementation

- The Batch Gradient Descent algorithm is implemented using Python numerical computation packages numPy and pandas. We have also used the matplotlib package in python to plot multiple graph plots which help us visually understand the results obtained.
- This algorithm is applied on two machine learning techniques Linear Regression and Logistic Regression. The algorithm is implemented with options to change Hyperparameters like Learning Rate, Convergence Threshold, Number of Features and Maximum Iterations.
- The formulas provided during class sessions have been used to calculate the cost value and several other values required for the implementation of the algorithm.
- Linear Regression Model Equation

MWG	NWG	KWG	MDIMC	NDIMC	MDIMA	NDIMB
0.498881	0.396446	0.085242	-0.395638	-0.38185	0.004815	-0.001653
KWI	VWM	VWN	STRM	STRN	SA	SB
-0.01184	-0.012338	-0.039625	-0.05846	-0.007106	-0.084961	-0.022521

Above are the beta values of all the corresponding 14 X values and

Bias(intercept)= - 0.0003304246606654711, MSE = 0.4387971998242283

Prepared by: Eeshan Phatak **Date: 02/13/2020**

Objective

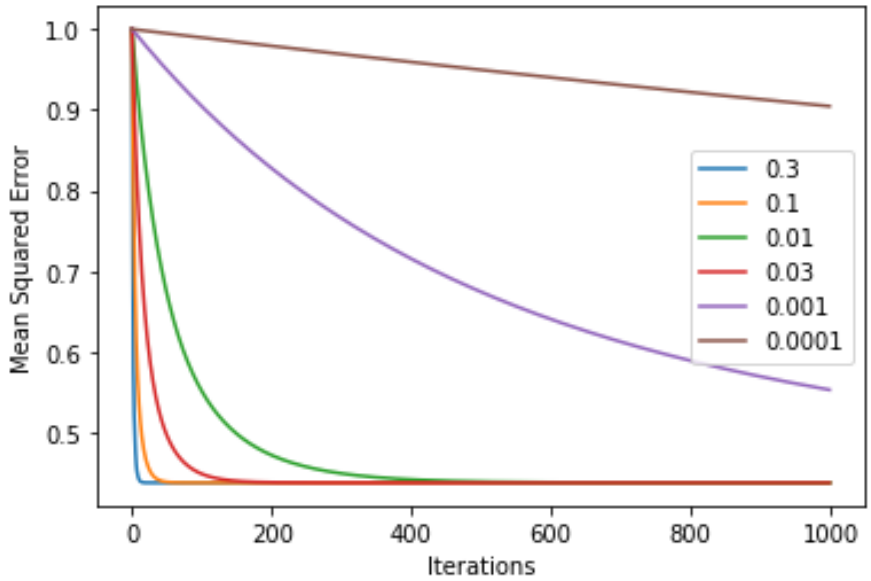
- Reduce the value of the cost function
- Find the optimum learning rate for a dataset
- Find the best threshold value for the convergence of gradient descent algorithm
- Find a reduced set of features which would still give good results

Experimentation

1. Change in Learning Rate

The learning rate is the tuning parameter in the algorithm which decides the step size at each iteration while moving towards the minimum of a loss function. We are going to change this parameter to find what value would give optimum results.

Linear Regression: Below is attached a plot of Iterations vs Mean Squared Error (Cost) for Different Learning Rates

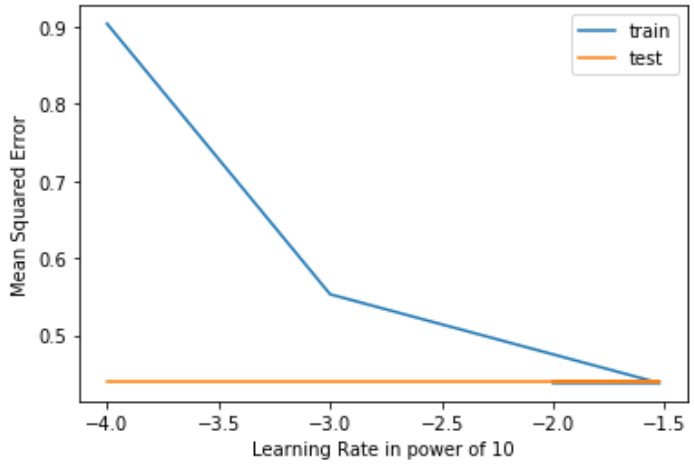


For Linear Regression we could reach the conclusion that the learning rate of 0.01 is the most optimal value. It has a very low MSE as well as a smooth gradient curve. It would not get stuck at any point and would find the global minima in optimum steps.

We can observe when the learning rate is very low the Error (MSE) is very high and it is substantially low for median (nither too high nor too low) values of Learning Rate. Even higher values of learning rate are not useful as they do not lower the error as much as the small values of learning rate.

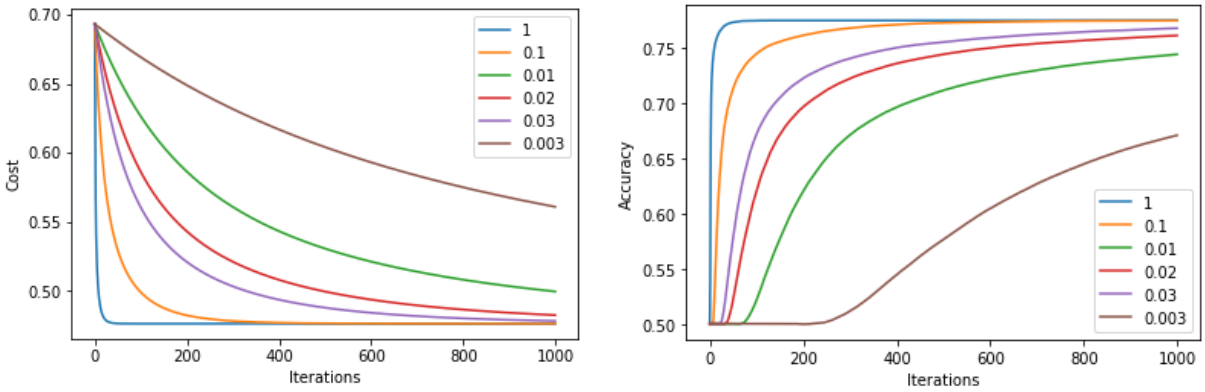
Prepared by: Eeshan Phatak **Date: 02/13/2020**

Train and Test plots of Learning Rate in power of 10 vs Mean Squared Error



Test set uses the previosly trained model and therefore the MSE value is already small. While we can see a sharp reduction in the MSE value for train set when the learning rate goes towards -2 and hence 0.01 is chosen as the optimal value for Learning Rate.

Logistic Regression: Below are attached plots of Iterations vs Cost and Iterations vs Accuracy for Different Learning Rates for Training set of Logistic Regression Algorithm



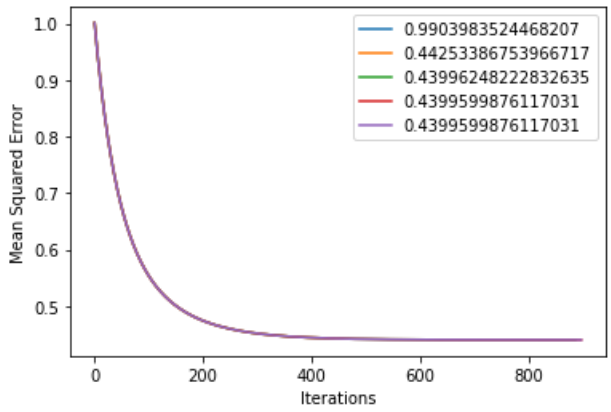
In this case the learning rate of 0.03 seems to be the optimal of all because it is not going very fast or very slow. It is the value which eventually reaches the minima and gives a very effective error value (higher accuracy)

2. Change in Convergence Threshold

Linear Regression: Below attached is the plot of number of iterations required to converge with threshold and the resulting MSE is seen in the legends in the image.

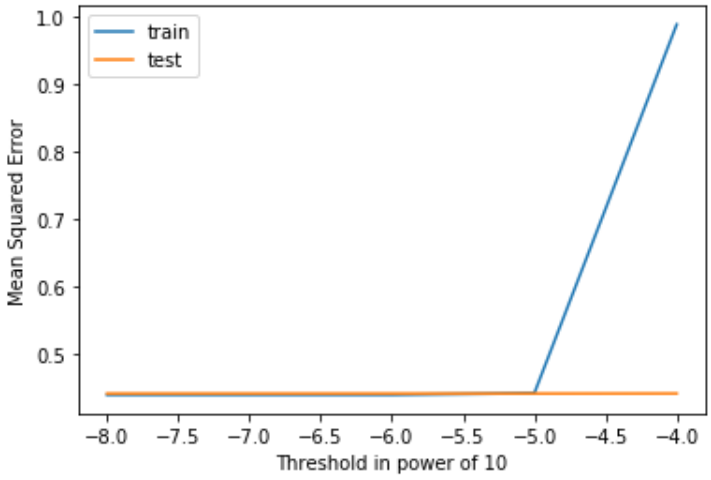
Prepared by: Eeshan Phatak **Date: 02/13/2020**

```
No. of Iterations required to reach Convergence with threshold  0.0001  is: 0
No. of Iterations required to reach Convergence with threshold  1e-05  is: 438
No. of Iterations required to reach Convergence with threshold  1e-06  is: 886
No. of Iterations required to reach Convergence with threshold  1e-07  is: 897
No. of Iterations required to reach Convergence with threshold  1e-08  is: 897
```



As we can see when the convergence threshold is very less it directly converges and gives a bad MSE value. The MSE value for convergence threshold 0.0001 is 0.99 while that for 0.000001 is 0.43 which is a much better value.

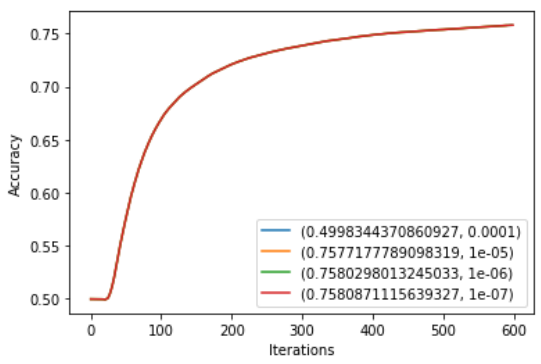
Train and Test plots of Threshold in power of 10 vs the Mean Squared Error

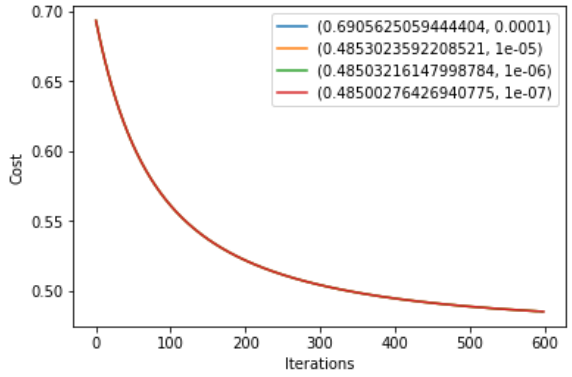


From this analysis 10 power -7 looks to be the optimal value for the threshold as the value of MSE neither the iterations seem to change much after this value of threshold.

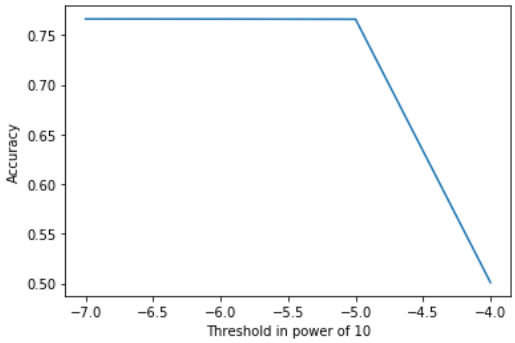
Logistic Regression: Below are attached plots of Iterations vs Cost and Iterations vs Accuracy for Different Convergence Thresholds for Logistic Regression Algorithm. The curves overlap each other but the values of accuracy and cost are seen in the legends.

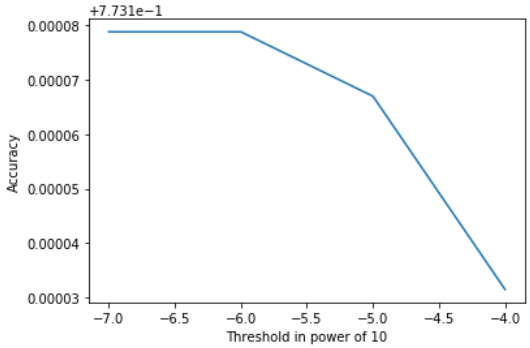
Prepared by: Eeshan Phatak **Date: 02/13/2020**



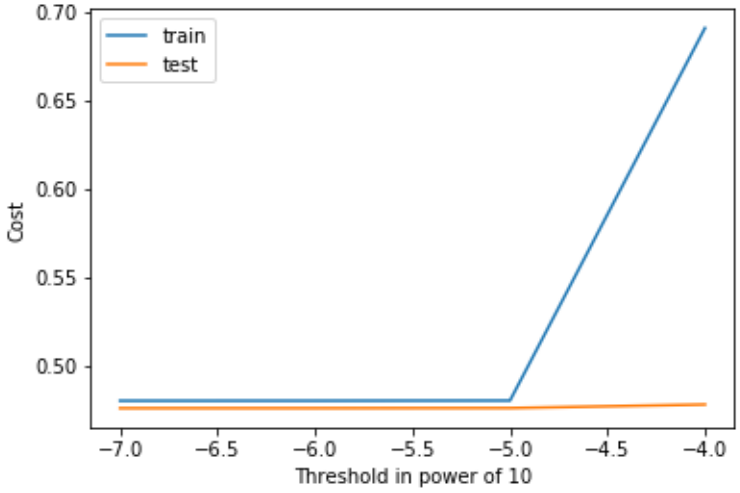


Train and Test plots of Threshold in power of 10 vs Accuracy





Train and Test plots of Threshold in power of 10 vs Cost



The cost value decreases and the accuracy increases in both train and test sets. When the threshold is 0.0001 the value of cost is very high and accuracy very low but the cost and accuracy values stabilize (very less change in cost and accuracy values) when the threshold is 0.000001 (10 power -6). That's why the optimal value could be 10 power -6 for the Convergence Threshold for Logistic Regression.

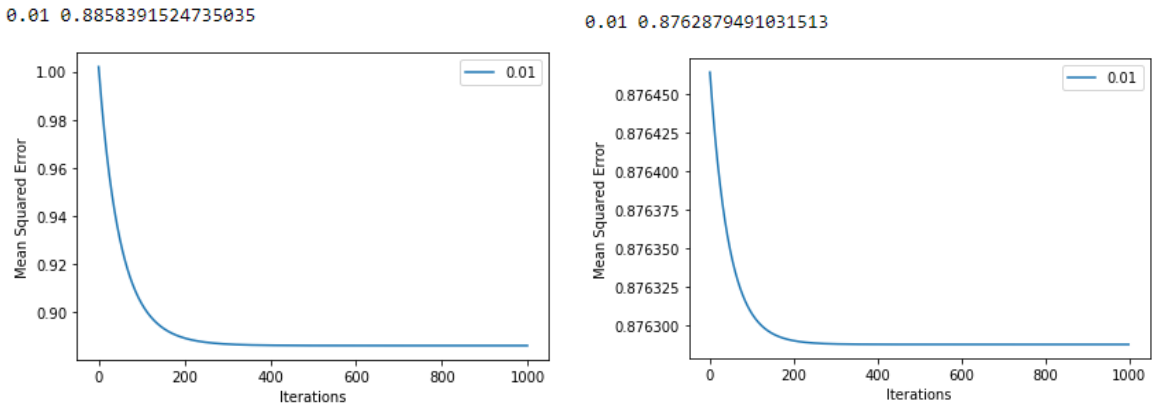
3. Random Features Selection

This is an experiment which we are doing to find out how much effect the randomly selected features have on the data. We will also find out how much different the error vales are from the values with all the features.

Linear Regression

```
updated_df = normalized_df.iloc[:,:14].sample(8,axis=1)
updated_df['Run_Avg'] = normalized_df['Run_Avg']
```

The above formula has been used to filter out any 8 features out of the total 14. Below are few plots obtained using these values to predict the dependent variable. Plot consists of Iterations vs MSE for Train and Test sets



In the above plot it is very clear that the random selection has went extremely wrong. The MSE value for this set of features is 0.885 for train while the MSE value for all the 14 features was 0.437. That means the MSE (error) is almost double that of the error calculated for all the features.

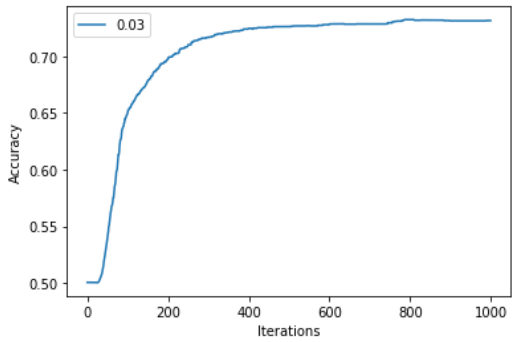
The randomly selected features in this case are:

NDIMC	MDIMA	SA	STRM	KWI	NDIMB	SB	VWM

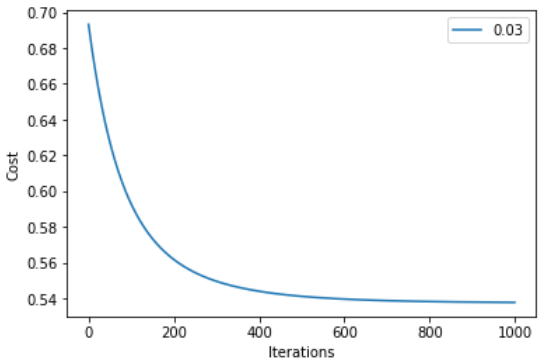
Logistic Regression

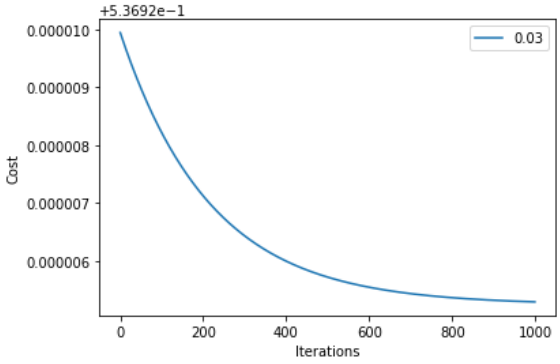
```
updated_df = project_df.iloc[:,:14].sample(8,axis=1)
updated_df['Run_Avg'] = project_df['Run_Avg']
```

The above formula has been used to filter out any 8 features out of the total 14. Below are few plots obtained using these values to predict the dependent variable. Plots consist of Iterations vs Cost and Iterations vs Accuracy for Learning Rates 0.03



Train and Test plots for No of iterations vs Cost





In the above plot it is very clear that the random selection has given an average result. The Accuracy value for this set of features is 0.73 while the Accuracy value for all the 14 features was 0.77 and the value of cost function is 0.54 which was 0.48 in the one with all 14 features. The change in the Accuracy and the cost function is also very less which indicates that this is not the right feature selection. The test plot shows a cost value of 0.53692 which is a small value but not the best we could get.

The Randomly Selected features in this case are:

MDIMA	KWG	NDIMB	STRN	NDIMC	MWG	STRM	MDIMC

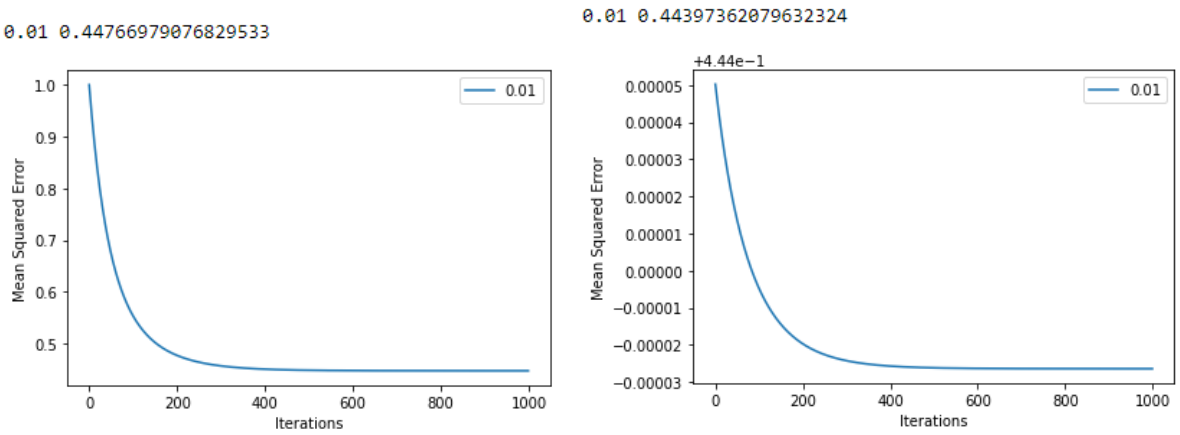
4. Handpicked 8 features which would better suit the output

To find 8 best features from the given 14 we plot a correlation matrix between the independent and dependent variables.

Linear Regression

We choose to drop the 6 the features which have the lowest correlation with the 'Run_Avg' column to get the best results.

Train and Test plots for Iterations vs Mean Squared Error

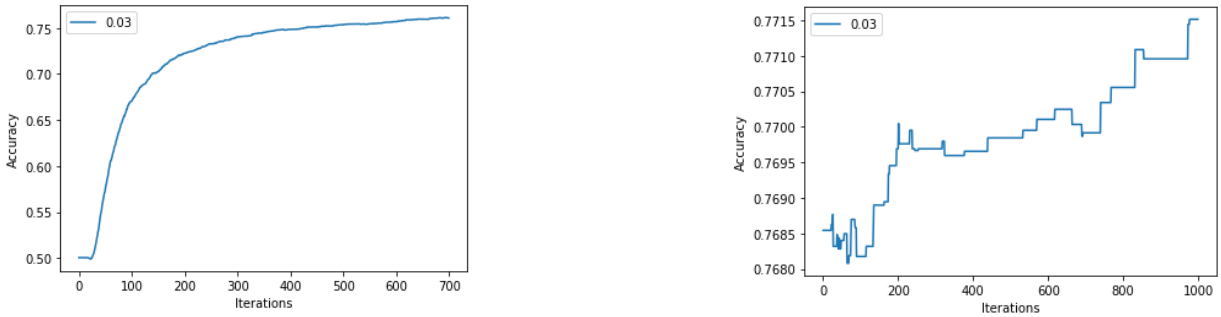


Here we can see that the MSE value reduces to 0.44. We get this value for both train and test plots and it is almost equal to the MSE test value of 0.43 value of the plot where all the 14 features were considered. So, these are the optimal 8 features we could pick to get the best results. The features picked here are:

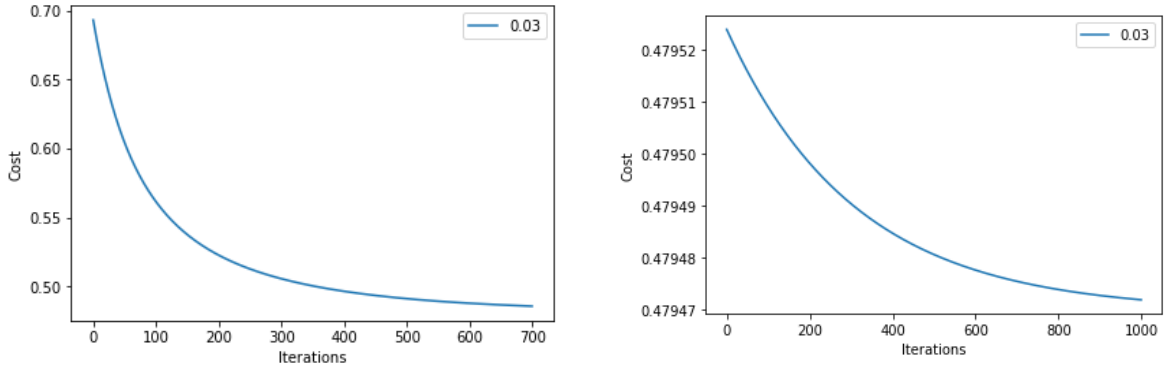
MWG	NWG	MDIMC	NDIMC	VWM	VWN	STRM	SA

Logistic Regression

We choose to drop the 6 the features which have the lowest correlation with the 'Run_Avg' column to get the best results. Train and Test plots for Iterations vs Accuracy



Train and Test Plots for Iterations vs Cost showing plots almost similar to the one with all the 14 features



Here we can see that the Accuracy (0.77) and Cost (0.479) are very close to the Accuracy (0.78) and Cost (0.46) values of the plots where all the 14 features were considered. So, these are the optimal 8 features we could pick to get the best results. The features picked here are:

MWG	NWG	MDIMC	NDIMC	VWM	VWN	STRM	SA

Results:

Learning Rates: Linear Regression 0.01, Logistic Regression 0.03

Convergence Threshold: Linear Regression: 10 power -7, Logistic Regression: 10 power -6

Interpretation of Results:

- From the values of Learning Rate, it can be said that a learning rate in the range 0.01 to 0.1 is a good range for the Learning Rate.
- A high conversion threshold (in range 0.001 and above) would mean that the algorithm would not reach the optimum value and would have more error. A very low conversion threshold (less than 10 power -15) would mean that we are wasting resources because the algorithm might already have found the optimum value long time back.
- Random Selection on variables is completely luck based it might give optimum value once but not every time.
- Selection of features using some method like using a correlation matrix to eliminate the less important features would help find optimum value.

Things that would matter most towards predicting GPU Run Time:

- Selection of correct features and removing irrelevant features
- Accurate learning rate
- Appropriate threshold values
- Data Preparation (log transformations and normalization) plays a very important role

Possible Improvements for Better Results:

- Changing Learning Rate by very small values once finding an optimum learning rate to improve it even more.
- Using better algorithm like Wrapper method which would calculate values for all combinations of the features and would return the best matching combination of Features for the data frame.
- Use of few more techniques while preparing dependent variable for Logistic Regression other than median classifier to classify the dependent variable to 1 and 0.