Linear and Logistic Regression with Gradient Descent

AMIT V GOTTIPATI

Introduction

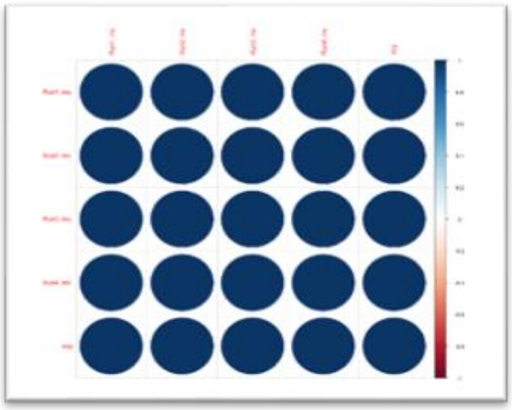
In the following project gradient descent with batch update for linear regression and logistic regression was implemented in RStudio using the SGEMM GPU kernel performance data set from UCI Machine Learning repository -> Dataset . The project involved data pre-processing, data visualization, algorithm creation, prediction and experimenting with different hyper-parameters like learning rate, convergence threshold, iterations.

Dataset Description

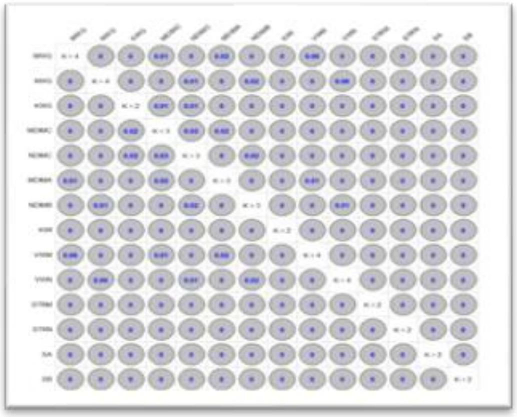
- The dataset contained 241600 observations and 18 variables with no missing values.
- There are 18 variables, the first 14 are parameters upon which we will train our model. The first 10 parameters are ordinal with 4 levels, the next 4 are binary and last 4 variables are 4 different run times for the SGEMM GPU
- The description for the independent variables is given in the dataset link.
- The dataset was divided into train and test with a 70:30 split.
- For linear regression the dependent variable 'avg' was created using average of the 4 run times provided which tells us the average running time for matrix-matrix product of the four runs
- For logistic regression the dependent variable 'avg' was converted into a binary variable using median as the threshold. (High/Low)

Exploratory Data Analysis

- Plotting the correlation of the four different runs and the average run time we can see that they are highly correlated. We have removed the four different run times so that we can train our model better.

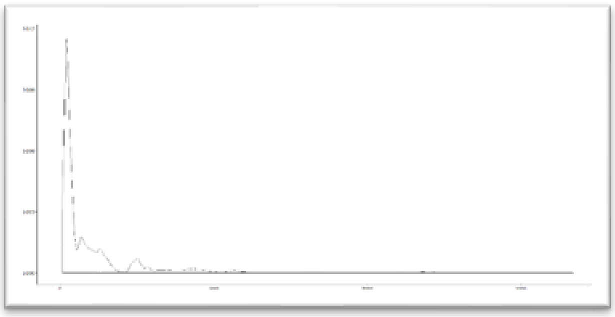


- We performed Goodman-Kruskal test for ordinal variables for checking multi-collinearity in the dataset. We see that there is weak association and no multi-collinearity between the ordinal variables.

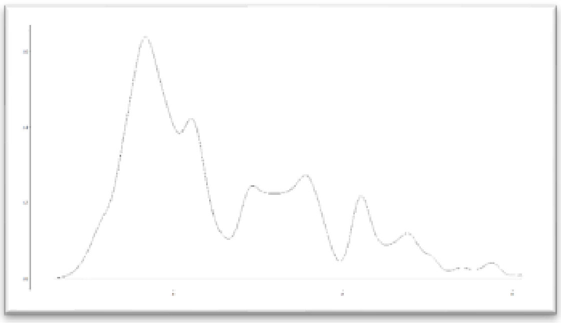


- We checked the distribution of our target variable 'avg'; we see that its distribution is right skewed. After applying log transformation, the data is not so much skewed anymore.

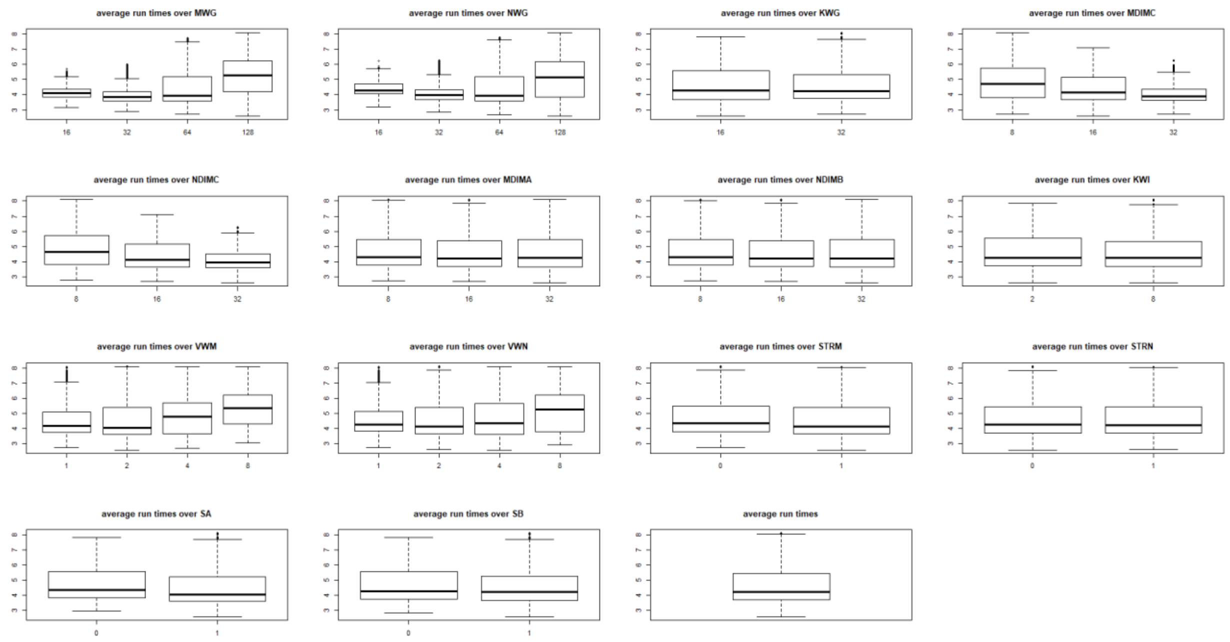
Before log transformation:



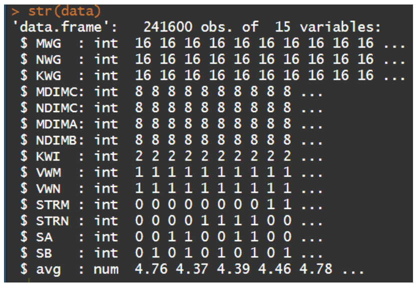
After log transformation:



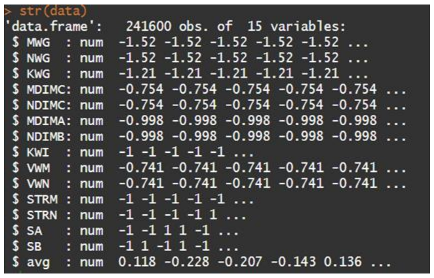
- We looked for outliers in our dataset by creating boxplots for each ordinal independent variable with respect to the 2continuous dependent variable ‘avg’. After applying log the outliers decreased.



- Looking at the parameters and target variable we see that they are of different scales. We have normalized them so that our model trains better. The features are standardized by subtracting by mean and dividing by standard deviation of respective variable. Standardization helps in performance of gradient descent when features have similar scale.



```
> str(data)
'data.frame':   241600 obs. of  15 variables:
 $ MWG  : int  16 16 16 16 16 16 16 16 16 16 ...
 $ NWG  : int  16 16 16 16 16 16 16 16 16 16 ...
 $ KWG  : int  16 16 16 16 16 16 16 16 16 16 ...
 $ MDIMC: int  8 8 8 8 8 8 8 8 8 8 ...
 $ NDIMC: int  8 8 8 8 8 8 8 8 8 8 ...
 $ MDIMA: int  8 8 8 8 8 8 8 8 8 8 ...
 $ NDIMB: int  8 8 8 8 8 8 8 8 8 8 ...
 $ KWI  : int  2 2 2 2 2 2 2 2 2 2 ...
 $ VWM  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ VWN  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ STRM : int  0 0 0 0 0 0 0 0 1 1 ...
 $ STRN : int  0 0 0 0 1 1 1 1 0 0 ...
 $ SA   : int  0 0 1 1 0 0 1 1 0 0 ...
 $ SB   : int  0 1 0 1 0 1 0 1 0 1 ...
 $ avg  : num  4.76 4.37 4.39 4.46 4.78 ...
```



```
> str(data)
'data.frame':   241600 obs. of  15 variables:
 $ MWG  : num  -1.52 -1.52 -1.52 -1.52 -1.52 ...
 $ NWG  : num  -1.52 -1.52 -1.52 -1.52 -1.52 ...
 $ KWG  : num  -1.21 -1.21 -1.21 -1.21 -1.21 ...
 $ MDIMC: num  -0.754 -0.754 -0.754 -0.754 -0.754 ...
 $ NDIMC: num  -0.754 -0.754 -0.754 -0.754 -0.754 ...
 $ MDIMA: num  -0.998 -0.998 -0.998 -0.998 -0.998 ...
 $ NDIMB: num  -0.998 -0.998 -0.998 -0.998 -0.998 ...
 $ KWI  : num  -1 -1 -1 -1 -1 ...
 $ VWM  : num  -0.741 -0.741 -0.741 -0.741 -0.741 ...
 $ VWN  : num  -0.741 -0.741 -0.741 -0.741 -0.741 ...
 $ STRM : num  -1 -1 -1 -1 -1 ...
 $ STRN : num  -1 -1 -1 -1 1 ...
 $ SA   : num  -1 -1 1 1 -1 ...
 $ SB   : num  -1 1 -1 1 1 ...
 $ avg  : num  0.118 -0.228 -0.207 -0.143 0.136 ...
```

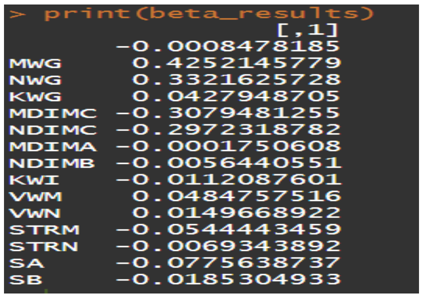
Task 1,2 and 3

Linear regression Model Equation for estimating the average run times from the 14 parameters is:

Log(avg) = B0 + B1*MWG + B2*NWG + B3*KWG + B4*MDIMC + B5*NDIMC + B6*MDIMA + B7*NDIMB + B8*KWI + B9*VWM + B10*VWN + B11*STRM + B12*STRN + B13*SA + B14*SB

The initial Parameters after implementing a linear regression are on the right.

We used Alpha of 0.01, Threshold as 0.0001 and 10000 iterations.



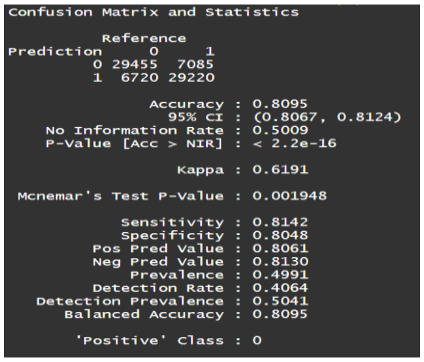
```
> print(beta_results)
              [,1]
     -0.0008478185
MWG   0.4252145779
NWG   0.3321625728
KWG   0.0427948705
MDIMC -0.3079481255
NDIMC -0.2972318782
MDIMA -0.0001750608
NDIMB -0.0056440551
KWI  -0.0112087601
VWM   0.0484757516
VWN   0.0149668922
STRM -0.0544443459
STRN -0.0069343892
SA   -0.0775638737
SB   -0.0185304933
```

Task 4

Implementing a logistic regression with Alpha =1, Threshold =0.0001 and Iterations= 10000

We got Train error as 0.07527 and our Test error as 0.0741.

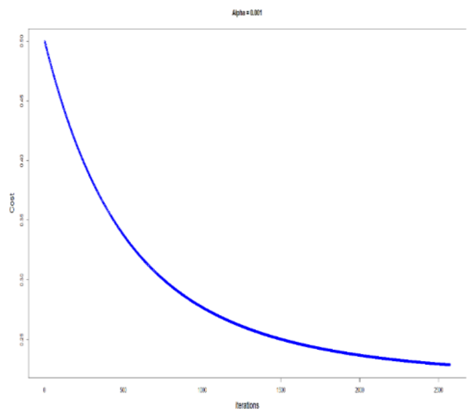
The confusion Matrix on the right shows the accuracy metrics

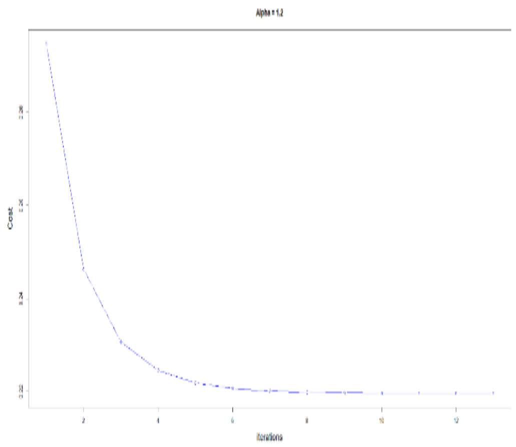


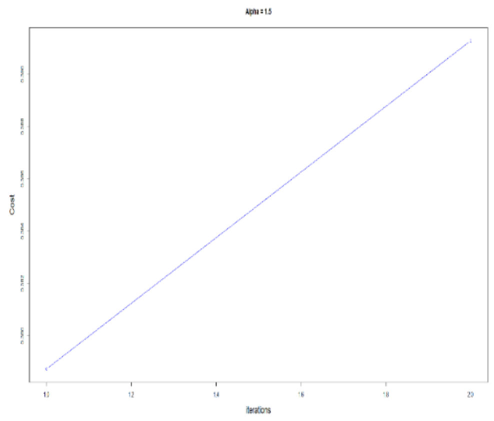
```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 29455  7085
         1  6720 29220

               Accuracy : 0.8095
                 95% CI : (0.8067, 0.8124)
    No Information Rate : 0.5009
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6191

 Mcnemar's Test P-Value : 0.001948

            Sensitivity : 0.8142
            Specificity : 0.8048
         Pos Pred Value : 0.8061
         Neg Pred Value : 0.8130
             Prevalence : 0.4991
         Detection Rate : 0.4064
   Detection Prevalence : 0.5041
      Balanced Accuracy : 0.8095

       'Positive' Class : 0
```

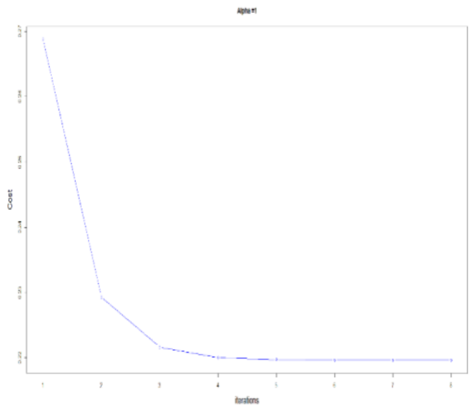
Experiment 1: From this experiment we observe that our cost will alpha increases, the algorithm reaches convergence when the cost does not decrease by a set threshold. Alpha is the learning rate; it determines the step size for gradient descent. With small alpha the convergence takes more iterations to converge. When the alpha is large, the number of iterations required for convergence are less. We experimented with different alphas for linear and logistic regression.

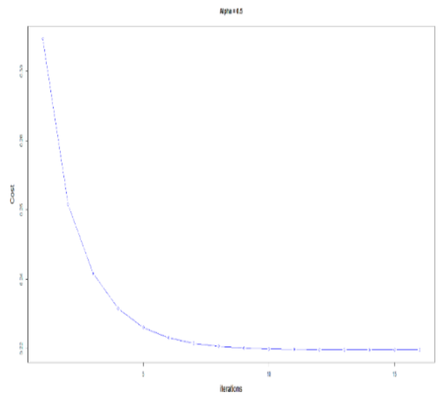
Linear Regression:

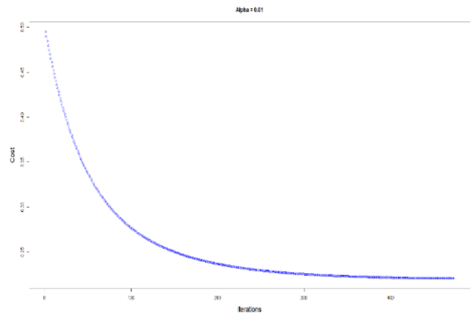


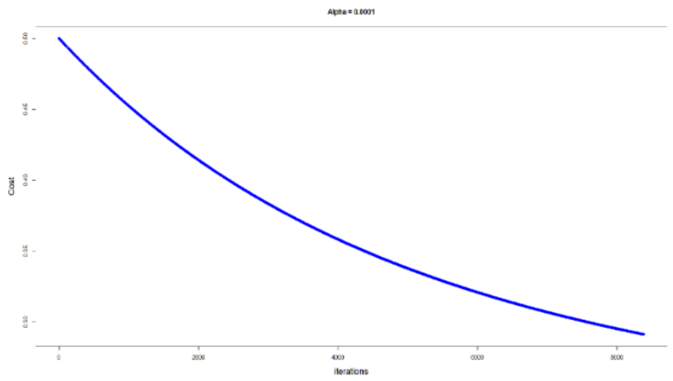


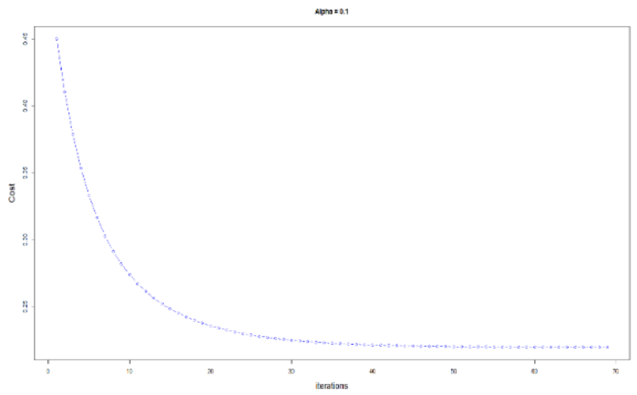












Threshold=0.00001					
Alpha	Iterations	iterations taken to converge	MSE train	MSE Test	MSE diff (train-test)
1.5	10000	2	0.3787273	0.3897829	-0.0110556
1.2	10000	13	0.2196152	0.2189185	0.0006967
1	10000	8	0.2196081	0.2189042	0.0007039
0.5	10000	16	0.2196152	0.2189095	0.0007057
0.1	10000	69	0.2196875	0.2189735	0.000714
0.01	10000	473	0.2205443	0.2198019	0.0007424
0.001	10000	2572	0.2288839	0.2280869	0.000797
0.0001	10000	8385	0.2912685	0.2905749	0.0006936

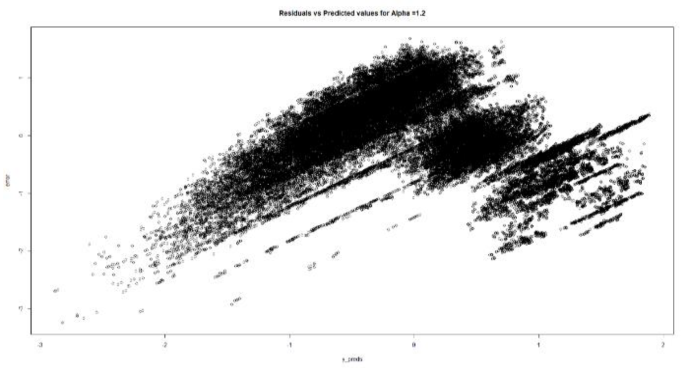
By varying alpha at threshold of 0.00001,

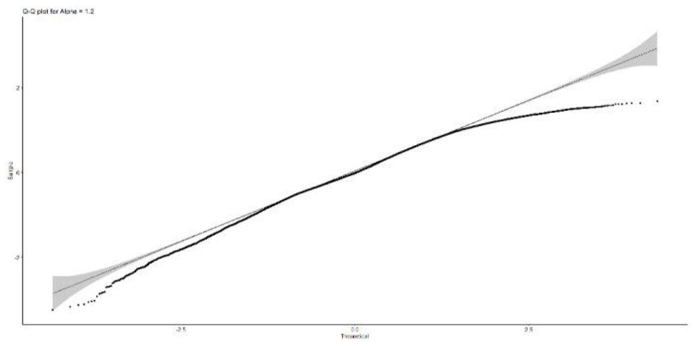
- We can see from the plots and above table that the best alpha value lies between 1.2 and 0.5.
- The best value of alpha from the following experiment turns out to be 1.2 as the steps taken to converge are less, the plot shows decreasing value of cost with each iteration and the difference between the MSE of test and train is lowest.
- We also see that taking alpha more than or equal to 1.5 the cost is not converging anymore, the cost increases.

Best Model at Alpha = 1.2, threshold =0.00001

Log(avg) = -0.0003 + 0.5011*MWG + 0.3973*NWG + 0.0854*KWG - 0.3987 *MDIMC -0.3841*NDIMC -0.0001*MDIMA -0.0020*NDIMB -0.0122*KWI -0.0133*VWM -0.0399*VWN -0.0588*STRM -0.0076*STRN -0.084*SA -0.0198*SB

Plotting the errors for checking homoscedasticity and normal distribution

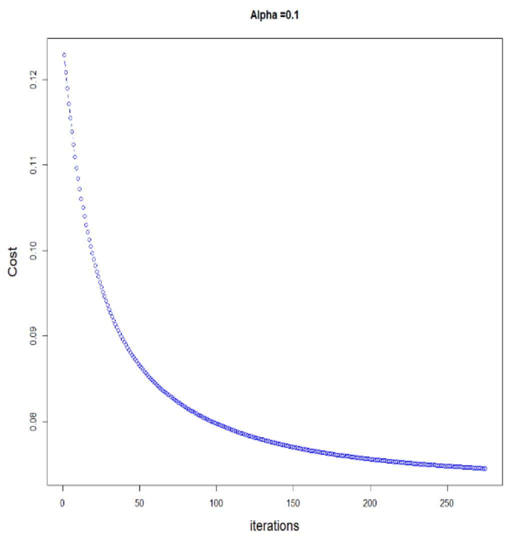


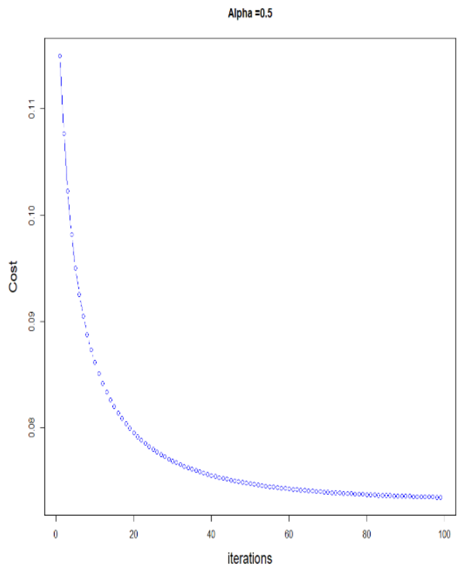


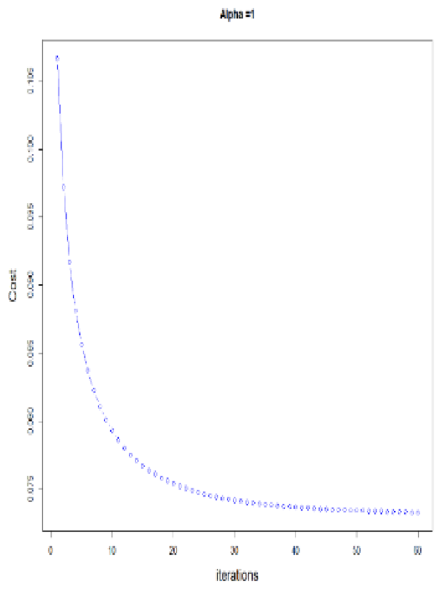
a)We need homoscedasticity for better prediction in linear regression. From the above plot of residuals vs predicted values we see that there is constant variance for most part of our data.

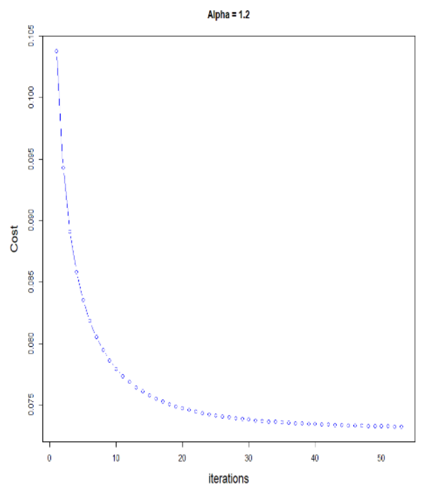
b)Q-Q plot ->We also need our errors to be normally distributed for better prediction in linear regression. We see here that only near the ends the errors are not normally distributed.

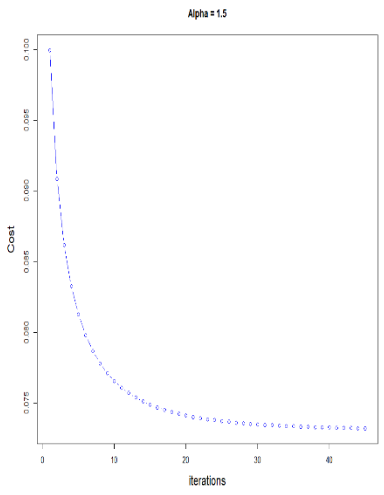
Logistic Regression:

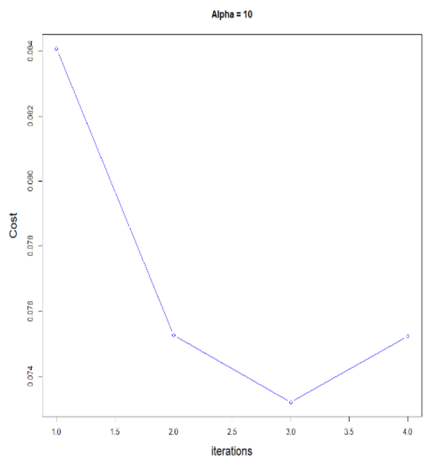


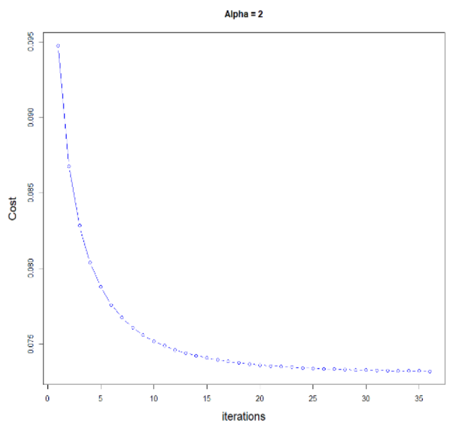


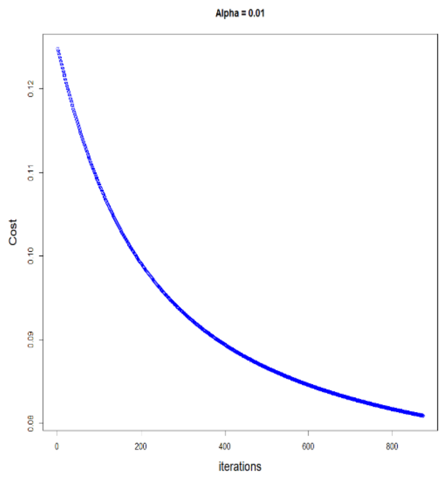


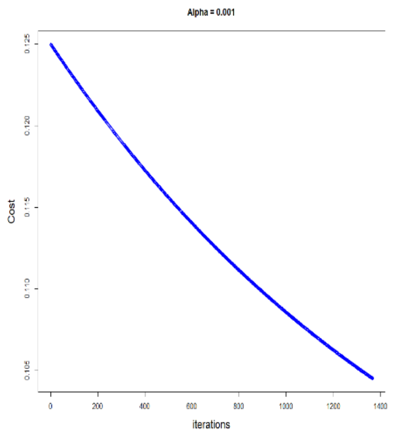




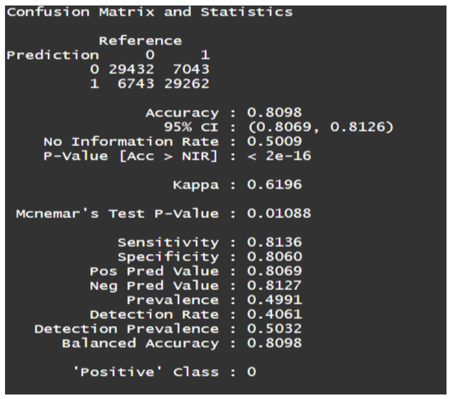




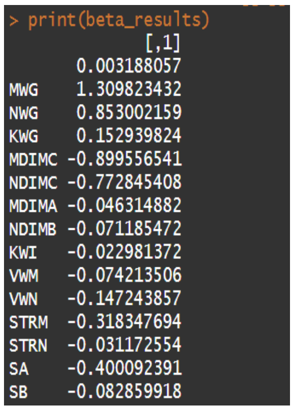




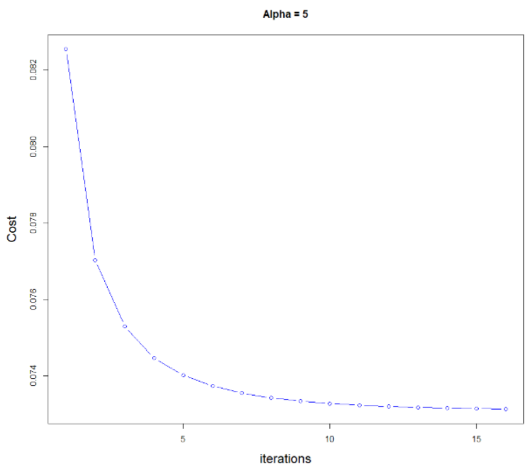
Confusion Matrix for Alpha =5



```
Confusion Matrix and Statistics

             Reference
Prediction      0     1
         0 29432  7043
         1  6743 29262

               Accuracy : 0.8098
                 95% CI : (0.8069, 0.8126)
    No Information Rate : 0.5009
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6196

 Mcnemar's Test P-Value : 0.01088

            Sensitivity : 0.8136
            Specificity : 0.8060
         Pos Pred Value : 0.8069
         Neg Pred Value : 0.8127
             Prevalence : 0.4991
         Detection Rate : 0.4061
   Detection Prevalence : 0.5032
      Balanced Accuracy : 0.8098

       'Positive' Class : 0
```



```
> print(beta_results)
             [,1]
      0.003188057
MWG   1.309823432
NWG   0.853002159
KWG   0.152939824
MDIMC -0.899556541
NDIMC -0.772845408
MDIMA -0.046314882
NDIMB -0.071185472
KWI   -0.022981372
VWM   -0.074213506
VWN   -0.147243857
STRM  -0.318347694
STRN  -0.031172554
SA    -0.400092391
SB    -0.082859918
```



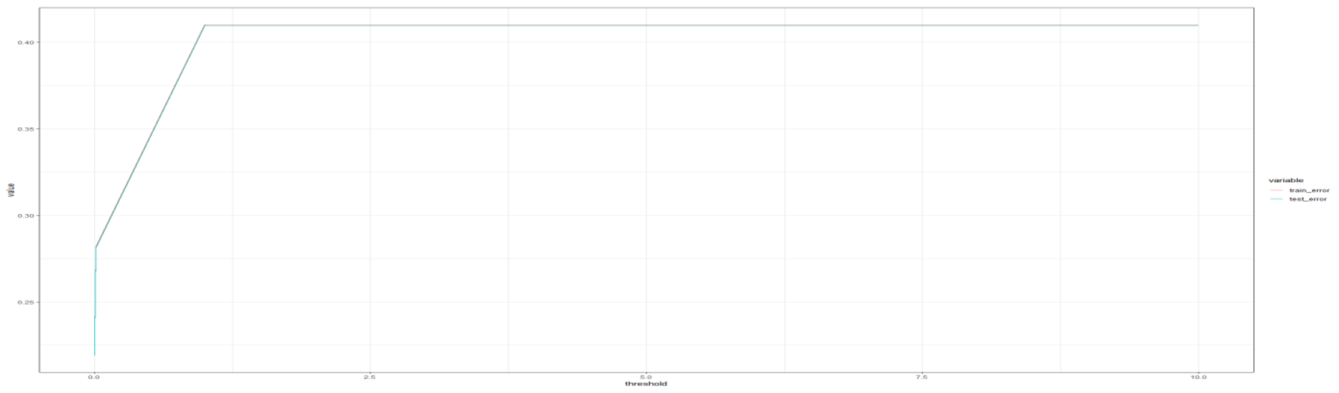
Threshold=0.00001					
Alpha	Iterations	iterations taken to converge	MSE train	MSE Test	MSEdiff(train-test)
0.001	10000	1366	0.1045083	0.1044226	8.57E-05
0.01	10000	873	0.08090957	0.08067033	0.00023924
0.1	10000	275	0.0744949	0.07427574	0.00021916
0.5	10000	99	0.0734454	0.07324586	0.00019954
1	10000	60	0.07328625	0.07309038	0.00019587
1.2	10000	53	0.07325107	0.07305603	0.00019504
1.5	10000	45	0.07321947	0.0730252	0.00019427
2	10000	36	0.07318951	0.07299596	0.00019355
5	10000	16	0.07313724	0.07294496	0.00019228
10	10000	4	0.07320717	0.07516083	-0.00195366

We see from this experiment that the best Alpha for logistic regression lies between 2 and 10. In our experiment alpha = 5 performed the best as there was less difference between train and test error and the plot was converging.

Experiment 2:

Linear Regression:

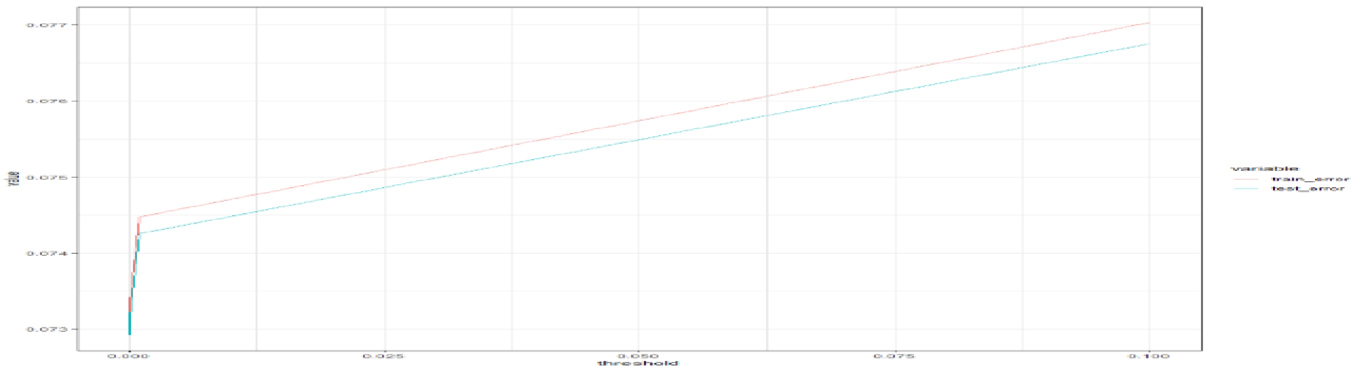
Alpha = 0.1					
Threshold	Iterations	iterations taken to converge	MSE train	MSE Test	MSEdiff(train-test)
10 (Not Converging)	10000	2	0.4103538	0.4097237	0.0006301
1 (Not Converging)	10000	2	0.4103538	0.4097237	0.0006301
0.01	10000	9	0.2819965	0.2812822	0.0007143
0.001	10000	26	0.2279243	0.2271263	0.000798
0.0001	10000	47	0.220464	0.2197232	0.0007408
0.00001	10000	69	0.2196875	0.2189735	0.000714
0.0000001	10000	112	0.2196078	0.2189042	0.0007036
1E-10	10000	176	0.219607	0.2189043	0.0007027



From this experiment the error was increasing with increasing the threshold for a fixed alpha.

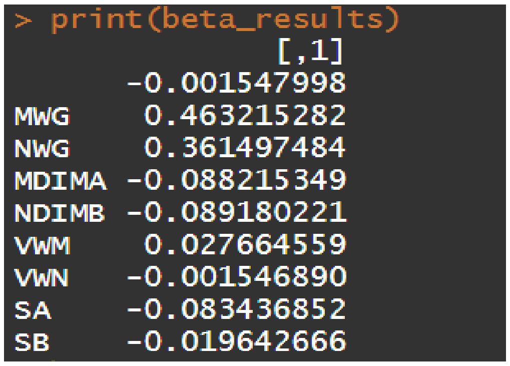
Logistic Regression:

Alpha=5					
Threshold	Iterations	iterations taken to converge	MSE train	MSE Test	MSEdiff(train-test)
1E-13	10000	82	0.07310571	0.07291524	0.00019047
1E-11	10000	65	0.07310571	0.07291524	0.00019047
0.000000001	10000	49	0.07310571	0.07291524	0.00019047
0.00001	10000	16	0.07313724	0.07294496	0.00019228
0.0001	10000	9	0.07334027	0.0731444	0.00019587
0.001	10000	4	0.07447203	0.07425548	0.00021655
0.1	10000	2	0.07703253	0.07675461	0.00027792

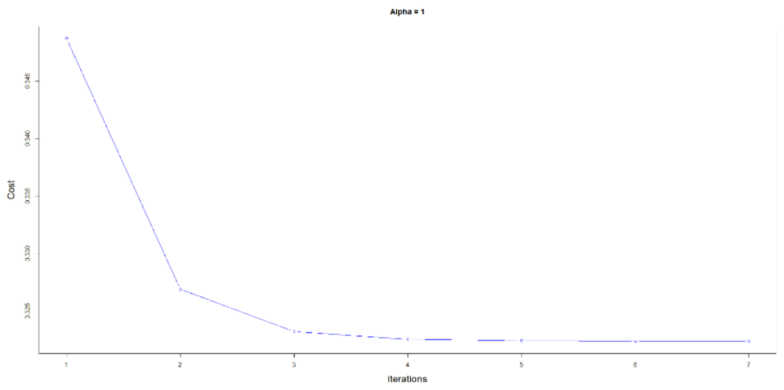


From this experiment we could not see much difference in error but increasing the threshold increased the error in this case also.

Experiment 3. Linear Regression:



```
> print(beta_results)
                [,1]
      -0.001547998
MWG    0.463215282
NWG    0.361497484
MDIMA -0.088215349
NDIMB -0.089180221
VWM    0.027664559
VWN   -0.001546890
SA    -0.083436852
SB    -0.019642666
```



No of features	Alpha	Iterations	iterations taken to converge	MSE train	MSE Test	MSE diff (train-test)
8 Random Features	1	10000	7	0.3223715	0.3195977	0.0027738
All 14 Features	1	10000	8	0.2196081	0.2189042	0.0007039

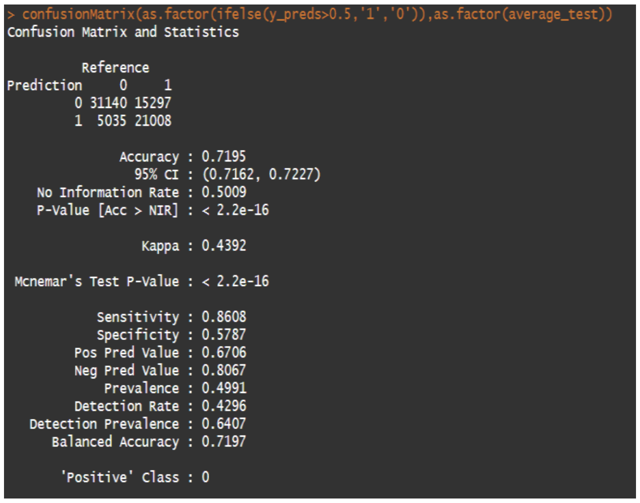
The Random features equation with threshold as 0.00001 and alpha=1

Log(avg) = -0.0015 + (0.4632)*MWG + (0.36149)*NWG + (-0.0882)*MDIMA + (-0.0891)*NDIMB + (0.0276)*VWM + (-0.0015)*VWN +(-0.0834)*SA + (-0.0196)*SB

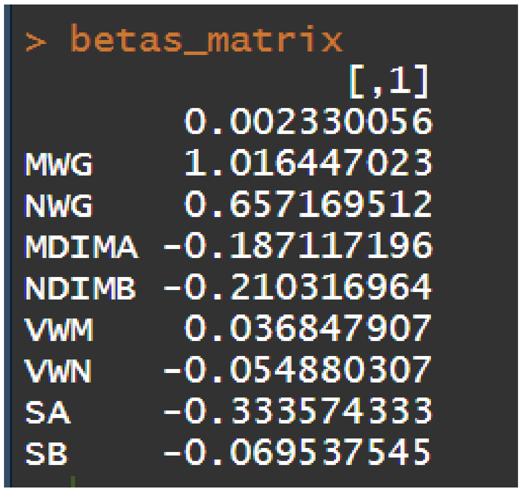
We see from this experimentation that the error is more when 8 features are randomly selected. This is because we do not know if these variables are significant or not for training this model. The all feature model performs better.

Logistic Regression:

Threshold=0.00001						
No of features	Alpha	Iterations	iterations taken to converge	MSE train	MSE Test	MSE diff (train-test)
8 Random Features	5	10000	7	0.09280245	0.7135058	-0.62070335
All 14 Features	5	10000	16	0.07313724	0.07294496	0.00019228



```
> confusionMatrix(as.factor(ifelse(y_preds>0.5,'1','0')),as.factor(average_test))
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 31140 15297
         1  5035 21008

               Accuracy : 0.7195
                 95% CI : (0.7162, 0.7227)
    No Information Rate : 0.5009
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4392

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8608
            Specificity : 0.5787
         Pos Pred Value : 0.6706
         Neg Pred Value : 0.8067
             Prevalence : 0.4991
         Detection Rate : 0.4296
   Detection Prevalence : 0.6407
      Balanced Accuracy : 0.7197

       'Positive' Class : 0
```



```
> betas_matrix
                [,1]
       0.002330056
MWG    1.016447023
NWG    0.657169512
MDIMA -0.187117196
NDIMB -0.210316964
VWM    0.036847907
VWN   -0.054880307
SA    -0.333574333
SB    -0.069537545
```

From this experiment we see that 8 selected features model has lower test error as compared to the all feature model but the accuracy from the confusion matrix we see is less as compared to all feature model by 10%.

Experiment 4.

Linear Regression:

Threshold=0.00001						
No of features	Alpha	Iterations	Iterations taken to converge	MSE train	MSE Test	MSE diff (train-test)
8 Random Features	1	10000	7	0.3223715	0.3195977	0.0027738
All 14 Features	1	10000	8	0.2196081	0.2189042	0.0007039
8 Selected features	1	10000	3	0.3294874	0.3269658	0.0025216

We see from this experiment that the 8 important features model works a little better than 8 random feature model but as compared to the all feature model it performs poorly. This happens because the all feature model captures more variation in the data and the predictions from it are better as a result of this.

Logistic Regression:

Threshold=0.00001						
No of features	Alpha	Iterations	iterations taken to converge	MSE train	MSE Test	MSE diff (train-test)
8 Random Features	5	10000	7	0.09280245	0.7135058	-0.62070335
All 14 Features	5	10000	16	0.07313724	0.07294496	0.00019228
8 Selected Features	5	10000	3	0.09276144	0.7190947	-0.62633326

From this experiment we see that there is huge difference in train and test error. The features selected are not able to classify the data properly. We need to add more variables so that our test error decreases.

Discussion

From this experiment we can conclude that most of the given features are significant and good for predicting GPU run times. Increasing the alpha the error rate increases as the steps for gradient descent are more aggressive leading to more errors. We can add features like temperature of GPU at each run, add more runs and try different GPU's for while experimenting to better predict our GPU run times. We can try Stochastic gradient descent and check which is working better and faster. We can also try different algorithms for prediction of our run times.