**Linear and Logistic Regression with Gradient Descent**

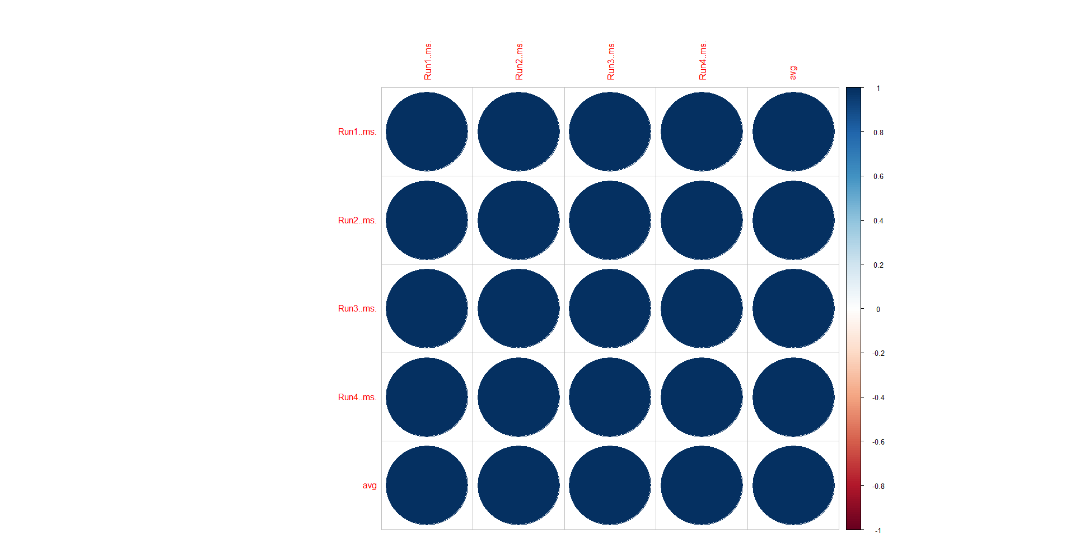
**AMIT V GOTTIPATI**

**Introduction**

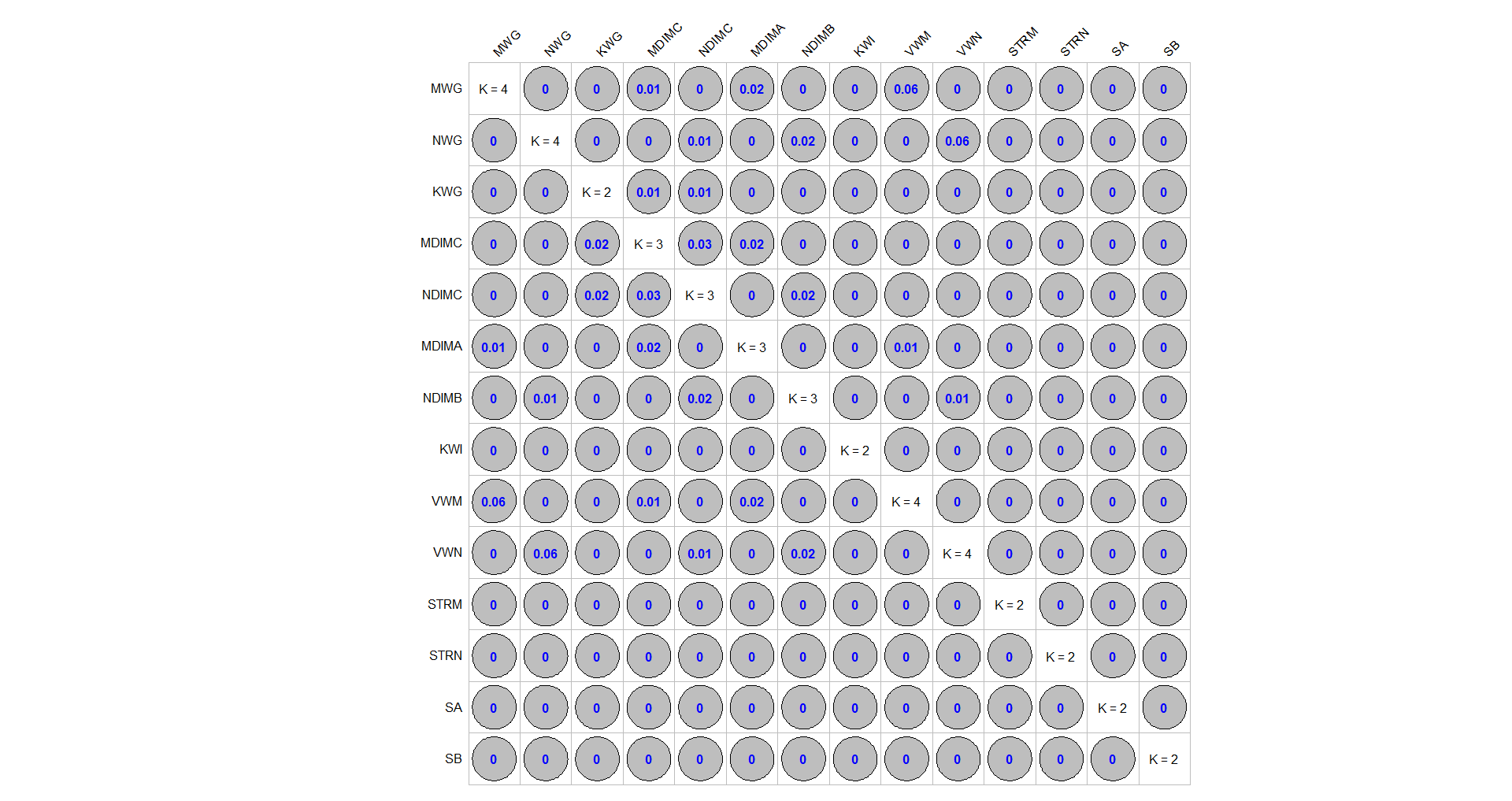
In the following project gradient descent with batch update for linear regression and logistic regression was implemented in RStudio using the SGEMM GPU kernel performance data set from UCI Machine Learning repository -> [Dataset](https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance) . The project involved data pre-processing, data visualization, algorithm creation, prediction and experimenting with different hyper-parameters like learning rate, convergence threshold, iterations.

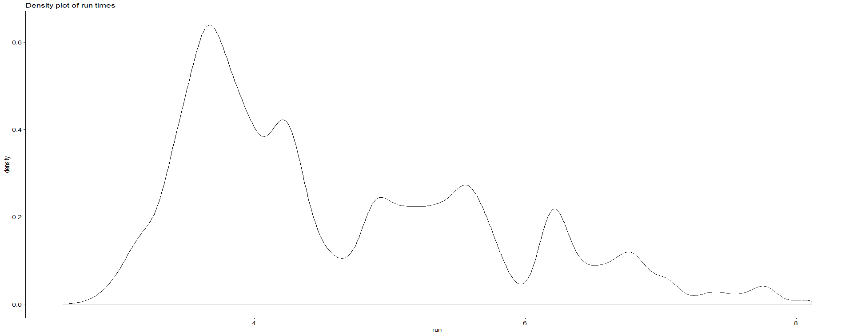
**Dataset Description**

* 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**

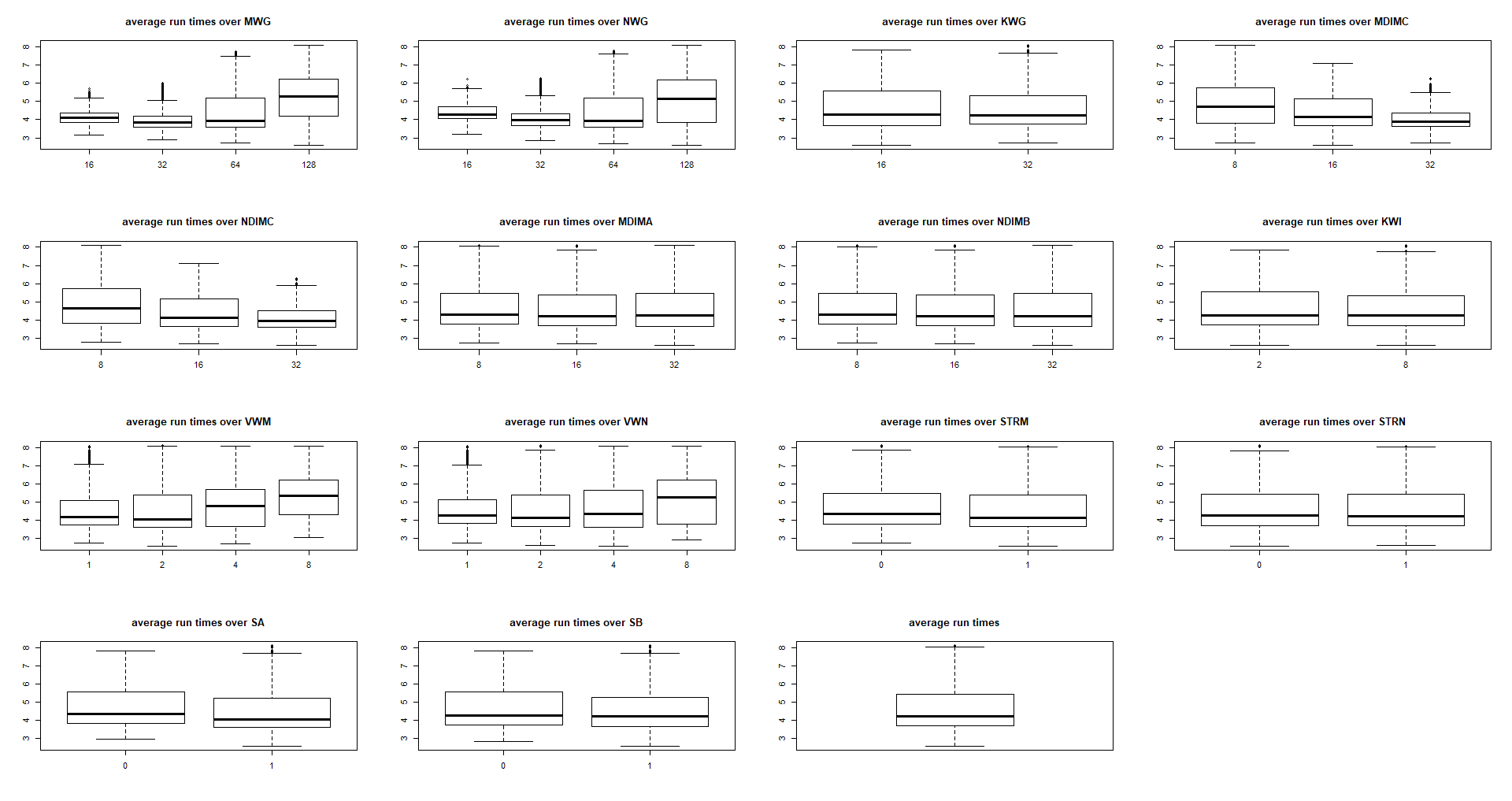
* Plotting the correlation of the four different runs and the average run time we can see that they are highly correlated. We have removed the four different run times so that we can train our model better.



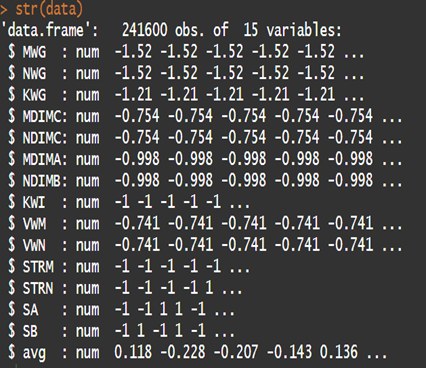
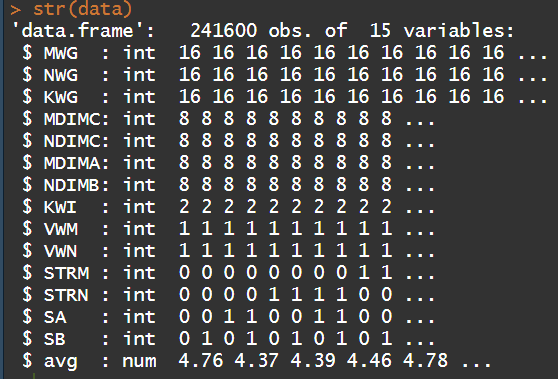
* We performed Goodman-Kruskal test for ordinal variables for checking multi-collinearity in the dataset. We see that there is weak association and no multi-collinearity between the ordinal variables.
* We checked the distribution of our target variable ‘avg’; we see that its distribution is right skewed. 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 continuous dependent variable ‘avg’. After applying log the outliers decreased.

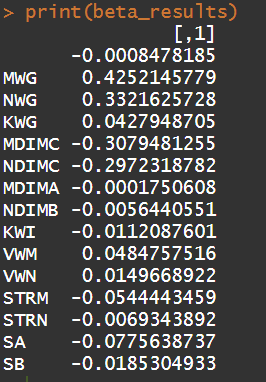


* Looking at the parameters and target variable we see that they are of different scales. We have normalized them so that our model trains better. The features are standardized by subtracting by mean and dividing by standard deviation of respective variable. Standardization helps in performance of gradient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



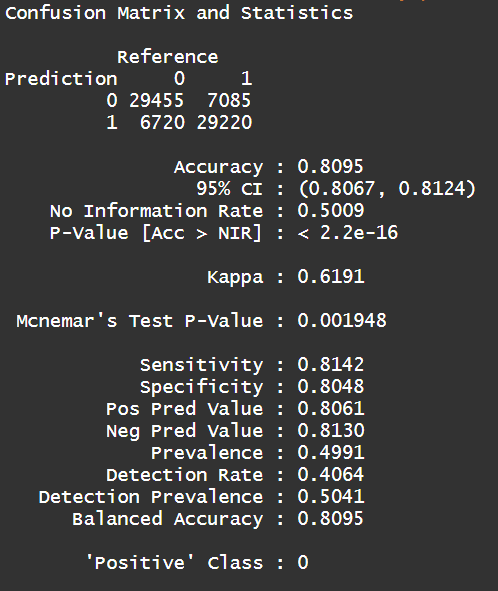
**Task 1,2 and 3**

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.

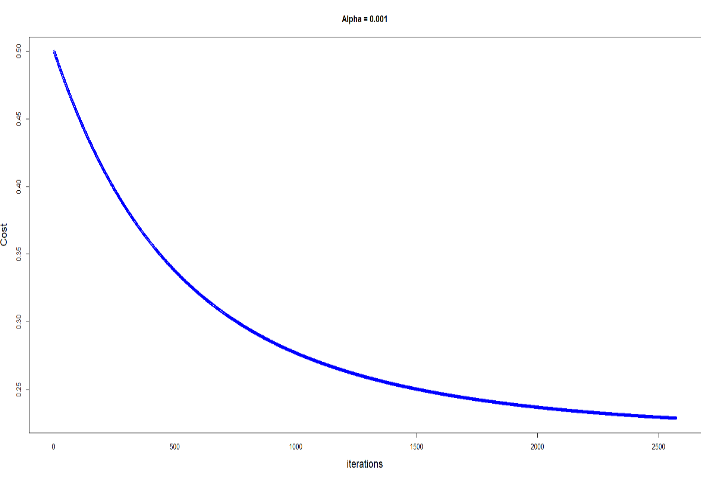
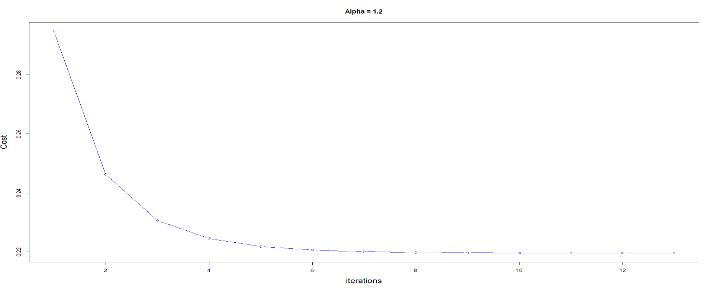
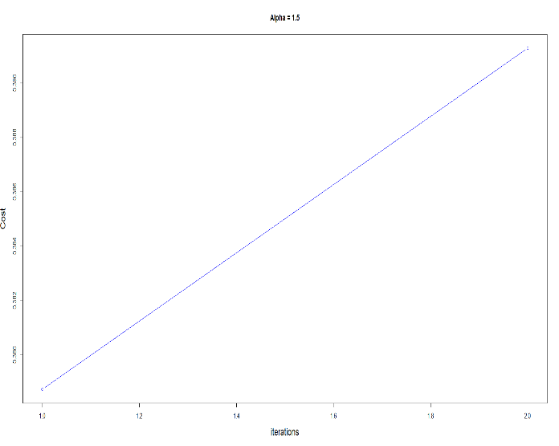
**Task 4**

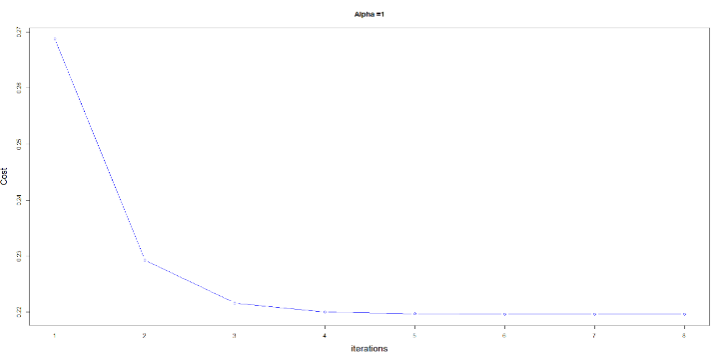
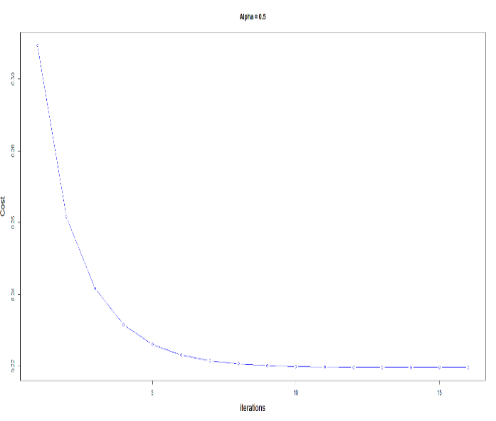
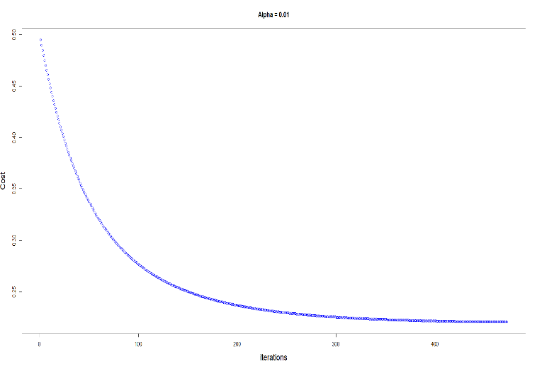
Implementing a logistic regression with Alpha =1, Threshold =0.0001 and Iterations= 10000

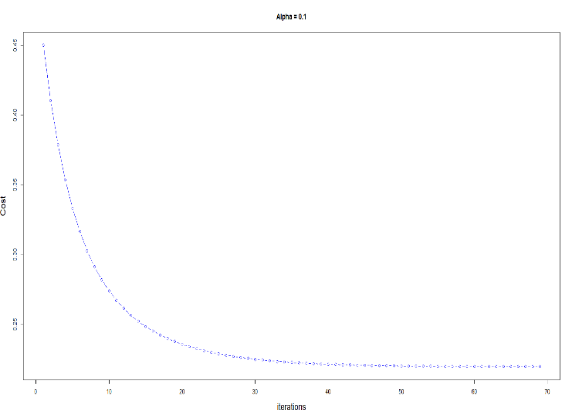
We got Train error as 0.07527 and our Test error as 0.0741.

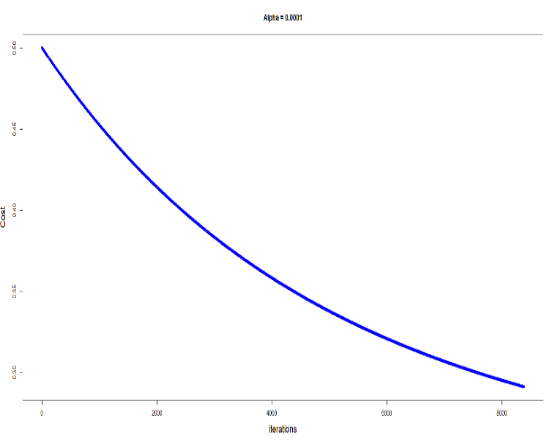
The confusion Matrix on the right shows the accuracy metrics

**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:**







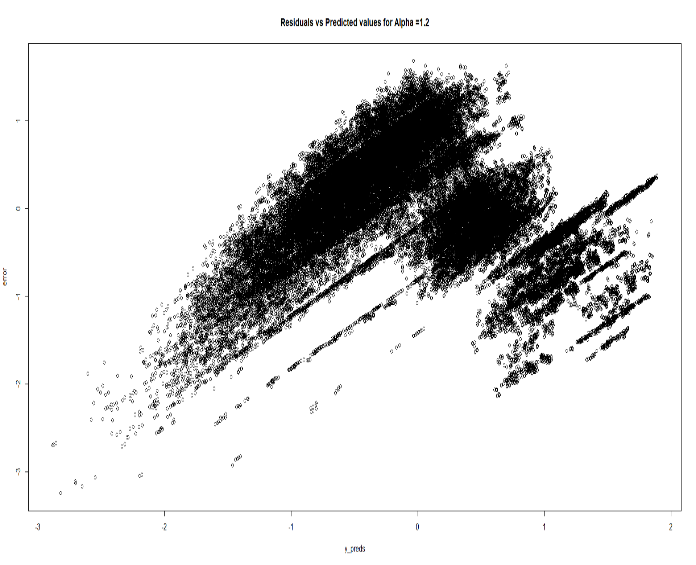
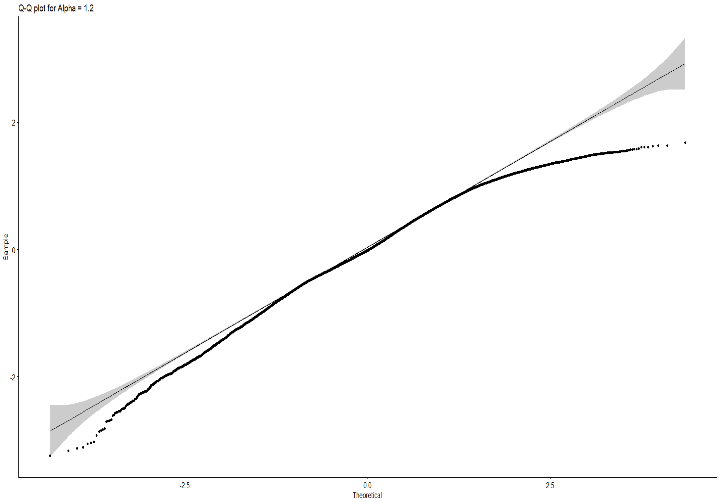
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Threshold=0.00001 |  |  |  |  |  |
| Alpha | Iterations | iterations taken to converge | MSE train | MSE Test | MSE diff  (train-test) |
| 1.5 | 10000 | 2 | 0.3787273 | 0.3897829 | -0.0110556 |
| 1.2 | 10000 | 13 | 0.2196152 | 0.2189185 | 0.0006967 |
| 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.

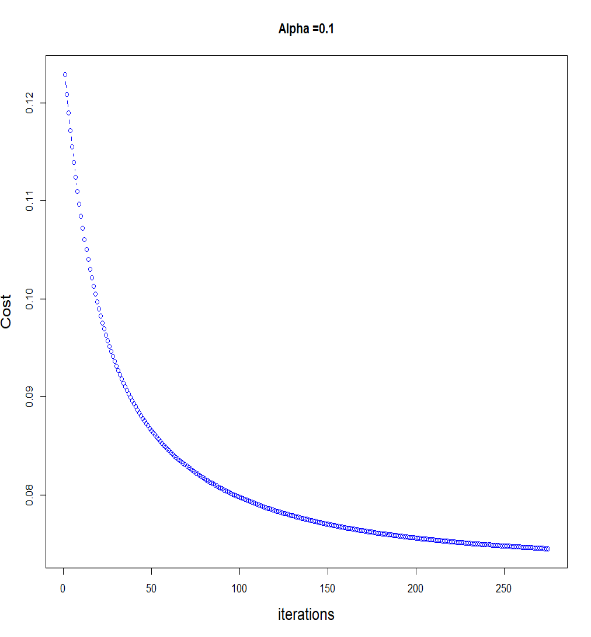
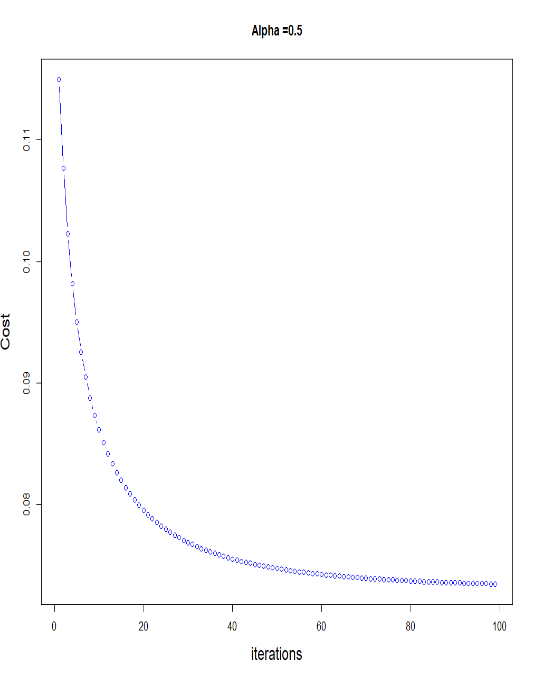
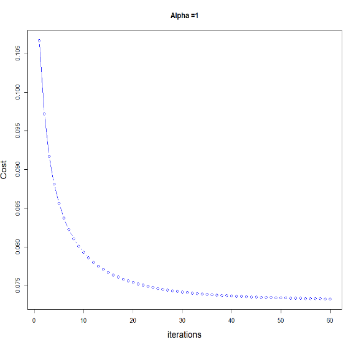
Best Model at Alpha = 1.2, threshold =0.00001

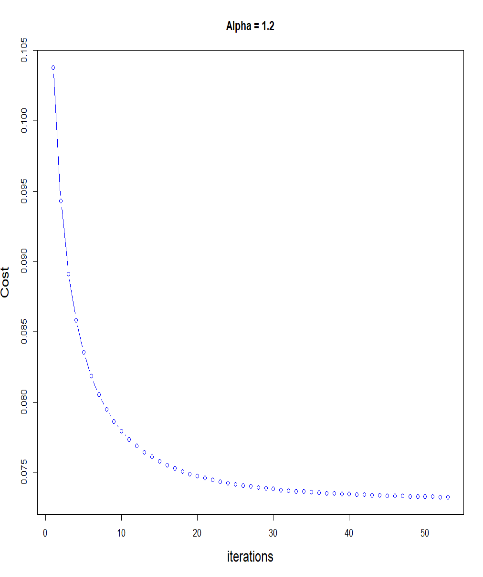
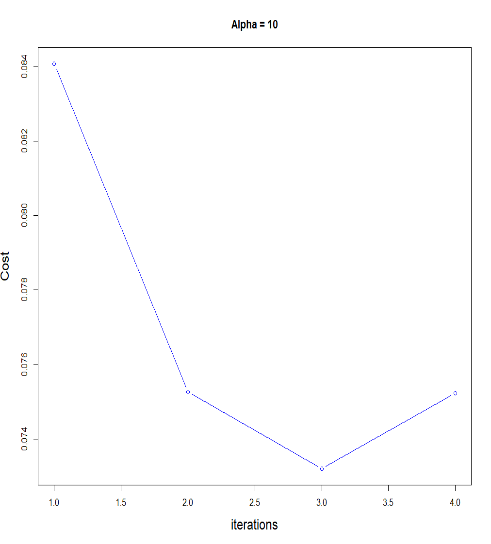
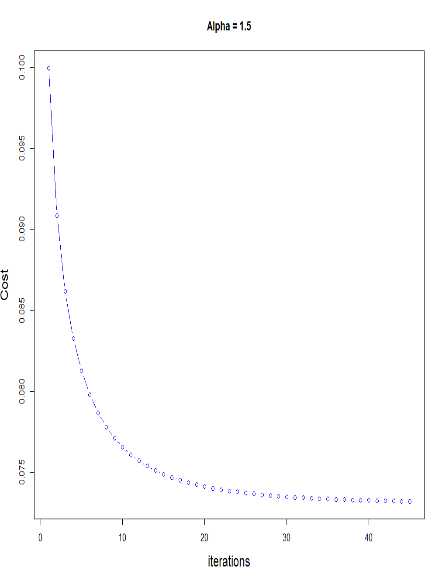
Log(avg) = -0.0003 + 0.5011\*MWG + 0.3973\*NWG + 0.0854\*KWG - 0.3987 \*MDIMC -0.3841\*NDIMC -0.0001\*MDIMA -0.0020\*NDIMB -0.0122\*KWI -0.0133\*VWM -0.0399\*VWN -0.0588\*STRM -0.0076\*STRN -0.084\*SA -0.0198\*SB

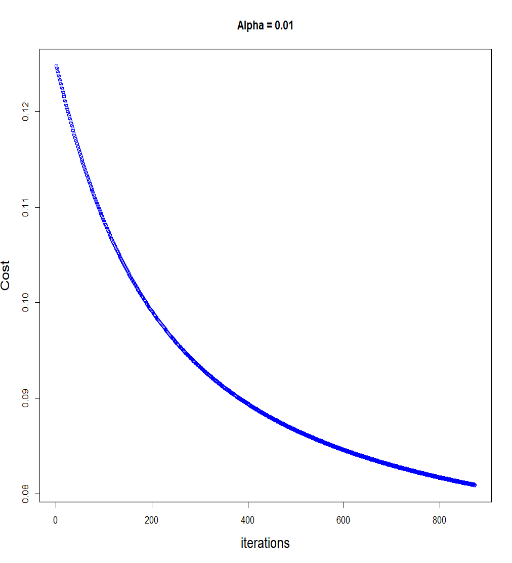
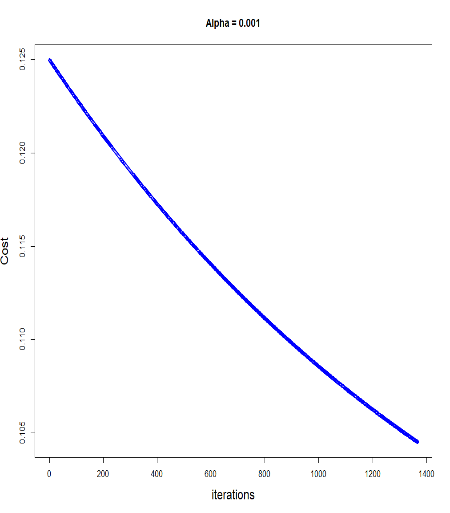
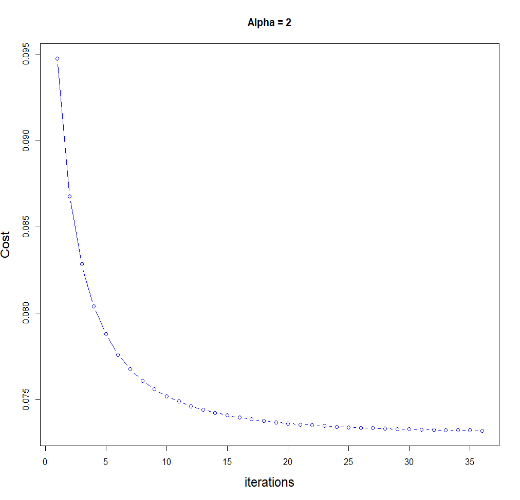
**Plotting the errors for checking homoscedasticity and normal distribution**

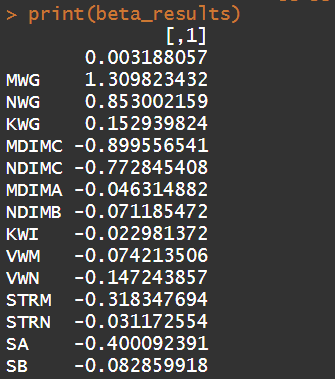
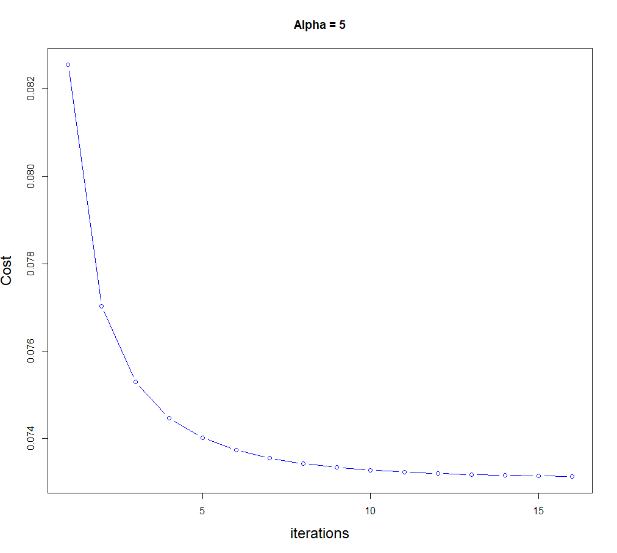
a)We need homoscedasticity for better prediction in linear regression. From the above plot of residuals vs predicted values we see that there is constant variance for most part of our data.

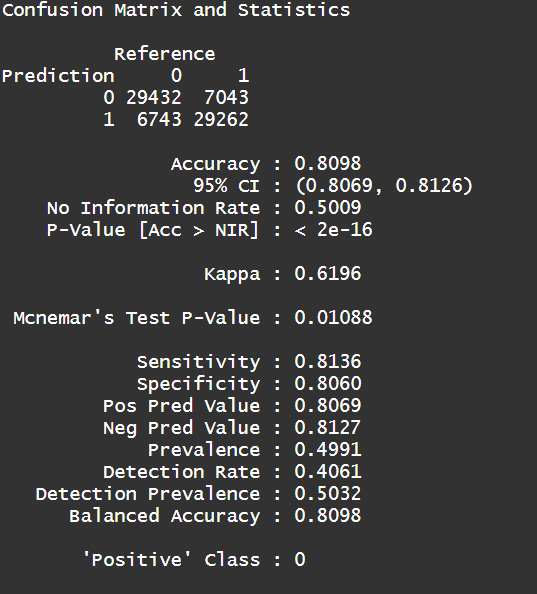
b)Q-Q plot ->We also need our errors to be normally distributed for better prediction in linear regression. We see here that only near the ends the errors are not normally distributed.

**Logistic Regression:**





Confusion Matrix for Alpha =5



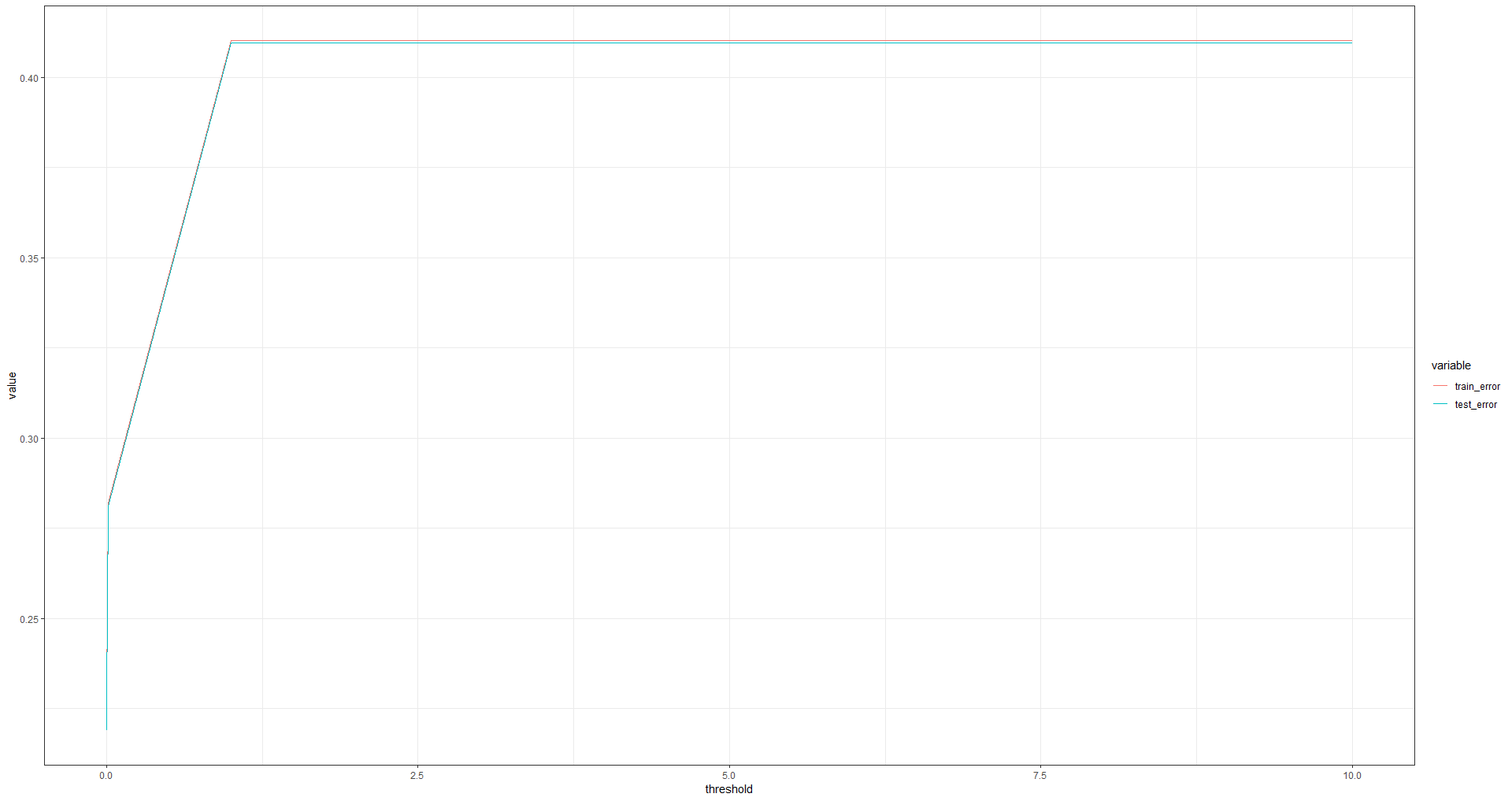
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Threshold=0.00001 |  |  |  |  |  |
| Alpha | Iterations | iterations taken to converge | MSE train | MSE Test | MSEdiff(train-test) |
| 0.001 | 10000 | 1366 | 0.1045083 | 0.1044226 | 8.57E-05 |
| 0.01 | 10000 | 873 | 0.08090957 | 0.08067033 | 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:**

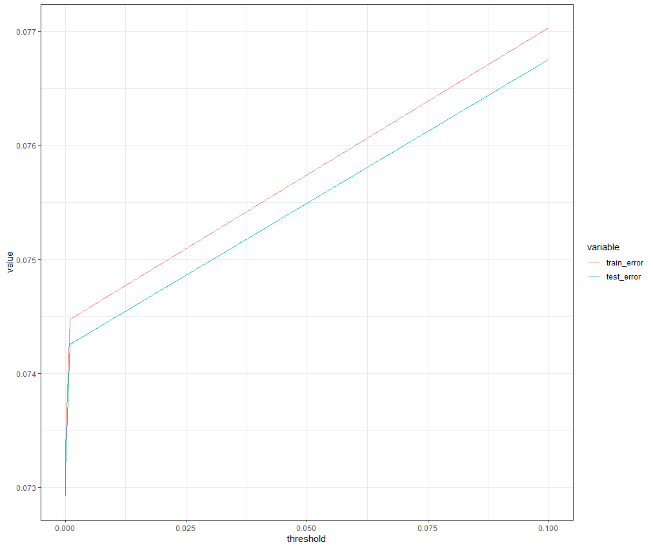
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Alpha = 0.1 |  |  |  |  |  |
| Threshold | Iterations | iterations taken to converge | MSE train | MSE Test | MSEdiff(train-test) |
| 10 (Not Converging) | 10000 | 2 | 0.4103538 | 0.4097237 | 0.0006301 |
| 1 (Not Converging) | 10000 | 2 | 0.4103538 | 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 |



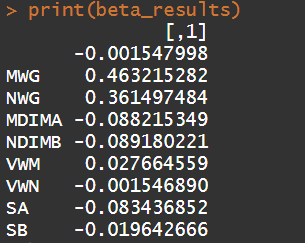
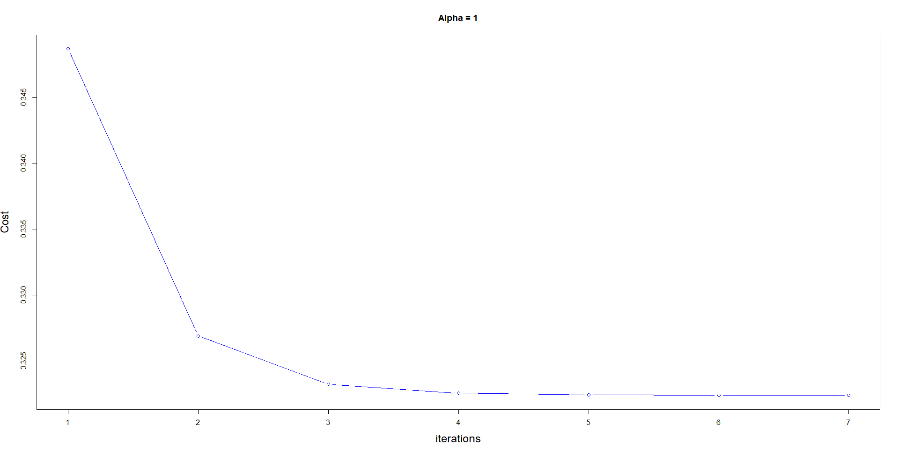
From this experiment the error was increasing with increasing the threshold for a fixed alpha.

**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:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |

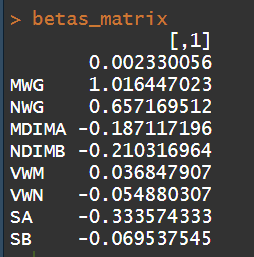
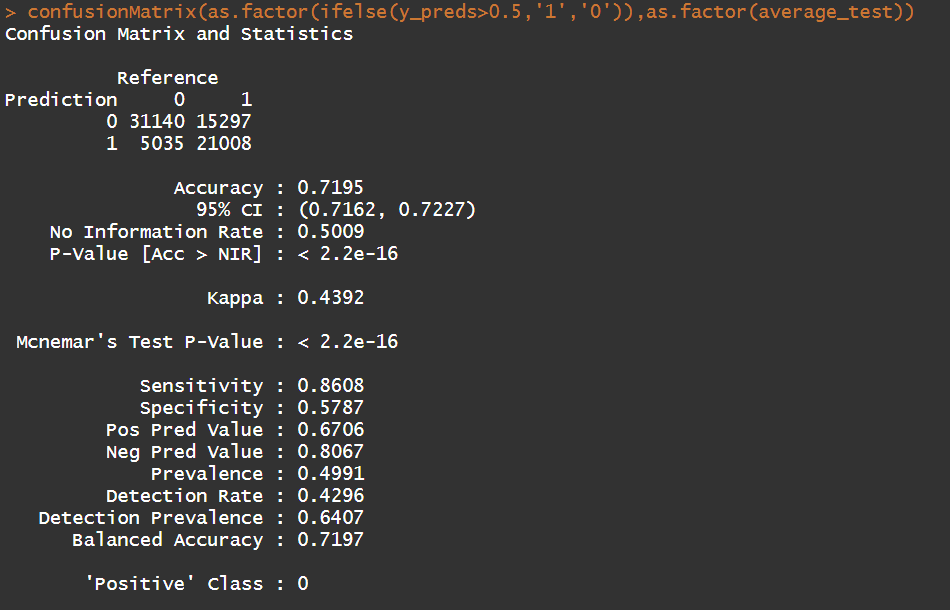
The Random features equation with threshold as 0.00001 and alpha=1

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 |



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.