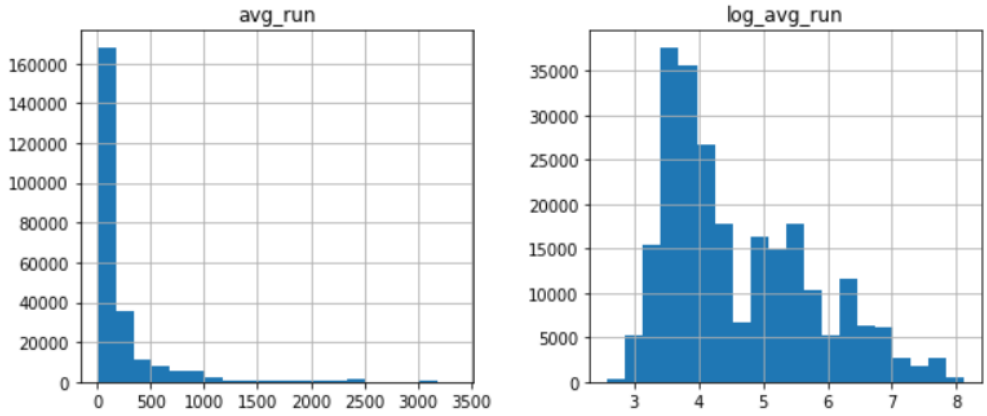
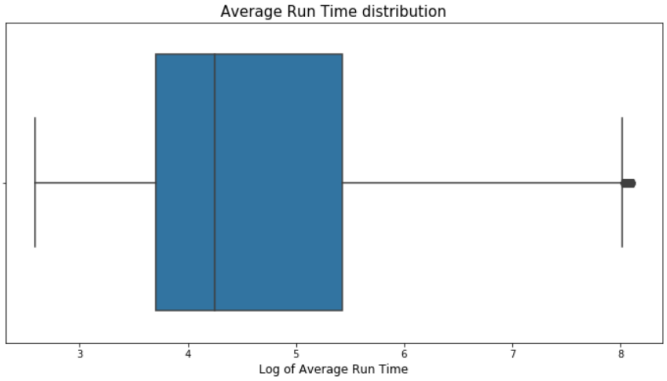
Linear Regression and Logistic Regression Report

Data Description & Preparation

- The dataset has 241600 observations with 18 features.
- None of the column contains any missing value, so no missing value imputation is required.
- The target variable will be the average of the four run times.
- After computing the average, the data in the target variable is right skewed. Taking the log of the values reduces the skewness of the target variable.



- Checking for the outliers in the target variable gives certain observations. Therefore, we remove these 480 records before running our models.



- The data is then scaled appropriately.

Linear Regression

Functions

- Linear regression uses the squared error cost function, which must be minimized to find the optimal beta values for prediction.

- This algorithm uses the batch gradient descent function to reduce the cost function.

Experimentations

The dataset is divided into training and test datasets by 80:20 ratio. The value of the cost function is 0.4995 with initial beta parameter values as all zeroes.

The linear regression function for the dataset is:

Target variable = β0 + β1(MWG) + β2(NWG) + β3(KWG) + β4(MDIMC) + β5(NDIMC) + β6(MDIMA) + β7(NDIMB) + β8(KWI) + β9(VWM) + β10(VWN) + β11(STRM) + β12(STRN) + β13(SA) + β14(SB)

Target variable = 4.152e-04 + 5.020e-01 (MWG) + 3.962e-01 (NWG) + 8.371e-02 (KWG) + -3.957e-01 (MDIMC) + -3.807e-01 (NDIMC) + 8.806e-04 (MDIMA) + -1.311e-03 (NDIMB) + -1.353e-02 (KWI) + -1.586e-02 (VWM) + -4.164e-02 (VWN) + -6.064e-02 (STRM) + -7.905e-03 (STRN) + -8.842e-02 (SA) + -2.434e-02 (SB)

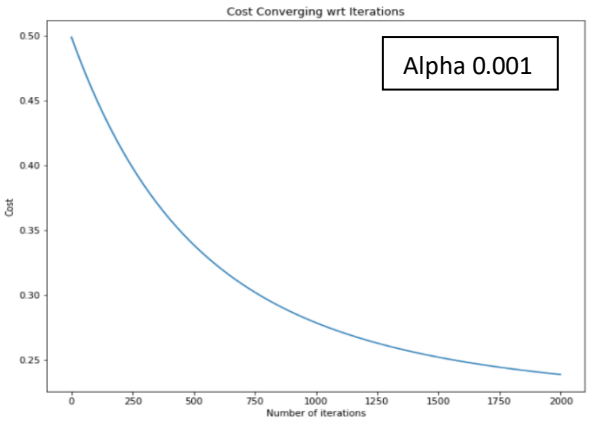
**Note the above beta coefficients are calculated with alpha 0.05 and 2000 iterations on a scaled dataset*

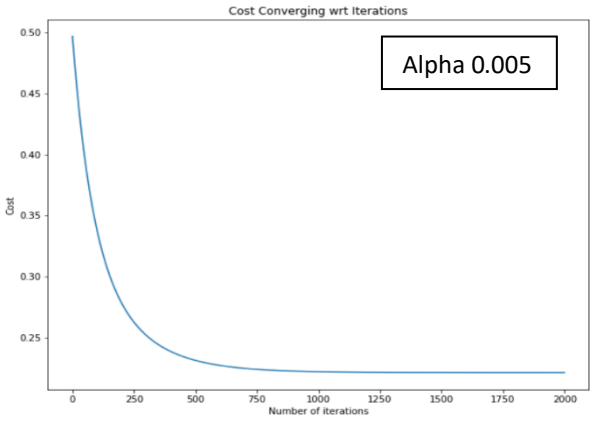
Experiment 1

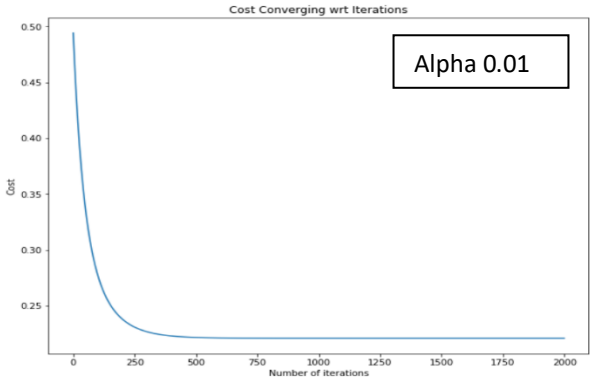
The first experiment is to vary the value of the learning rate alpha. Alpha determines how fast the algorithm learns and converges the cost function towards its minimum value.

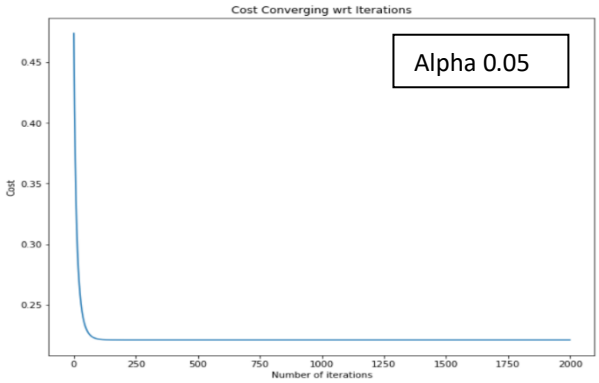
Below are the graphs for Cost function converging as a function of number of iterations for different values of alpha.

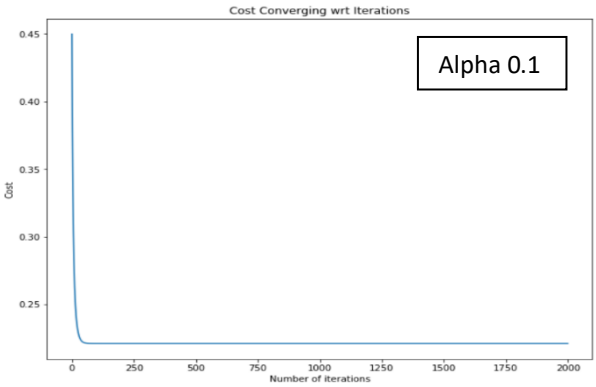
**Note the below graphs are made with 2000 iterations and 1e-07 convergence threshold on a scaled dataset*

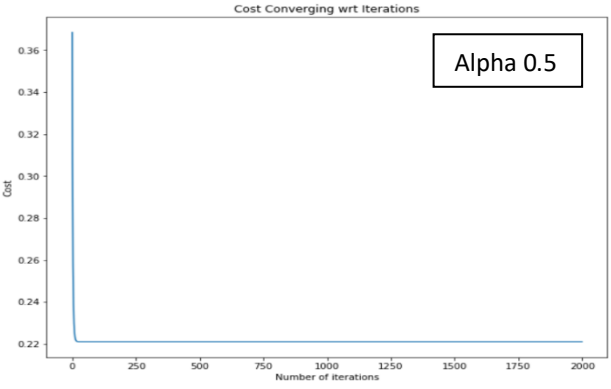


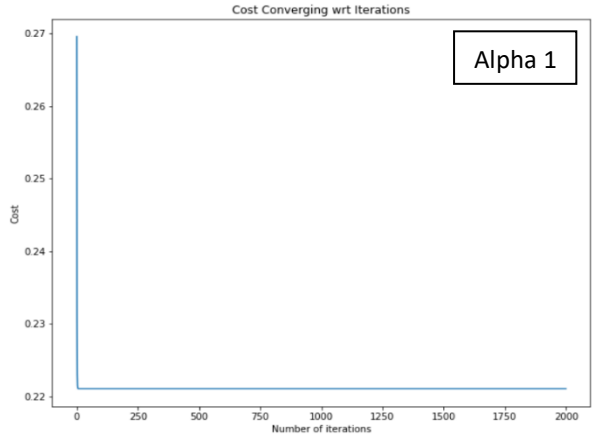


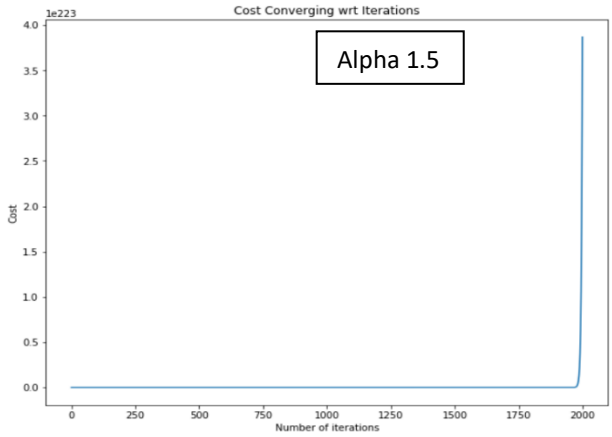












Alpha	Cost function after 2000 iterations	Converging iteration	Test Data MSE	Test Data r-squared value
0.001	0.2384681544	Not found	0.47767	0.52423
0.005	0.2210595920	1698	0.44281	0.55895
0.01	0.2210556121	914	0.44280	0.55896
0.05	0.2210556119	212	0.44280	0.55896
0.1	0.2210556119	111	0.44280	0.55896
0.3	0.2210556119	38	0.44280	0.55896
0.5	0.2210556119	23	0.44280	0.55896

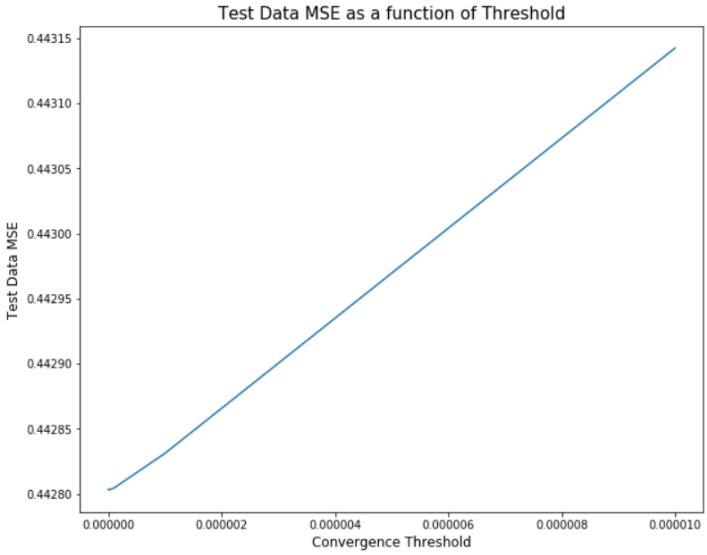
1	0.2210556119	10	0.44280	0.55896
1.3	0.2210556119	37	0.44280	0.55896
1.5	3.867537e+223	Not found	7.7352e+223	-7.70445

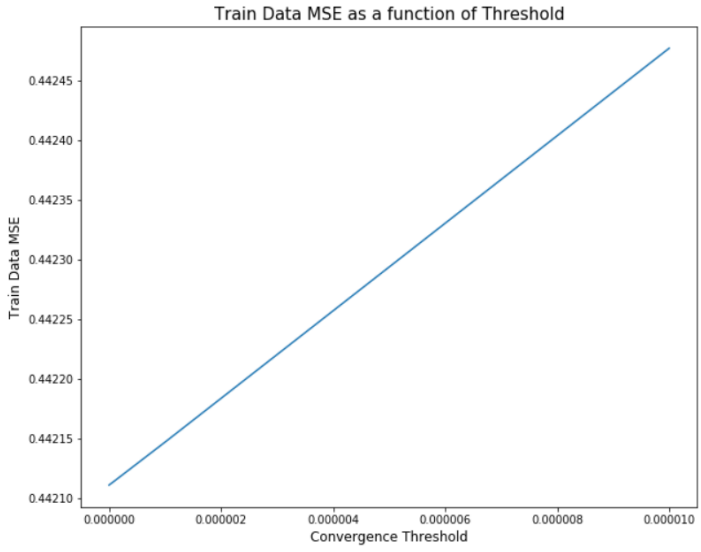
Based on the above graphs and the table, the **alpha values in the range of 0.05 – 0.1 are the most optimal**:

- For values of alpha too low (0.001 and 0.005) the algorithm takes many iterations to converge and is adding a lot of computational time.
- The value of alpha 0.05 is converging at iteration number 212 which is 77% times faster than that of alpha 0.01.
- The large values of alpha are making the algorithm converge too fast and therefore, they should not be used because they can miss the minima value and can bounce back (such as in case of alpha 1.5)

Experiment 2

The second experiment is varying the value of convergence threshold. This threshold decides where the gradient descent iterations should stop once the required convergence has been achieved. It is a smarter way to reduce the computing time of the gradient descent than blindly giving a very large number of iterations to run.



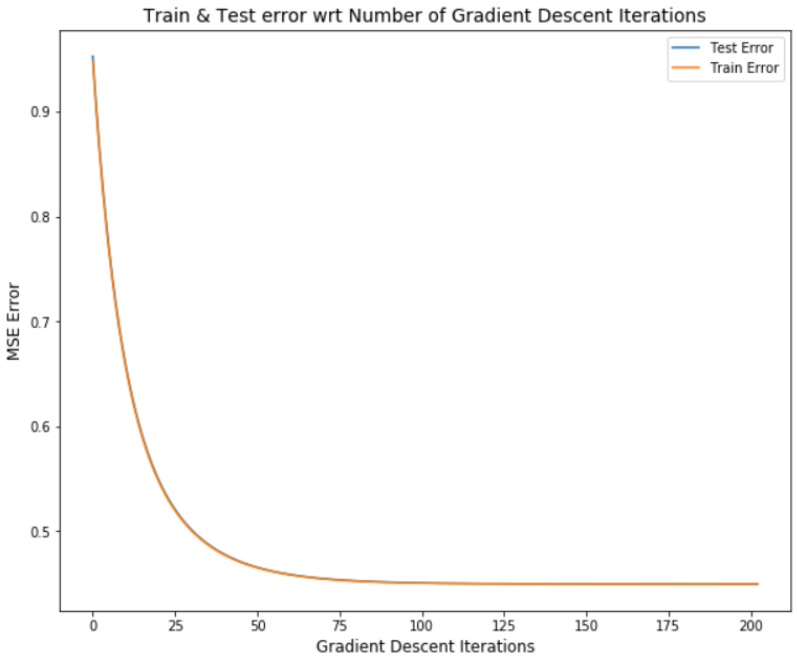


**Note the above graphs are made with optimal alpha 0.05*

The graphs show that the MSE for both test and train datasets decrease with decrease in convergence threshold. This is because low thresholds lead to a greater number of iterations and hence the cost function is reduced further which gives better predictions.

However, we notice that the maximum decrease in the MSE values is when the threshold is decreased from 1e-05 to 1e-06, and the MSE decreases further with threshold 1e-07. Below 1e-07 there is negligible difference in the error values.

Therefore, the optimal convergence threshold is 1e-07



The train and test data MSEs follow approximately the same graphical line when plotted against the number of gradient descent iterations for convergence threshold of 1e-07 and alpha of 0.05.

Experiment 3

The third experiment is selecting 8 independent features at random and running the linear regression model using only those features. The 8 randomly selected variables are: 'NDIMB', 'STRM', 'MDIMC', 'VWN', 'KWG', 'KWI', 'VWM' and 'MWG'.

As expected, because little care is given while selecting the variables, the MSE of the model has increased as compared to using all the features.

	Train MSE	**Test MSE**
All Features	0.44211	0.44280
8 Randomly selected Features	0.67373	0.67633

**Note the above MSEs are reported by using the optimal values of alpha = 0.05 and 2000 iterations*

Experiment 4

The fourth experiment is selecting 8 independent features which are best suited to predict the output and running the linear regression model using only those features.

We will remove the features which are least correlated to the target variable. These features with their correlation to the target variable are STRN 0.007790, KWI 0.011200, SB 0.020323,

KWG 0.020358, MDIMA 0.023076 and NDIMB 0.033688. Therefore, we will run the model using 'MWG', 'NWG', 'MDIMC', 'NDIMC', 'VWM', 'VWN', 'STRM' and 'SA' features.

	Train MSE	Test MSE
All Features	0.44211	0.44280
8 Randomly selected Features	0.67373	0.67633
8 Carefully Selected Features	0.44957	0.44971

**Note the above MSEs are reported by using the optimal values of alpha = 0.05 and 2000 iterations*

We can clearly see that the MSE has decreased significantly by carefully selecting features as compared to randomly selecting them. This is because we dropped some features in random selection like 'NWG', which is highly correlated to the target variable.

The accuracy is comparable to that of running the model on all features. This shows that those six features have negligible effect on the target variable and can be dropped from the model. Selecting the variables also reduces the time of computation.

Logistic Regression

Functions

- Logistic regression uses a logarithm-based cost function, which heavily penalizes incorrect classification.
- This algorithm uses the batch gradient descent function to reduce the cost function.
- The predicted values of the target variable are calculated by using the sigmoid function which gives the values between 0 and 1 and can be interpreted as probabilities.
- The class of the target variable is predicted by carefully selecting a decision threshold value, which we will see later in the experimentation.

Creating Classes

The target variable (log of the average run times) is a continuous variable that must be converted into a categorical variable to run logistic regression on it.

The division into classes is done by taking the median value of the target variable as the decision criterion. The values less than or equal to the median are classified as 0, and the values above the median are classified as 1.

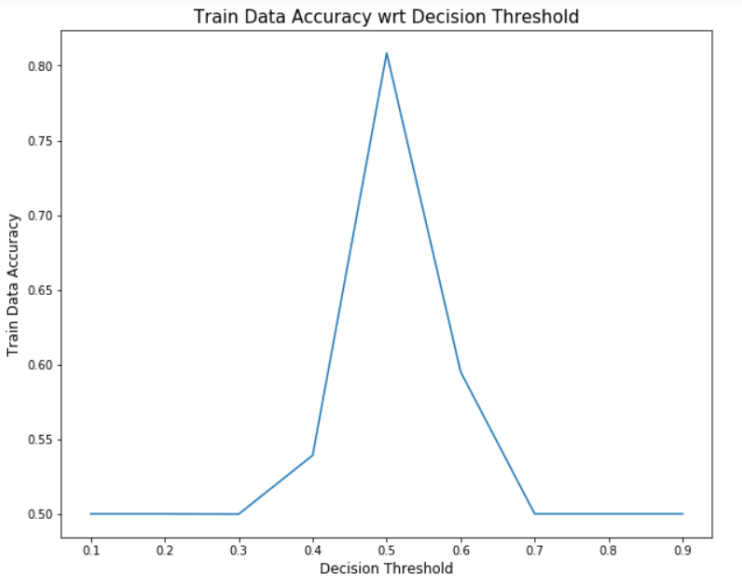
Threshold for classification	4.243123698247448
Class 0 records	120564
Class 1 records	120556
Total records = Class 0 + Class 1	241120

We have chosen the median to divide the target variable into classes because median is a better measure of central tendency to divide the dataset equally. The mean is affected by the outliers and extreme values.

Experimentation

Experiment 1

The first experiment is varying the value of decision threshold that decides the predicted class of the target variable based on the probabilities returned by the sigmoid function. The aim is to determine how the accuracy of the model increases/decreases by taking different thresholds.

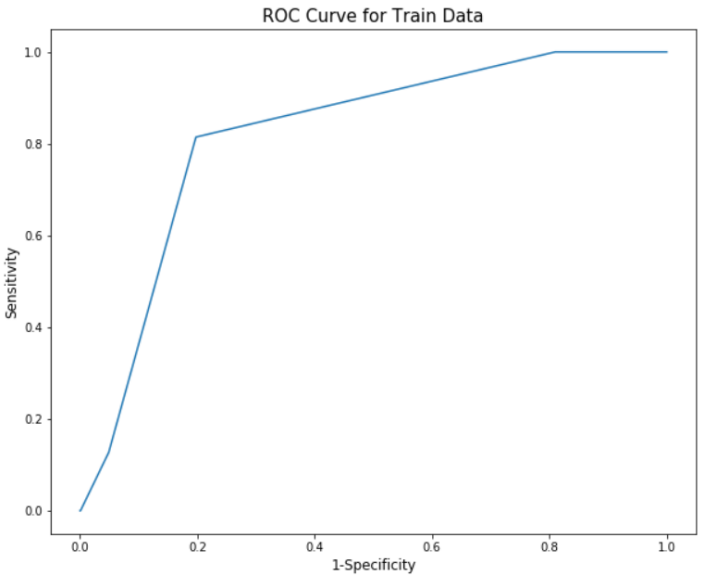


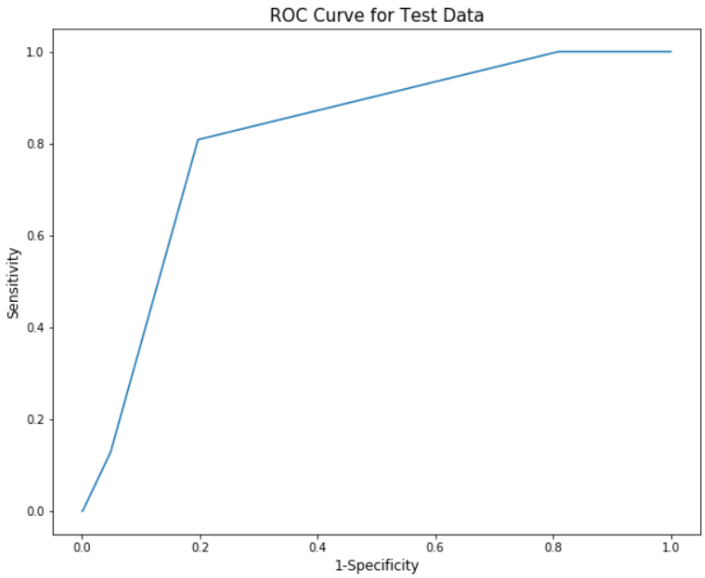
**Note the above Accuracy values are reported by using the optimal values of alpha = 0.1 and iterations = 500*

The accuracy of the model is maximum 0.8 at the decision threshold of 0.5 in both training and test data sets.

This is expected behavior because we divided our target variable into classes based on medina such that there were approximately equal number of records in both the classes 0 and 1. Increasing or decreasing the decision threshold from 0.5 decreases the accuracy.

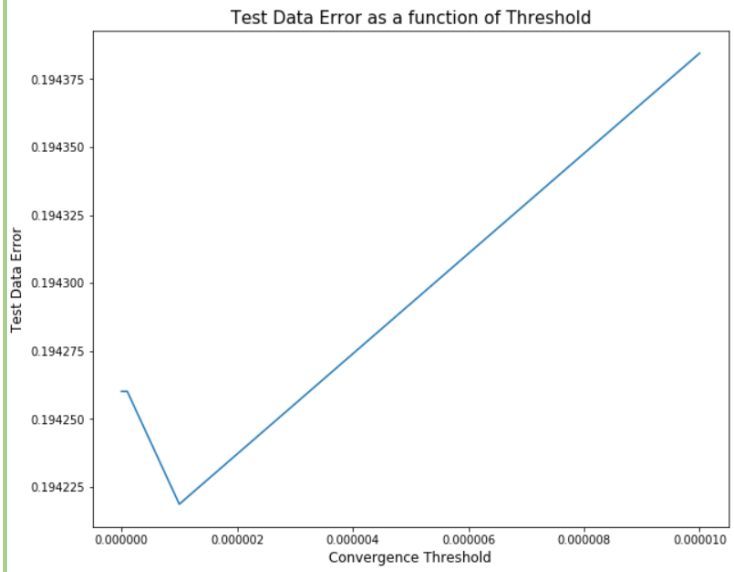
The ROC curves below show that the AUC is way more than 0.5 and that our logistic regression model is performing good.

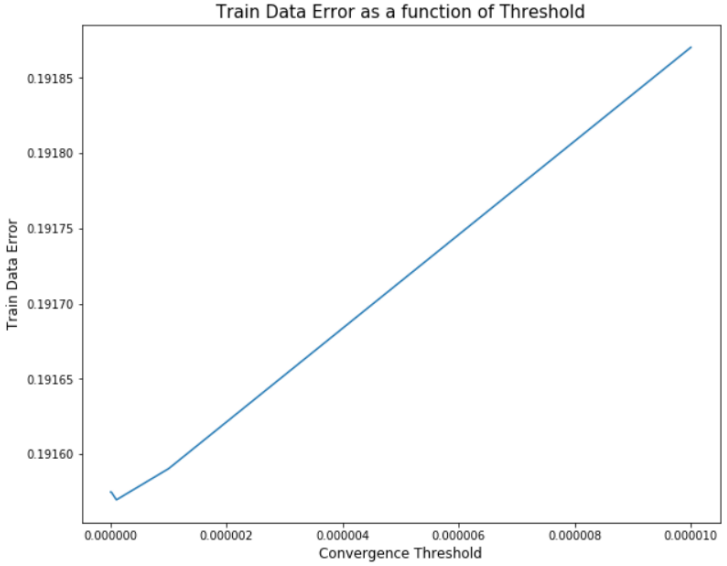




Experiment 2

The second experiment is varying the value of convergence threshold. As explained in the linear regression section, it is used to stop the iterations in gradient descent once the level of convergence is achieved.





**Note the above Errors are reported with optimal values of alpha = 0.1 and decision threshold = 0.5*

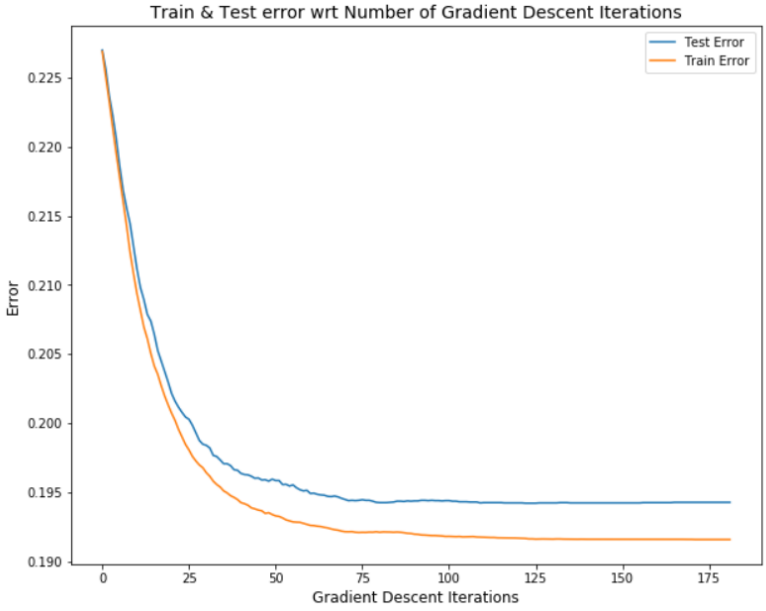
The graphs show that the prediction error initially decreases with decrease in threshold, but then increases by a very small amount for both train and test datasets with further decrease in threshold.

- More iterations are run when the threshold is less, which accounts for the decrease in cost function value when threshold is decreased to 1e-06 and 1e-07 from 1e-05.
- Below the convergence threshold of 1e-07 the improvement in cost function is very small and there is negligible change in the test dataset error.

Convergence Threshold	Cost Function	# Iteration	%Improvement in Cost Function	Test Data Error
1e-05	0.615831	95		0.19438
1e-06	0.615669	138	0.02	0.19422
1e-07	0.615652	182	0.002	0.19426
1e-08	0.615651	225	0.0001	0.19426
1e-09	0.615651	268	0.00	0.19426
1e-10	0.615651	312	0.00	0.19426
1e-11	0.615651	355	0.00	0.19426
1e-12	0.615651	399	0.00	0.19426

**Note the above metrics are reported with optimal values of alpha = 0.1 and decision threshold = 0.5*

Therefore, by looking at these results the **optimal convergence threshold seems to be 1e-07**. The cost function at this convergence threshold is sufficiently low to give the minimum error Also, the number of iterations required to converge at 1e-07 are way less than those of smaller convergence thresholds.



The plot of train and test errors with respect to the number of gradient descent iterations at convergence threshold 1e-07 shows that the algorithm just needs approximately 180 iterations to converge at alpha of 0.1

Experiment 3 & 4

The third experiment is selecting 8 independent features at random and running the logistic regression model using only those features. The 8 randomly selected variables are: 'MWG', 'NWG', 'MDIMC', 'NDIMC', 'VWM', 'VWN', 'STRM' and 'SA'

The fourth experiment is selecting 8 independent features which are best suited to predict the output. As explained in linear regression, the 8 selected variables are: 'MWG', 'NWG', 'MDIMC', 'NDIMC', 'VWM', 'VWN', 'STRM' and 'SA'

Similar to linear regression, the accuracy of the model with randomly selected features has decreased. When we are carefully selecting the features, the accuracy is better than randomly selecting features and comparable to using all the features.

	Train Accuracy	**Test Accuracy**
All Features	0.808425	0.805739
8 Randomly Selected Features	0.656177	0.658842
8 Carefully Selected Features	0.808197	0.805905

**Note the above accuracy levels are reported by using the optimal values of alpha = 0.1, decision threshold = 0.5 and iterations = 500*

Discussion

The linear model we formulated has a test MSE of approximately 0.44, which is very close to the train dataset MSE. The logistic regression model is giving an accuracy of 0.81 in test dataset as well as train dataset. We achieved these MSE/accuracy values by using all features and also by carefully selecting 8 independent features. Therefore, we can say that our models are performing good.

The GPU run time is most dependent on the feature 'MWG' because it has the highest correlation of 0.46 with the target variable and also the highest beta coefficient (for linear model) of 0.502

In addition to 'MWG', the features which matter the most for predicting the GPU run time, based on beta values and correlation are NWG, MDIMC and NDIMC.

Random selection of variables does not lead to an improvement in the cost function, adding all the variables was better than adding just a random few.

Improvements: The R-squared value in linear model is 0.56. This is quite low for linear models. The correct modelling algorithm for this dataset might be non-linear in nature. It would be interesting to try other non-linear models and check their performance.

The first 10 independent features are all ordinal in nature. We treated them as continuous variables for the scope of this assignment. The performance of the model might improve by weighting and encoding the values in these columns.

We can also further drop the features VWN and VWM from the model of 8 carefully selected features. These two features are significantly correlated to NWG and MWG features. Removing these features will decrease multicollinearity and might give better results.