Applied Machine Learning Assignment 1 **MUKIL RAJ KANNAN(mxk180040)**

Dataset description and summary statistics:

The dataset for implementing the gradient descent algorithm for linear and logistic regression is downloaded from UCI Machine Learning repository. The dataset name is “sgemm_product.csv” . The details about the dataset is mentioned below :

➔ The dataset has 241600 records with 18 features

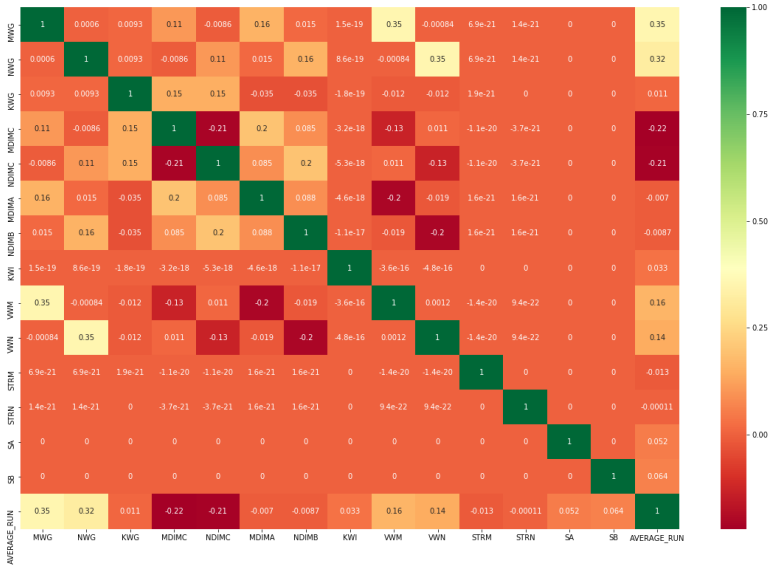
➔ The dependent variable is average of four runs (in ms)

➔ This is a clean dataset as there are no missing values in the dataset in any of the columns

Correlation :

Correlation is a term that quantifies the degree to which two variables are related to each other . We can try to find the correlation between the independent variables by using the correlation plot. When variables are highly correlated, they can introduce instability in the errors on the betas and are, in fact, measuring the same thing . So, the simple solution, assuming you cannot greatly increase sample size, is to throw out all but one of any group of variables that is highly correlated. Since our data doesn’t have any highly correlated data, we can use all 15 variables.

Correlation matrix for all 15 columns:



TASK 1,2 and 3: First , we scaled the data using the StandardScaler() built in function. We scaled all the 16 columns . Secondly , we split the dataset (with 16 columns) into training and test for a ratio of 0.7 : 0.3 . Then we create the X_train, X_test, y_train and y_test data for our training and test set. The regression equation for the Average Run is given below :

Average Run = β0 + βMWG ∗ MWG + βNWG ∗ NWG + βKWG ∗ KWG+ βMDIMC ∗ MDIMC + βNDIMC∗ NDIMC + βMDIMA ∗ MDIMA + βNDIMB ∗ NDIMB + βKWI ∗ KWI + βVWM ∗ VWM + βVWN ∗ VWN +

βSTRM ∗ STRM + βSTRN ∗ STRN + βSA ∗ SA + βSB ∗ SB. The initial parameter values from the implementation of the linear regression model to the above model equation is given below . I also ran Linear Regression() function to check if the coefficients values are matching and they match perfectly with my Gradient descent values.

Coefficients found using Gradient Descent & LinearRegression():

Coefficients	β0	βMWG	βNWG	βKWG	βMDIMC	βNDIMC	βMDIMA	βNDIMB
Values from GD	-0.0017	0.383242	0.352695	0.109868	-0.35456	-0.34691	0.025415	0.026148
Values from LR()	-0.0017	0.383242	0.352695	0.109868	0.354563	-0.34691	0.025415	0.026148
Coefficients	βKWI	βVWM	βVWN	βSTRM	βSTRN	βSA	βSB	
Values from GD	0.031913	-0.00875	-0.01734	-0.0138	0.001807	0.051191	0.063083	
Values from LR()	0.031913	-0.00875	-0.01734	-0.0138	0.001807	0.051191	0.063083	

Experimentation and Results

Mainly 4 experimentation was undertaken using the dataset in which we explored the effect of changing the hyperparameters of the algorithm and discuss the effectiveness of feature selection in predictive accuracy. These experiments were repeated both for linear and logistic regression. Finally, the results of the experiments were discussed in the results section.

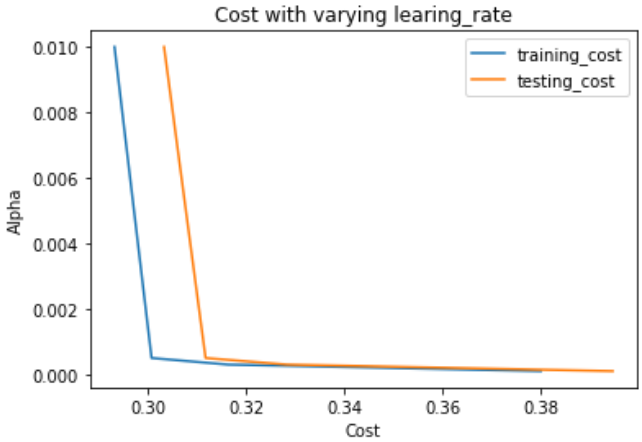
Experimentation:

1. **Hyperparameter Tuning**

EXPERIMENT 1 – PART A : We are going to change the alpha values for running the Gradient Descent function. The values that I have chosen are 0.0001,0.0003,0.0005,0.01.

Iterations =5000				
Cost found before running GD=0.49465903977303277				
Alpha	0.0001	0.0003	0.0005	0.01
Cost Training Set	0.379961394	0.316324747	0.300795	0.293253
Cost Test Set	0.39458946	0.328274824	0.31178	0.303331

Graph for Training and Test set cost with varying learning rate:



Conclusion : When we checked the cost for other alpha values , we could see that the value 0.01 has the least/minimized cost/error in both training and test set. Also , the number of iterations to attain is also significantly less when compared to other alpha values . It is also evident that the cost error for test set is always high compared with training set error.

EXPERIMENT 1 - PART B : For logistic regression , I found that the median is 69.79. The distribution for Average run is skewed, therefore I chose median as a threshold to assign the binary classification of 0 and 1. There are 120801 values below 69.79 and 120799 values above 69.79.

Coefficients found using Gradient Descent & LinearRegression():

Coefficients	β0	βMWG	βNWG	βKWG	βMDIMC	βNDIMC	βMDIMA	βNDIMB
Values from GD	0.009039	1.318639	0.848658	0.153623	-0.89816	-0.76339	-0.05116	-0.06745
Values from LogR()	0.004515	1.31847	0.848542	0.15358	-0.89801	-0.76326	-0.05114	-0.06744
Coefficients	βKWI	βVWM	βVWN	βSTRM	βSTRN	βSA	βSB	
Values from GD	-0.01051	-0.07767	-0.14498	-0.32463	-0.03295	-0.40157	-0.08649	
Values from LogR()	-0.0105	-0.07762	-0.14492	-0.3246	-0.03295	-0.40153	-0.08648	

This is the initial cost for training set = 0.693127180759928 all β values as zero .Minimized cost after implementing the gradient descent with above found coefficients: 0.477286 for alpha = 0.1.

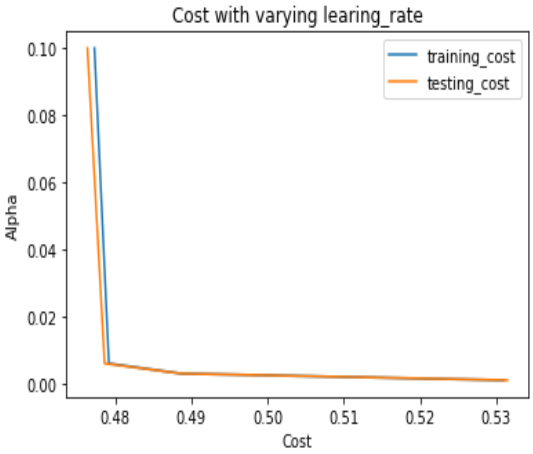
Iterations =5000				
Cost found before running GD=0.693127180759928				
Alpha	0.001	0.003	0.006	0.1
Cost Training Set	0.53098881	0.48864706	0.479181	0.477286
Cost Test Set	0.531457746	0.488545567	0.478614	0.476365

Accuracies for different values of alpha:-

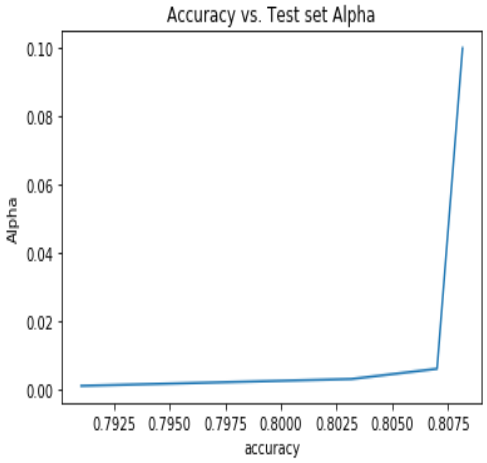
Alpha	Accuracy
0.001	0.7910182119205298
0.003	0.8031594922737307
0.006	0.807008830022075
0.1	0.8081539735099338

Conclusion : From the above plot we could see that the accuracy is highest in alpha = 0.1 for which we have minimized cost when compared to other alpha values. But for other values of alpha , it is little less. This might be because of varying the alpha that, the prediction efficiency is high by reducing the alpha.

Graph for Training and Test set cost with varying learning rate:



Plot for Alpha vs Accuracy:-



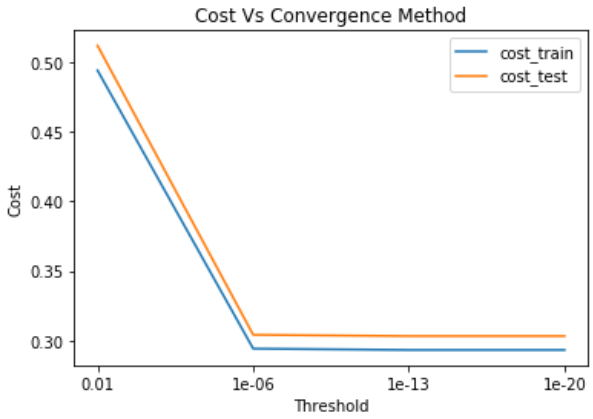
2. Changing Convergence Method:

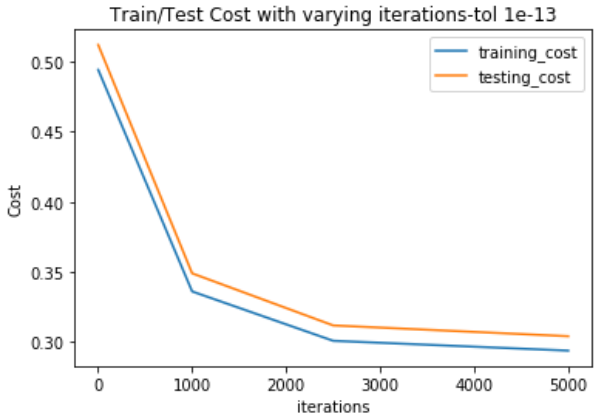
When the Change in Cost is within the Convergence threshold, the gradient descent algorithm is said to be converged. So, the algorithm converges faster at higher threshold value but at a certain point increasing threshold is found to increase the cost in both train and test dataset. We find the optimal value of threshold by plotting cost as a function of convergence threshold and find the point at which the test error is minimum.

EXPERIMENT 2 -Part 1 : For this experiment with linear, we are going to pass different values of threshold for evaluating the cost values convergence with respect to iterations. We have taken alpha = 0.001 for this experiment.

The minimum test error is found for a tolerance of 1e-13.
The Train/Test cost is found to be exponentially decreasing at starting and nearly after ~2000 epoch, the learning is found to be slow with a tolerance of 1e-9 achieving at nearly 4000 iterations.
Part B: The same can be done with logistic, we are going to pass different values of threshold like [0.0001,0.000003,0.000001,0.0000001] for evaluating the cost values convergence with respect to iterations. We have taken alpha = 0.001 for this experiment.

Threshold	Cost Train Error	Cost Test Error
0.01	0.493916	0.511683
1.00E-06	0.294201	0.304147
1.00E-13	0.293253	0.303201
1.00E-20	0.293253	0.303201





Conclusion: Tuning convergence threshold improves accuracy of prediction and convergence threshold of minimum error for both train and test dataset can be different. The cost and learning of the algorithm exponentially decrease with iteration. After reaching the global minimum, increasing the iteration has no effect on threshold.

3. Random Feature Selection:

EXPERIMENT 3 – PART A : For this experiment , we are going to pick 8 random features from the dataset, and we are going to perform the same procedures that we did in Experiment 1. For the initial findings also, we are going to assign theta values as zero to find the cost without using the gradient descent . Then we are going to find the least cost/error by using GD function. The initial coefficients of all Betas are found by running the Gradient Function for alpha = 0.01.

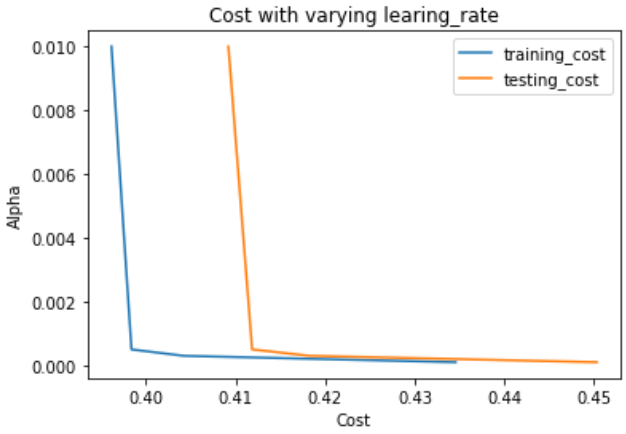
We are going to change the alpha values for running the Gradient Descent function. The values that I have chosen are 0.0001, 0.0003, 0.0005, 0.01.

Coefficients found using Gradient Descent:

Coefficients	β0	βKWG	βMDIMC	βNDIMC	βMDIMA	βNDIMB	βKWI	βVWM	βVWN
Values from GD	-0.00208	0.16648	-0.01209	-0.25972	0.001888	0.047281	0.064	-0.01908	0.354316

Iterations =5000				
Cost found before running GD= 0.49465903977303277				
Alpha	0.0001	0.0003	0.0005	0.01
Cost Training Set	0.43457	0.40425	0.398408	0.396174
Cost Test Set	0.450318	0.418281	0.411868	0.40922

Graph for Training and Test set cost with varying learning rate:



Conclusion : When we checked the cost for other alpha values , we could see that the value 0.01 has the least/minimized cost/error in both training and test set. Also , the number of iterations to attain is also significantly less when compared to other alpha values. The error/cost for the same alpha value 0.01 is a little high for the 8 random feature model than the 14-feature model . So , we can conclude that the features that we have taken for this experiment is not suitable in reducing cost when compared to the model in experiment 1. So , when compared to this Experiment , Experiment 1 model does well in minimizing cost.

Alpha=0.01	14 feature model	8 feature model
Cost Training Set	0.293253427	0.396174
Cost Test Set	0.303331045	0.40922

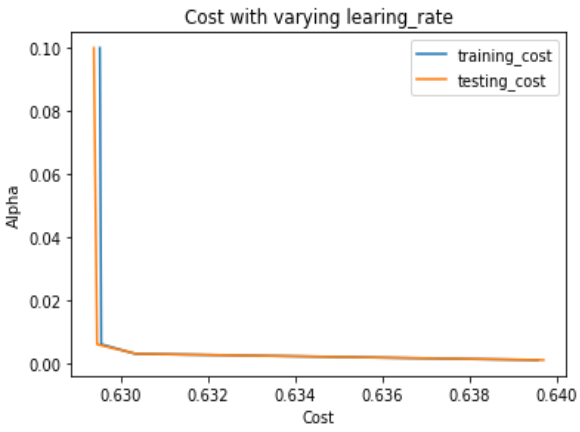
EXPERIMENT 3 – PART B : Calculate the accuracy for 8 feature model for varying alpha values 0.001,0.003,0.006,0.1 . For the mentioned alpha values , the accuracy is the highest for alpha value = 0.1 . At the end, we are going to compare the accuracies of this model of 14 features and previously built model in experiment 1.

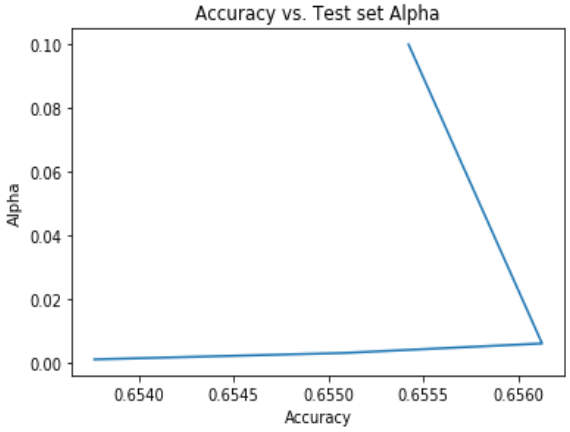
Coefficients found using Gradient Descent:

Coefficients	β0	βKWG	βMDIMC	βNDIMC	βMDIMA	βNDIMB	βKWI	βVWM	βVWN
Values from GD	0.01026	0.41444	-0.11429	-0.40477	-0.02145	0.006459	-0.06502	-0.10634	0.606308

Iterations =5000				
Cost found before running GD=0.693127180759928				
Alpha	0.001	0.003	0.006	0.1
Cost Training Set	0.639565	0.630323	0.629544	0.629512
Cost Test Set	0.639699	0.630346	0.629448	0.629374

Graph for Training and Test set cost with varying learning rate:





Accuracies for different values of alpha:-

Alpha	Accuracy for random 8 feature model
0.001	0.65376656
0.003	0.65507726
0.006	0.65612583
0.1	0.65542219

Alpha	Accuracy for 14 feature model	Accuracy for random 8 feature model
0.001	0.791018212	0.65376656
0.003	0.803159492	0.65507726
0.006	0.80700883	0.65612583
0.1	0.808153974	0.65542219

Conclusion: From this experiment of having 8 features , we could see that the highest accuracy is attained in alpha = 0.1 value for which we have minimized cost when compared to other alpha values. When we compare this with the previous model with 14 features in experiment 1, we could see that the accuracy in Experiment 1 is more than Experiment 3 for alpha = 0.1.

Alpha=0.01	14 feature model	8 feature model
Cost Training Set	0.477286	0.629512
Cost Test Set	0.476365	0.629374

The error/cost for the same alpha value 0.01 is a little high for the 8 random feature model than the 14-feature model . So , we can conclude that the features that we have taken for this experiment is not suitable in reducing cost when compared to the model in experiment 1. So , when compared to this Experiment , Experiment 1 model does well in minimizing cost.

4. Manual Feature Selection

EXPERIMENT 4 – PART A: For this experiment , we are going to pick 8 features from the dataset that are best suited to predict the output, and we are going to perform the same procedures that we did in Experiment 1. For the initial findings also, we are going to assign theta values as zero to find the cost

without using the gradient descent . Then we are going to find the least cost/error by using GD function. The initial coefficients of all Betas are found by running the GF for alpha = 0.01.

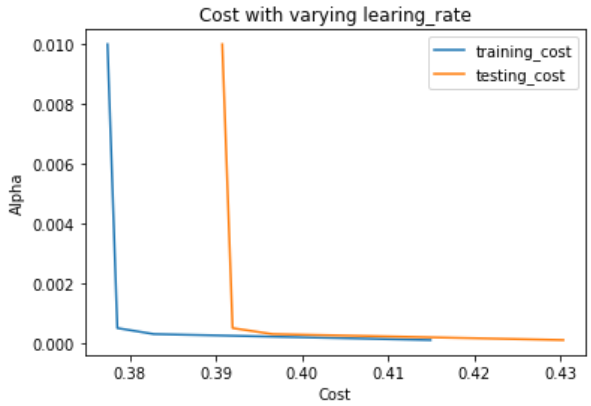
Coefficients	β0	βMWG	βNWG	βKWG	βKWI	βVWM	βVWN	βSA	βSB
Values from GD	-0.00187	0.334878	0.306374	0.005558	0.030344	0.044226	0.034556	0.052115	0.063082

We are going to change the alpha values for running the Gradient Descent function. The values that I have chosen are 0.0001,0.0003,0.0005,0.01.

Iterations =5000				
Cost found before running GD= 0.49465903977303277				
Alpha	0.0001	0.0003	0.0005	0.01
Cost Training Set	0.41494135	0.38279754	0.378542653	0.377402799
Cost Test Set	0.43037325	0.396565887	0.391937697	0.390739812

This is the initial cost for training set = 0.4946590397 all β values as zero .Minimized (global minima) cost after implementing the gradient descent with above found coefficients: 0.377402799 for alpha = 0.01.

Graph for Training and Test set cost with varying learning rate:



Conclusion : When we checked the cost for other alpha values , we could see that the value 0.01 has the least/minimized cost/error in both training and test set. Also , the number of iterations to attain is also significantly less when compared to other alpha values for this model with 8 best features.

Alpha=0.01	Random 8 feature model	Selected 8 feature model
Cost Training Set	0.396174	0.377402799
Cost Test Set	0.40922	0.390739812

The error/cost for the same alpha value 0.01 for the 8 selected feature model is lower than the 8 random feature model . So , we can conclude that the features that we have taken for this experiment is best suited to predict the output reducing cost when compared to the model in experiment 3.

EXPERIMENT 4 – PART B : We are going to pick 8 features from the dataset that are best suited to predict the output, and we are going to perform the same procedures that we did in Experiment 1. For the initial findings also, we are going to assign theta values as zero to find the cost without using the gradient descent . Then we are going to find the least cost/error by using GD function. The initial coefficients of all Betas are found by running the Gradient Function for alpha = 0.01.

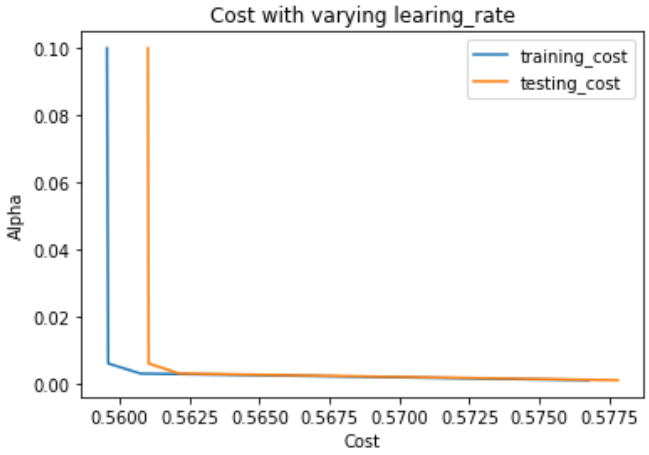
This is the initial cost for training set = 0.693127180759928 all β values as zero. Minimized cost after implementing the gradient descent with above found coefficients: 0.55955295 for alpha = 0.1.

Iterations =5000				
Cost found before running GD=0.693127180759928				
Alpha	0.001	0.003	0.006	0.1
Cost Training Set	0.57676253	0.56074446	0.55959482	0.55955295
Cost Test Set	0.57780782	0.562128603	0.561041388	0.561012237

Alpha=0.01	Random 8 feature model	Selected 8 feature model
Cost Training Set	0.629512	0.55955295
Cost Test Set	0.629374	0.561012237

Alpha	Accuracy for random 8 feature model	Accuracy for selected 8 feature model
0.001	0.65376656	0.719936534
0.003	0.65507726	0.721840508
0.006	0.65612583	0.721578366
0.1	0.65542219	0.720033113

Graph for Training and Test set cost with varying learning rate:



Conclusion: From this experiment of having 8 features , we could see that the highest accuracy is attained in alpha = 0.003. The randomly selected features performed worse compared to manually selected 8 features and original set of features because random selection doesn't incorporate the domain knowledge of the user. Using domain knowledge to understand which features and functional

form are important for prediction is key for good learning of the algorithm. When we compare this with the previous model with 14 features in experiment 1, we could see that the accuracy in Experiment 1 is more than Experiment 4.The manually selected features performed worse than the original features because we are restricting important features from the model. Restricting important features causes the effect of such features to be shifted to the error term and therefore impacting the performance of the algorithm.

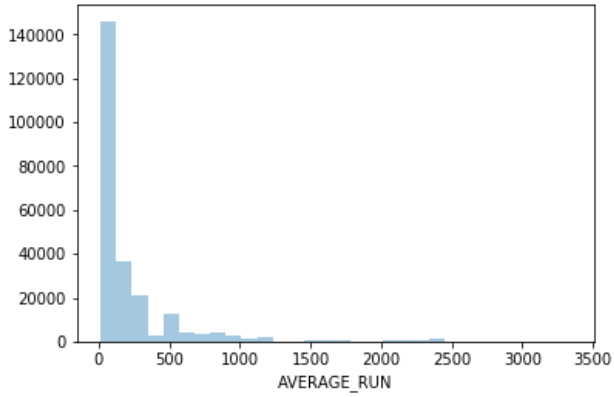
Results:

- Through these experiments we saw how the optimizing hyperparameters like learning rate and convergence threshold has a significant effect on minimizing the cost(global minimum) and thus attaining a high accuracy. For some values of learning rate and threshold, the time taken for the algorithm to fit the training set was extremely high. Therefore, we must balance between accuracy and computational limits while building an algorithm.

- By running the model against randomly selected features and handpicked features we came to know the importance of feature selection and engineering in building an ML model. The randomly picked feature performed worst, since the features ae selected at random without any domain knowledge on the data.

- The best predictors for predicting the GPU runtime in linear and logistic regression are MWG and NWG.

- We can use the correlation matrix to pick the top N feature from the dataset to help build our initial model. We could see different functions such as SelectKBest() and chi2 to let us know the best or top N features which will add more significance to the model that we are building rather than choosing the features by ourselves .



- As we can see from the distribution plot, the average run time spreads over till 3500 whereas the median average run time is 69. This clearly shows that values above 1500 can be considered as an outlier. The current model can be greatly improved by removing these outliers in order to get higher accuracy.