GRADIENT DESCENT REPORT

Username on miner: **Spydyyy**

Objective:

To encode an optimization (gradient descent) algorithm for learning a logistic regression model.

Analyzing Technical Results:

Gradient Descent Learning rate: 10^{-5}

Iterations = 10,000	**Accuracy**	**Classification error**	**Cross-entropy error**
Training Data	0.78	0.22	0.6098
Test Data	0.86	0.14 [estimated]	Time for Model: **997ms**

Iterations = 100,000	**Accuracy**	**Classification error**	**Cross-entropy error**
Training Data	0.80	0.20	0.4639
Test Data	0.86	0.14 [estimated]	Time for Model: **9.67s**

Iterations = 1M	**Accuracy**	**Classification error**	**Cross-entropy error**
Training Data	0.83	0.17	0.4261
Test Data	0.87	0.13 [estimated]	Time for Model: **1min 9s**

- By looking at the results generated by gradient descent algorithm with different iterations, we could clearly notice that the Cross-entropy error on the training data, gets gradually reduced as we increase the iterations, indicating that gradient descent is converging more effectively.
- There is decrease in the classification error of train data as we increase the iterations, whereas the test error seems to be the same because we are learning more about train data and minimizing training error, but still test data is unknown.
- The Classification error on the train data reduces as the Cross-entropy error on training data reduces.
- As the number of iterations increases, the runtime of the algorithm increases because we are trying to run the algorithm and continue to learn for longer time to reduce error. Apart from algorithm termination based on iteration, additional termination criteria added in this algorithm - (gradient<10^{-3}).

Logistic Regression using Scikit-learn

\-	Accuracy	Classification error	Time for Model:
Training Data	0.82	0.18	**399ms**
Previously without Using Scikit-learn			
Training Data	~ 0.79	~ 0.21	~ **1-10s**

Implementing logistic regression using machine-learning libraries such as scikit-learn fits the training set with high accuracy and speed than my gradient descent algorithm.

After Scaling Features in Train Data

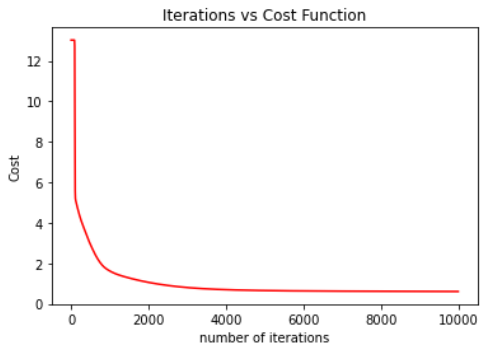
Learning Rate	Iterations for Tolerance (10^{-6})	Accuracy	Classification error	Final Cross-entropy error	Time for Model
10e-3	16,331	0.81	0.19	0.4147	1.59s
10e-4	163,338	0.81	0.19	0.4147	15.6s
10e-5	163,338	0.80	0.20	0.4147	1min 35s

So, by analyzing these results, we could clearly see the impact of Feature Scaling.

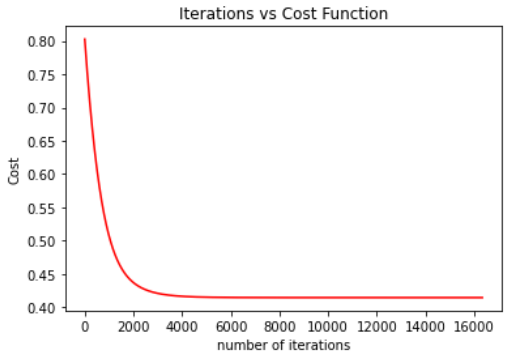
	Accuracy	Classification error
Before Scaling	~ 0.79	0.21
After Scaling	0.81	0.19

Now, it is approximately close to the same result we get by using scikit-learn library. Therefore, through these experiments, we can understand the convenience and efficiency of pre-existing machine-learning libraries such as scikit-learn, that scale features automatically and provides accurate results within very less time.

Before Scaling:



After Scaling:



Conclusion:

Therefore, by analyzing and examining the above results and points, I successfully implemented an optimization algorithm (Gradient Descent) for learning a logistic regression model and was able to efficiently test it on an unseen test dataset.