CLASSIFICATION USING A LOGISTIC REGRESSOR MODEL FOR A TWO-CLASS PROBLEM

Sriparna Chakraborty
Department of Computer Science
University of Buffalo
Buffalo, NY 14214
sriparna@buffalo.edu

ABSTRACT

The purpose of this experiment is to build a Logistic Regressor Model for a two-class classifier with multiple input features using Gradient Descent approach. I have used Wisconsin Diagnostic Breast Cancer (WDBC) dataset for training, validation and testing. The dataset contains 569 instances with 32 attributes (ID, diagnosis, 30 real-valued input features). The output will be to predict whether the FNA (Fine Needle Aspirate) cells belong to Benign (Class 0) or Malignant (Class 1). The training and validation accuracy have been compared with respect to the epochs and evaluated as a graph plot. The accuracy, precision and recall values for the testing dataset have been measured with the tuned hyper-parameters (weight, bias and learning rate) and the predicted output has been found to be around 93 percent accurate with respect to the actual output values.

1 INTRODUCTION

In machine learning and statistics, classification is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. In the terminology of machine learning, it is a type of Supervised Learning and a logistic regressor is used to determine its results. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes. In our experiment, we need a binary logistic model that has a dependent variable with two possible values: 0/1.

The WDBC dataset in our experiment has 569 instances and 32 attributes out of which around 80 percent of the instances have been identified as training set, 15 percent as testing set and the remaining 5 percent as validation set. Out of 32 attributes, only 30 real-valued attributes that have been computed from a digitized image of a fine needle aspirate (FNA) of a breast mass have been considered.

The input vector space X is of dimension 569x30 and the output vector space Y is of dimension 569x1. I have used scikit-learn libraries to perform the preprocessing tasks like data partitioning, data profiling and data normalization. The logistic regression algorithm has been written in Python from scratch using numpy methods and the graph plots have been implemented using matplotlib package. The dataset has been loaded in a Pandas dataframe for manipulation and processing tasks.

For each epoch, the training dataset is first split into batches of 50 datasets each and then Batch Gradient Descent algorithm is run for each batch to calculate the appropriate values of weights (w) and a bias(b) using the following formula:

$$w' = w - \eta \nabla_w f(w)$$

$$b' = b - \eta \nabla_b f(b)$$

where, η is the learning rate of the curve and ∇ is the derivative of the weight and bias functions.

The accuracy, precision and recall have been calculated from the confusion matrix for each epoch. The prediction has been improved for each epoch by tuning the hyper-parameters in a trial and error method.

The graph plots that have been used for evaluation of the model are:

a. Training Error and Validation Error VS Epoch
b. Training Accuracy and Validation Accuracy VS Epoch
c. Validation Accuracy VS Learning Rate

2 DATASET

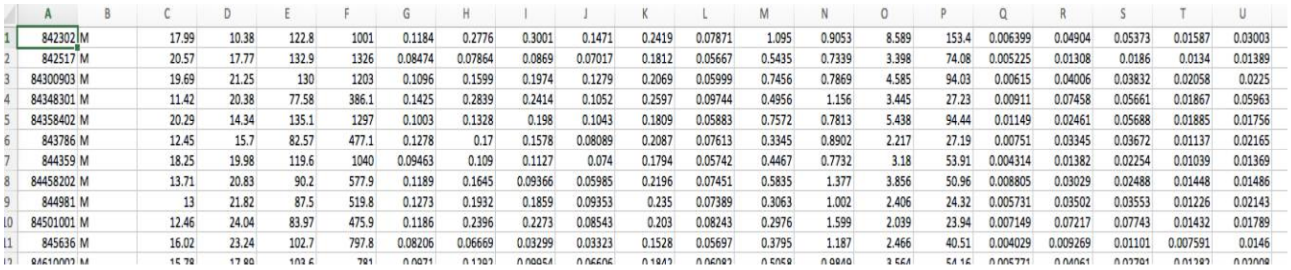
In the WDBC dataset used in the experiment, the input attributes have been computed for each digitized image of a fine needle aspirate (FNA) of a breast mass and the computed features describe the following characteristics of the cell nuclei present in the image:

1	radius (mean of distances from center to points on the perimeter)
2	texture (standard deviation of grey-cell values)
3	perimeter
4	area
5	smoothness (local variable in radius length)
6	compactness (perimeter2/area − 1.0)
7	concavity (severity of concave portions of the contour)
8	concave points (number of concave portions of the contour)
9	symmetry
10	fractal dimension (“coastline approximation” -1)

The mean, standard error and "worst" or largest of these features were computed for each image, resulting in 30 features.

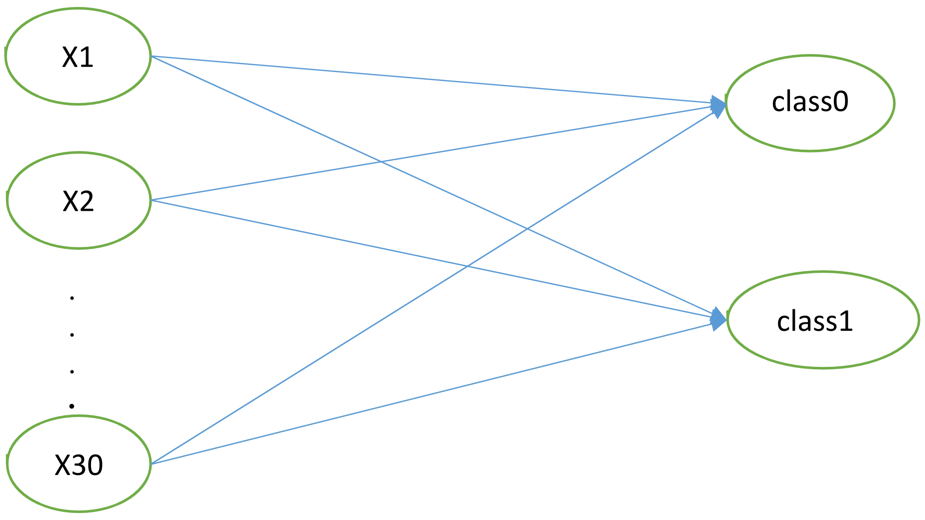
The dataset has been provided in .csv format and while preprocessing the dataset, I have used Pandas framework in Python to read the .csv file and load it in a dataframe.

An extract of the dataset has been presented below:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	1.095	0.9053	8.589	153.4	0.006399	0.04904	0.05373	0.01587	0.03003
2	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.0186	0.0134	0.01389
3	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	0.7456	0.7869	4.585	94.03	0.00615	0.04006	0.03832	0.02058	0.0225
4	84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	0.4956	1.156	3.445	27.23	0.00911	0.07458	0.05661	0.01867	0.05963
5	84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	0.7572	0.7813	5.438	94.44	0.01149	0.02461	0.05688	0.01885	0.01756
6	843786	M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	0.3345	0.8902	2.217	27.19	0.00751	0.03345	0.03672	0.01137	0.02165
7	844359	M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742	0.4467	0.7732	3.18	53.91	0.004314	0.01382	0.02254	0.01039	0.01369
8	84458202	M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451	0.5835	1.377	3.856	50.96	0.008805	0.03029	0.02488	0.01448	0.01486
9	844981	M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389	0.3063	1.002	2.406	24.32	0.005731	0.03502	0.03553	0.01226	0.02143
10	84501001	M	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243	0.2976	1.599	2.039	23.94	0.007149	0.07217	0.07743	0.01432	0.01789
11	845636	M	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697	0.3795	1.187	2.466	40.51	0.004029	0.009269	0.01101	0.007591	0.0146
12	84610002	M	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082	0.5058	0.9849	3.564	54.16	0.005771	0.04061	0.02791	0.01282	0.02008

As per my understanding, the first column is the ID column which uniquely identifies each row, the second column consists of the diagnosis values or the output values (M: Malignant and B: Benign) and the rest of the columns represent the 30 real-valued inputs.



The dataset from the.csv file has been imported in the Pandas framework in Python for easy manipulation of data. Each of the training, validation and testing input and output datasets has been reduced to a numpy matrix.

3 PREPROCESSING

Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn.

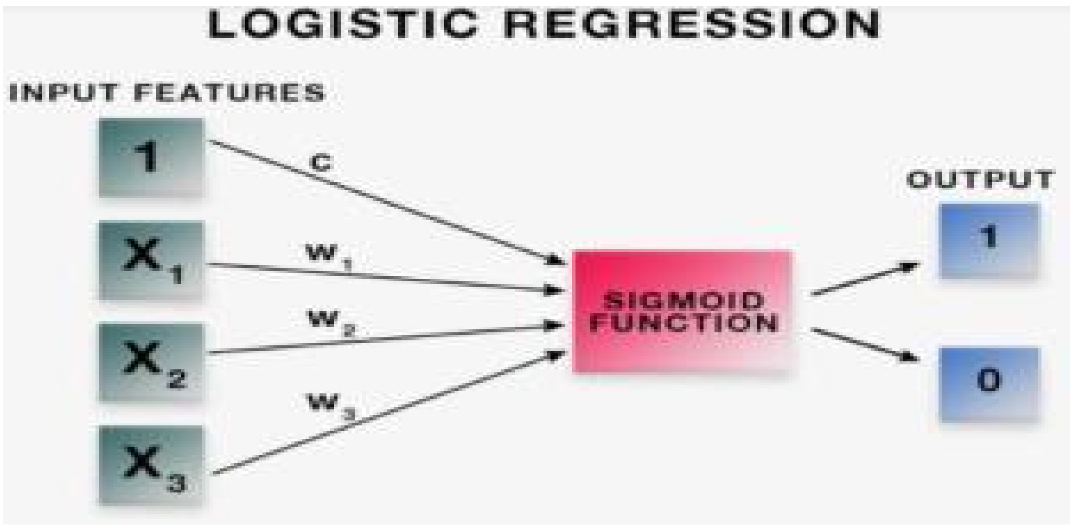
Therefore, it is extremely important that we preprocess our data before feeding it into our model.

The preprocessing steps that I performed in this experiment on the dataset are as follows:

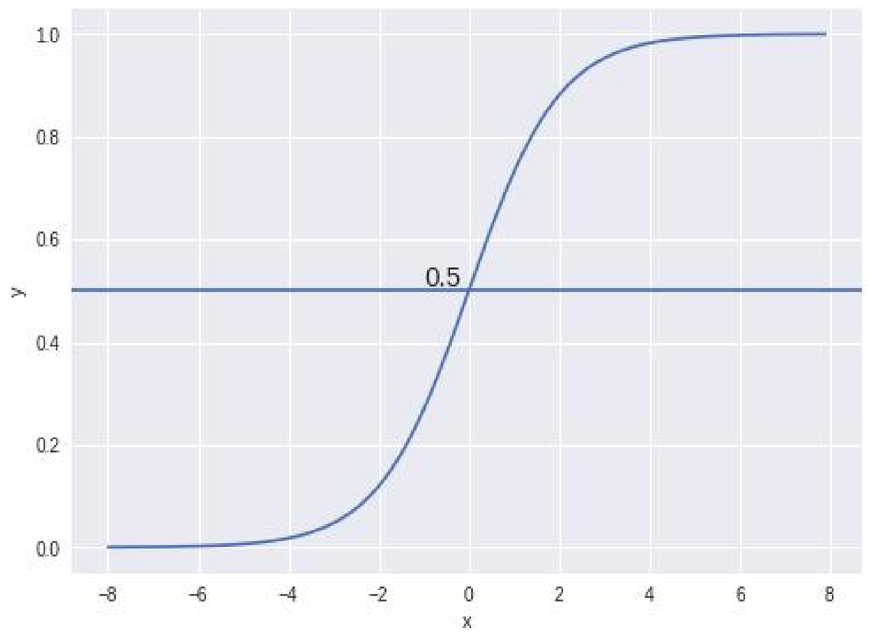
1. The ID column of the dataset has been dropped as it does not serve any purpose in our model construction. Hence, redundancy handling has been taken care of.
2. The second column of the dataset has been mapped to integer values/labels: 0 for B (Benign) and 1 for M (Malignant).
3. The remaining real-valued input features have been normalized using scikit-learn preprocessing library modules like Normalizer. Normalization of data is required when features have different ranges. The goal of normalization is to change the values of numeric columns in the dataset to a common scale without distorting differences in the range of values. The values in X required normalization as it has features that vary largely in ranges.
4. 80 percent of the dataset (X and Y) have been identified as the Training dataset (X_train and Y_train), 15 percent of the dataset have been identified as the Testing dataset (X_test and Y_test) and the remaining 5 percent of the dataset have been identified as the Validation dataset (X_val and Y_val). The partition task has been performed using train_test_split method of scikit-learn model_selection library.

4 ARCHITECTURE

The basic model for the Logistic regression classifier can be structured as follows:



In order to scale our real-valued continuous inputs and map our predicted values to probabilities, we use the Sigmoid function (also called the Activation function for Logistic regression). The function maps any real value into another value between 0 and 1.



For each epoch I calculated $Z = (W^T.X) + b$ for the input dataset X and pass it to $\sigma(Z) = \frac{1}{1+e^{-z}}$, where Z is the genesis function, W is the weight associated with the input dataset, b is a bias and, $\sigma(Z)$ is our sigmoid function. The dimensions of W are 30x1 and that of X is 569x30.

The current prediction function returns a probability score between 0 and 1. In order to map this to a discrete class I selected a threshold value or tipping point above which I classified values into class 1 and below which we classify values into class 0.

$$p \geq 0.5, \text{class}=1$$
$$p < 0.5, \text{class}=0$$

For every batch, I calculated the Gradient Descent and tune the values of W and b so that the difference of Cross Entropy function (Cost function denoting error in our prediction from actual output values) of our training and validation output values decreases.

The formulae of Gradient descent and Cross Entropy function that have been implemented in the experiment are given below:

The values of W, b and learning rate are initialized.

For epoch in range (1000):

$$Z = W^T X + b$$
$$p = \sigma(Z)$$

$$L = \frac{-(y\log p + (1-y)\log(1-p)}{m}$$

$$= \frac{-(ylog(\sigma(Z)) + (1-y)\log(1-\sigma(Z)}{m}$$

Taking the partial derivative of L,

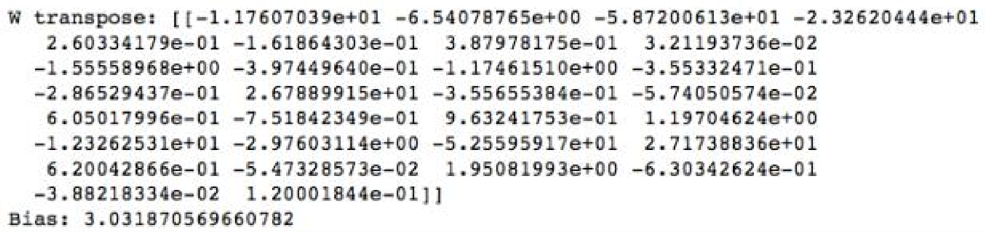
$$\frac{\partial L}{\partial \theta} = \frac{\partial}{\partial \theta}(-\frac{1}{m}((ylog(\sigma(Z)) + (1-y)\, log(1-\sigma(Z)))$$

$$= -\frac{1}{m}\left(\left(\frac{y\,\sigma(Z)(1-\sigma(Z))}{\sigma(Z)}\right) X1 + \left(\frac{(1-y)(\sigma(Z)(1-\sigma(Z))}{1-\sigma(Z)}\right) X1\right)$$

$$= -\frac{1}{m}(y(1-\ \sigma(Z))X_1$$

$$= \eta\ (-\frac{1}{m}(y(1-\ \sigma(Z))X_1)$$

Using the above derivations of the Cross Entropy Function L, I have tuned the parameters W and b for 1000 epochs for different learning rates η.



```
W transpose: [[-1.17607039e+01 -6.54078765e+00 -5.87200613e+01 -2.32620444e+01
   2.60334179e-01 -1.61864303e-01  3.87978175e-01  3.21193736e-02
  -1.55558968e+00 -3.97449640e-01 -1.17461510e+00 -3.55332471e-01
  -2.86529437e-01  2.67889915e+01 -3.55655384e-01 -5.74050574e-02
   6.05017996e-01 -7.51842349e-01  9.63241753e-01  1.19704624e+00
  -1.23262531e+01 -2.97603114e+00 -5.25595917e+01  2.71738836e+01
   6.20042866e-01 -5.47328573e-02  1.95081993e+00 -6.30342624e-01
  -3.88218334e-02  1.20001844e-01]]
Bias: 3.031870569660782
```

With the final values of W and bias, I have predicted the output values of the Testing dataset using the same genesis function and compared with the Actual output using the confusion matrix and its metrics.

CONFUSION MATRIX:

A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa).

The four metrics of the confusion matrix are:

a. TP/ True Positive: Actual output is positive and Predicted is positive
b. TN/ True Negative: Actual output is negative and Predicted is negative
c. FP/ False Positive: Actual output is negative and Predicted is positive
d. FN/ False Negative: Actual output is positive and Predicted is negative

The resultant values Accuracy, Precision and Recall are calculated from the confusion matrix values as follows:

$$\text{ACCURACY} = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

$$\text{PRECISION} = \frac{TP}{(TP+FP)}$$

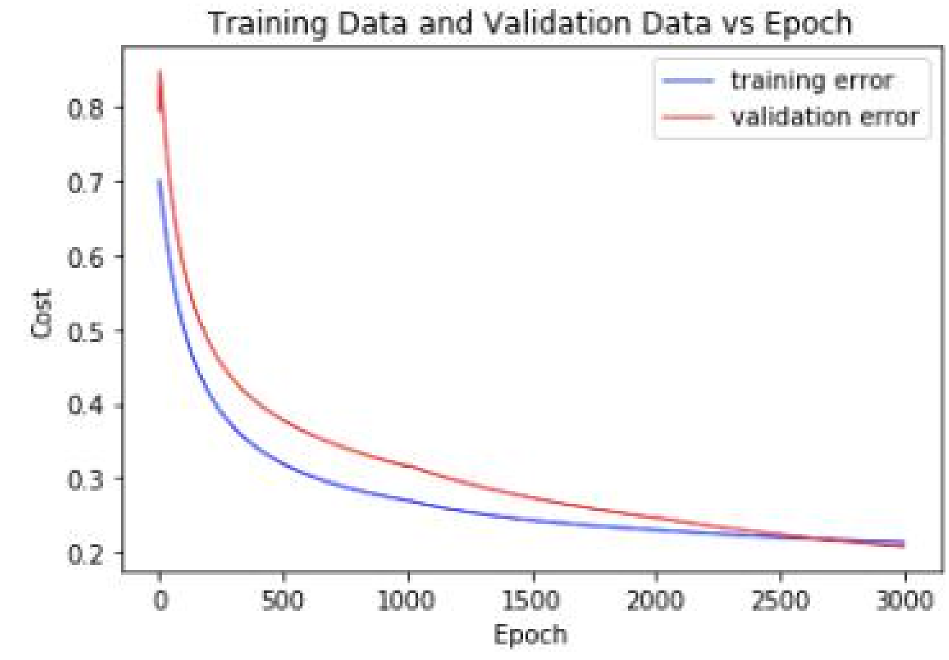
$$\text{RECALL} = \frac{TP}{(TP+FN)}$$

5 RESULTS

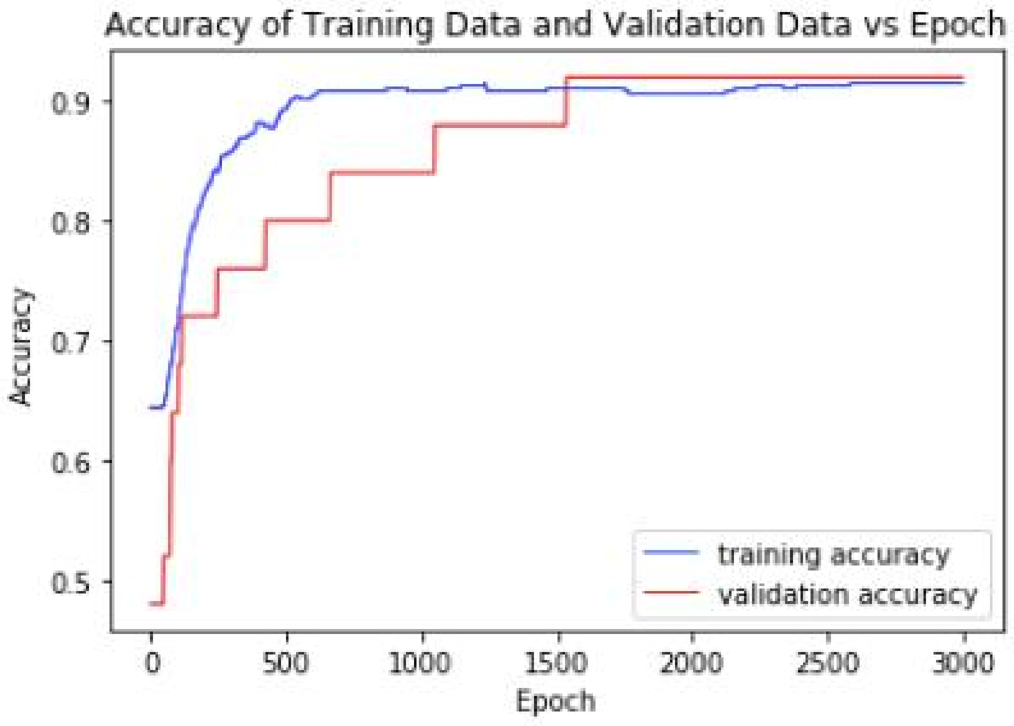
The prediction of my testdata set goes as follows:

a. Accuracy = 0.93 (approx.)
b. Precision = 0.97 (approx.)
c. Recall = 0.86 (approx.)

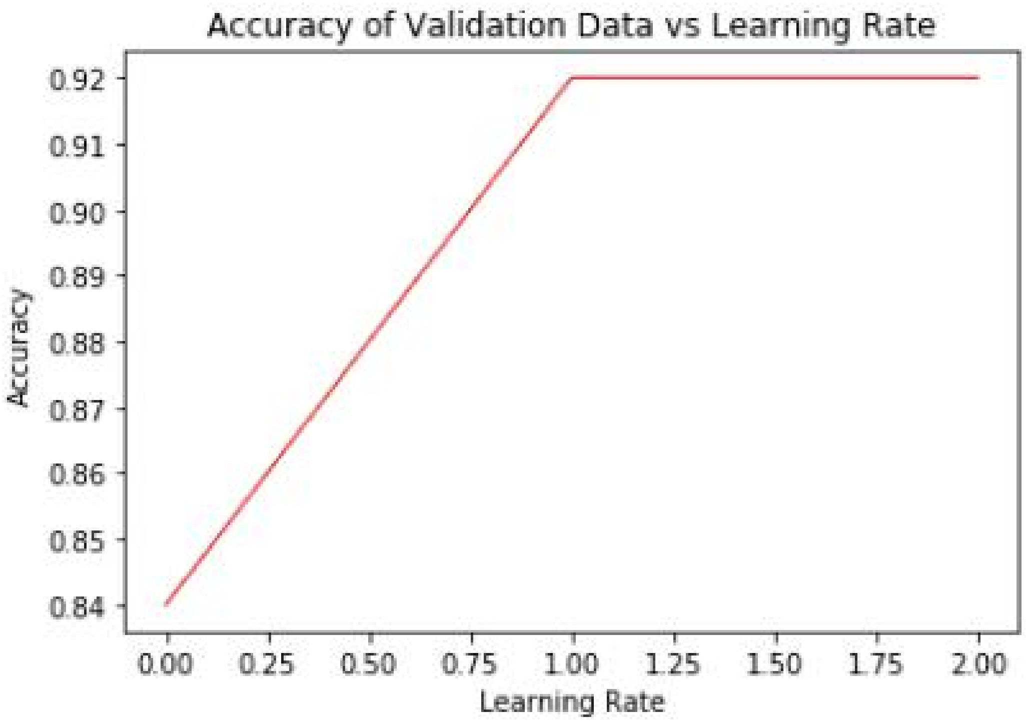
For testing the correctness of the values of hyperparameters, I have plotted a Training error function and a Validation error function V/S epoch:



The second graph plots the accuracy of Training data and Validation data against epoch to check how similar my predictions are with respect to the training dataset.



The third graph plots the variation of Accuracy of Validation data with respect to the learning rate.



6 CONCLUSION

The experiment provided with an overall understanding of how a logistic regressor works on real-life datasets for a classification problem. The accuracy of the model shows that it can predict on unknown datasets too with a pretty decent accuracy rate in future.

ACKNOWLEDGEMENTS:

[1] https://www.quora.com

[2] https://medium.com/

[3] https://ml-cheatsheet.readthedocs.io/

[4] https://towardsdatascience.com/

[5] https://www.wikipedia.org