SOLVING TWO CLASS PROBLEM USING LOGISTIC REGRESSION

Shwetasree Chowdhury

Person Id: 50296995

Department of Computer Science

State University of New York at Buffalo

shwetasr@buffalo.edu

ABSTRACT

The task of this project is to perform classification using machine learning. It is for a two class problem. The features used for classification are pre-computed from images of a fine needle aspirate (FNA) of a breast mass. The task at hand is to classify suspected FNA cells to Benign (class 0) or Malignant (class 1) using logistic regression as the classifier. The dataset in use is the Wisconsin Diagnostic Breast Cancer (wdbc.dataset).

PROBLEM STATEMENT

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset will be partitioned into 3 sets for training, validation and testing of our Logistic Regression Model. The problem at large is to consider each set of attributes associated with a Benign (class 0) or Malignant (class 1) to train, validate and finally predict the classifier for a 3rd set of data. It is a 2 class problem, as the classifier is either a binary 0 or 1, as in “Benign” or “Malignant”. Logistic Regression is an ideal regression algorithm for this dataset as the model predicts P(Y=1) as a function of X, i.e label the output as a probability of a categorical dependent variable.

THEORY

Dataset

Wisconsin Diagnostic Breast Cancer (WDBC) dataset will be used for training, validation and testing. The dataset contains 569 instances with 32 attributes (ID, diagnosis (B/M), 30 real-valued input features). Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. Computed features describes the following characteristics of the cell nuclei present in the image:
The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features.

What is Logistic Regression and why use it?

Logistic Regression is a model that is used to solve classification problems. It is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

Logistic regression uses the logistic sigmoid function to transform the output resulting in the probability values between 0 and 1, which then can be assigned a discrete classes 0 and 1 or "no" and "yes" based on a particular threshold lying between 0 and 1. For example, values lying above 0.5 can be assigned class 1 and values less than 0.5 can be assigned a class 0.

In our case since we have two classes "Benign" and "Malignant", this is binary classification problem.

Logistic regression is generally used where the dependent variable is Binary or Dichotomous. That means the dependent variable can take only two possible values such as "Yes or No", "Default or No Default. Independent factors or variables can be categorical or numerical variables.

The equation for the logistic regression is given by:

$$y(x, w) = w^T \phi(x) + b \tag{1}$$

And the sigmoid function is given by:

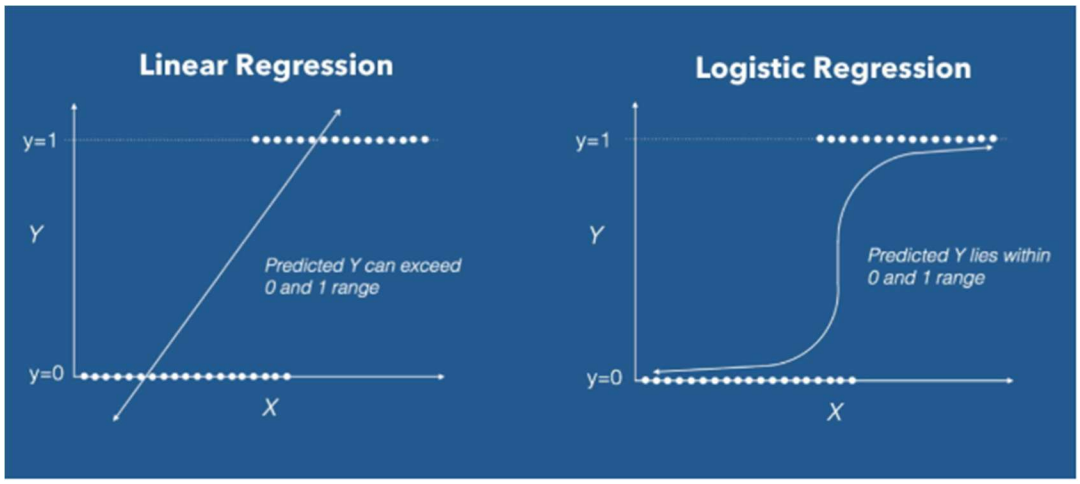
$$P(X) = \frac{1}{1+ e^{-x}} \tag{2}$$

Where $X = w^T x + b$. (Note: Notation wise y = X)

The underlying algorithm of Maximum Likelihood Estimation (MLE) determines the regression coefficient for the model that accurately predicts the probability of the binary dependent variable.

The algorithm stops when the convergence criterion is met or maximum number of iterations are reached. Since the probability of any event lies between 0 and 1 (or 0% to 100%), when we plot the probability of dependent variable by independent factors, it will demonstrate an 'S' shape curve.

Image below describes the difference between Linear Regression and Logistic Regression.



IMPLEMENTATION

In order to work on about any dataset, there are a few steps that we have to follow in order for Logistic Regression to be conducted.

1. Read the Dataset
2. Dataset Normalization and Pre-Processing
3. Partitioning into Training, Validation and Testing sets
4. Construction of Logistic Regression Model
5. Training the model to get weights using Training set
6. Validation on validation set.
7. Hyper-Parameter Tuning
8. Testing the final trained weights on the Test set

All the steps are explained as follows:

Read the Dataset

- The data is read from the csv file using Pandas Dataframe and store for further processing.

Dataset Pre-Processing and Normalization

The goal of **normalization** is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values, usually required for datasets with different kinds of values.
This is required since if the features in the data set have values in different ranges, it takes the model a long time to converge and achieve the global/local minima. Hence, we do normalization to get the feature values in similar ranges and help the model to converge more quickly.

- The first column which is the **'id'** column, has been dropped as we do not really require that as a useful feature. It is unique in nature and does not aid to the overall classification problem we are trying to solve.
- The dataset is then transformed from a dataframe into a numpy array (for easier manipulation)
- Initial weights are chosen randomly.
- Then the **B- "Benign" and M-" Malignant"** in the first columns labels are changed to 0 and 1 - indicative of 0 for B and 1 for M. This is done since the output from our prediction is in the form of probabilities and numeric, and can be compared to the original labels easily for calculation accuracy and other parameters if they are also numeric.
- The data is then shuffled to randomize it so that the model and train well.
- Then the data is normalized to get the feature values in the similar range so that the model can converge easily and we can get better results.

Data Partition

Data partitioning is an interesting and a very important step while using Logistic Regression for a dataset. The basic idea behind this being, we train our model on a set of labelled data, validate the training using another set, and finally test the training set with the aforementioned trained and validated parameters. Since our model uses a culmination of factors called 'hyper-parameters', we need to execute our model with each of these hyper-parameters and deduce our inferences accordingly.

- In this project, the wdbc dataset has 570 sets of FNA images with 30 attributes to each.
- We partition the data in the following ways:
    - **Training Data**: 80% of original dataset = 80% of 570 ~ 456 rows
    - **Validation Data**: 10% of original dataset= 10% of 570 ~ 56 rows
    - **Testing data**: 10% of original dataset= 10% of 570 ~ 57 rows
- For each of the dataset both feature matrix and the label matrix is created.

Implementing Gradient Descent solution for Logistic Regression Model

- Logistic regression is used for solving classification problem. Logistic regression uses the logistic sigmoid function to transform the output resulting in the probability values between 0 and 1, which then can be assigned a discrete classes 0 and 1 or "no" and "yes" based on

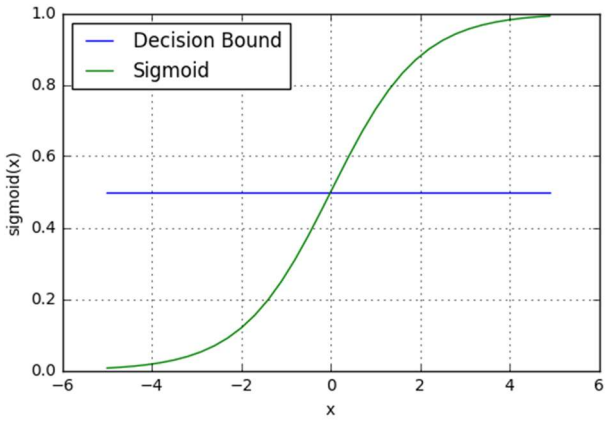
a particular threshold lying between 0 and 1. For example, values lying above 0.5 can be assigned class 1 and values less than 0.5 can be assigned a class 0.

- In our case since we have two classes "Benign" and "Malignant", this is binary classification problem.

- Equation(1) $\boldsymbol{y(x, w) = w^T \phi(x) + b}$ is used to get the output which is then transformed by using a logistic sigmoid function that outputs a probability value between 0 and 1. The sigmoid function is given

$$\boldsymbol{P(X) = \frac{1}{1+ e^{-x}}}$$

- A threshold or a boundary value is chosen that divides the output values from the above step into 2 classes namely 0 and 1 corresponding to **Benign and Malignant** respectively.
- In our case the threshold is chosen as 0.5 i.e. the values of P(X) which are greater than threshold are assigned label 1 and others label 0.
- This behaviour can be seen from the below graph.



- A cost function is required to calculate the performance of our model.
- Since our function that predicts the target is not linear we, cannot use the root mean square function as squaring it results in non-convex function and has multiple minima points.
- Hence, the gradient descent algorithm, which works on moving towards a single minima point, won't work well. Therefore, we use a cost function **called log loss function** given by:

$$E(w) = \frac{-1}{N} \sum_{n=1}^{N} \{t_n \ln y_n \quad + (1 - t_n) \ln(1 - y_n)\}$$

Where $y_n = sigmoid(w^T x_n + b)$

- Since we need to minimize the log loss function or the cost function, we use the derivative/gradient of this function to update our weights.
- The gradient of the above equation is given as:

$$\nabla E(w) = (y_n - t_n)x$$

- To update the weights, we use the following equation:
$$w^{(\tau+1)} = w^{(\tau)} + \Delta w^{(\tau)}$$
- The learning rate is set.
- For each iteration, the weights are then calculated by using the derivative of the log loss function.
- After the epochs, we get the final weights and use them to predict the outputs and get the classes of those outputs using a threshold of 0.5
- The accuracies and costs for each epoch for training and validation sets are saved and reported.
- Also, the accuracy, precision and recall for the test set are found and reported.

Tuning of Hyper-parameter

Parameters are tuned to bring out the change in the accuracy, cost, precision and recall. Gradient Descent is the process of minimizing a function by following the gradients of the cost function. This evaluates and updates the coefficients every iteration called stochastic gradient descent to minimize the error of a model on our training data.

The way this optimization algorithm works is that each training instance is shown to the model one at a time. The model makes a prediction for a training instance, the error is calculated and the model is updated in order to reduce the error for the next prediction.

This procedure can be used to find the set of coefficients in a model that result in the smallest error for the model on the training data. Each iteration, the coefficients (b) in machine learning language are updated

- Learning rate and number of epochs are changed to find the best accuracy and see the behaviour how it is changing with both of them.
- Cost is calculated using the log loss function.
- Confusion matrix is calculated using the inbuilt sklearns metrics library and the accuracy, recall and precision are calculated for the test set.
- The results are reported and analysed.

Testing final weights on the test set

- After the Hyper-Parameters are tuned and all the results obtained, the best parameters are chosen and the final weights calculated.
- The final weights are used to predict the outputs using the test set.

- Confusion matrix is calculated for the predicted outputs.
- The accuracy, precision and recall for test set are then calculated and reported. These are found using the following formulas:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Where TP = true positive, TN = true negative, FP = false positive and FN = false negative.

The following shows all the test cases run for the logistic regression using gradient descent and the graphs and the metrics are reported for each.

The graphs and the obtained data are analyzed and inferred in the Conclusion section and the best results are reported.

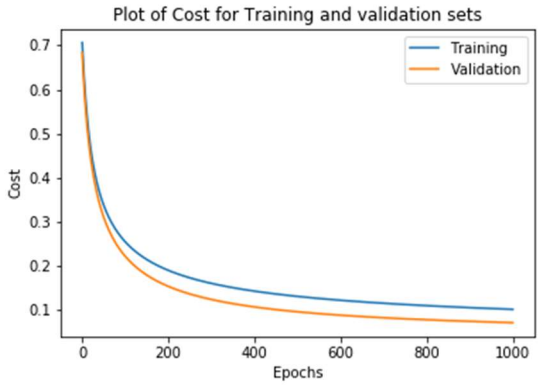
HYPER-PARAMETER TUNING RESULTS

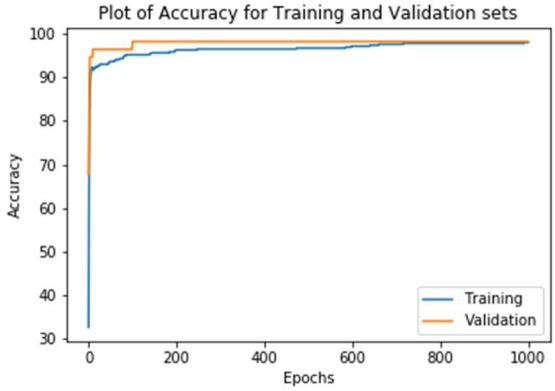
Case 1

Observation: A very smooth learning curve for the model. As Epoch increases, loss/cost decreases, accuracy increases.

Parameters:

Parameter	Value
Learning Rate	0.01
Epochs	1000



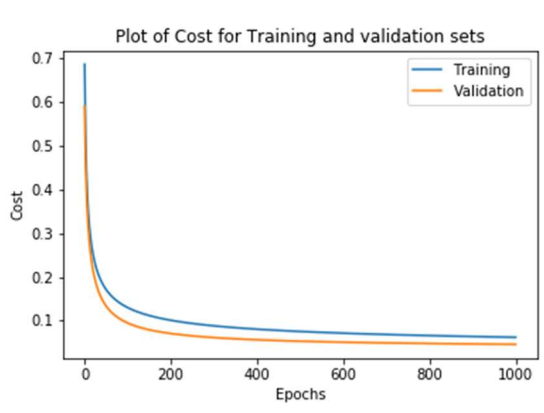


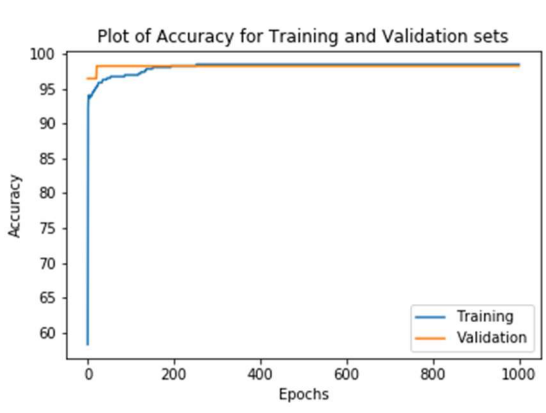
```
Number of Epochs =  1000
Learning rate =  0.01

-----------------------Training Set----------------------

Confusion Matrix for Training set:
 [[287   2]
 [  7 160]]

Training Accuracy   =  98.02631578947368 %
Training Cost   =  0.10120746486102584
Training Precision   =  0.9876543209876543
Training Recall   =  0.9760479041916168

-----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[34  0]
 [ 1 21]]

Validation Accuracy =  98.21428571428571 %
Validation Cost =  0.07094753518262156
Validation Precision   =  1.0
Validation Recall   =  0.9760479041916168

------------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 1 22]]

Testing Accuracy    =  98.24561403508771 %
Testing Cost    =  0.1367116882581812
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

Case 2

Observation: Case 2 also has a smooth learning curve, very similar to case 1.
Parameters:

Parameter	Value
Learning Rate	0.05
Epochs	1000





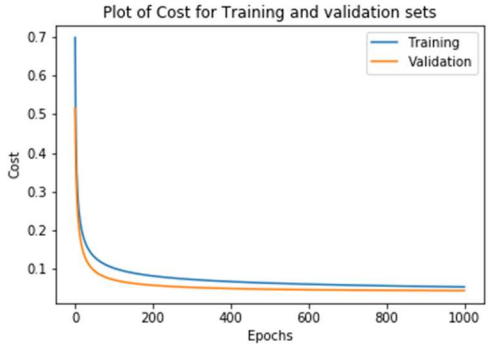
```
Number of Epochs =  1000
Learning rate =  0.05

----------------------Training Set---------------------

Confusion Matrix for Training set:
 [[287   2]
 [  5 162]]

Training Accuracy   =  98.46491228070175 %
Training Cost   =  0.06192907459352847
Training Precision   =  0.9878048780487805
Training Recall   =  0.9760479041916168

----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[34  0]
 [ 1 21]]

Validation Accuracy =  98.21428571428571 %
Validation Cost =  0.045828243873517985
Validation Precision   =  1.0
Validation Recall   =  0.9760479041916168

------------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 1 22]]

Testing Accuracy    =  98.24561403508771 %
Testing Cost    =  0.13483929944259268
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

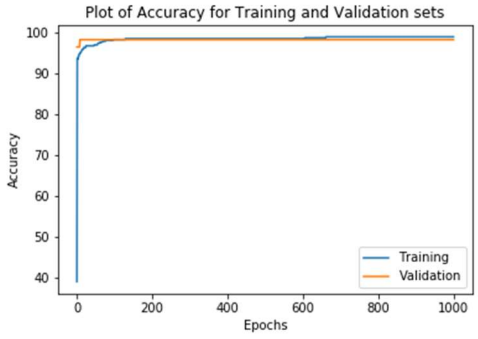
Case 3

Observation: Accuracy slightly lower than other cases above, as no of epochs is low.

Parameters:

Parameter	Value
Learning Rate	0.1
Epochs	1000





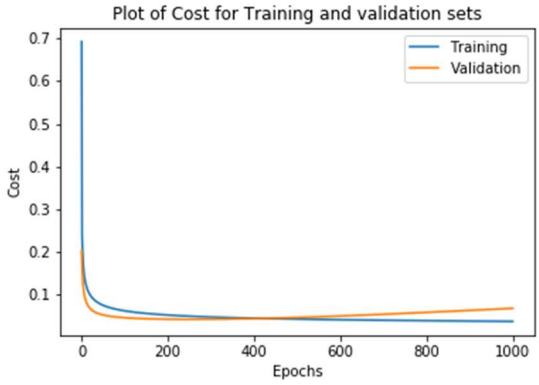
```
Number of Epochs =  1000
Learning rate =  0.1

----------------------Training Set----------------------

Confusion Matrix for Training set:
 [[288   1]
 [  3 164]]

Training Accuracy   =  99.12280701754386 %
Training Cost   =  0.052020749043537165
Training Precision   =  0.9939393939393939
Training Recall   =  0.9760479041916168

----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[34  0]
 [ 1 21]]

Validation Accuracy =  98.21428571428571 %
Validation Cost =  0.04256841792483996
Validation Precision   =  1.0
Validation Recall   =  0.9760479041916168

-----------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 2 21]]

Testing Accuracy    =  96.49122807017544 %
Testing Cost    =  0.14280167318964188
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

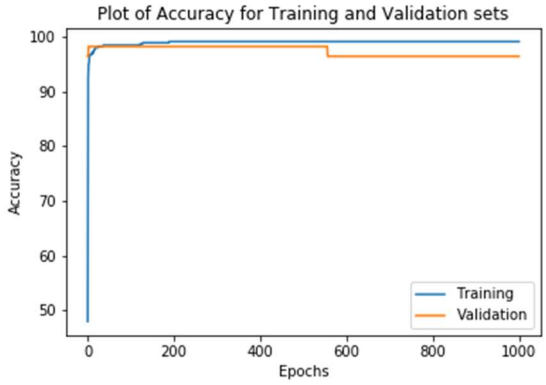
Case 4

Observation: Performance slightly better than Case 3, but lower than other cases, due to low number of epochs.

Parameters:

Parameter	Value
Learning Rate	0.5
Epochs	1000






```
Number of Epochs =  1000
Learning rate =  0.5

----------------------Training Set----------------------

Confusion Matrix for Training set:
 [[288   1]
 [  3 164]]

Training Accuracy   =  99.12280701754386 %
Training Cost   =  0.03705424229325287
Training Precision   =  0.9939393939393939
Training Recall   =  0.9760479041916168

----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[34  0]
 [ 2 20]]

Validation Accuracy =  96.42857142857143 %
Validation Cost =  0.0675512049868452
Validation Precision   =  1.0
Validation Recall   =  0.9760479041916168

-----------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 2 21]]

Testing Accuracy    =  96.49122807017544 %
Testing Cost    =  0.1659093251130803
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

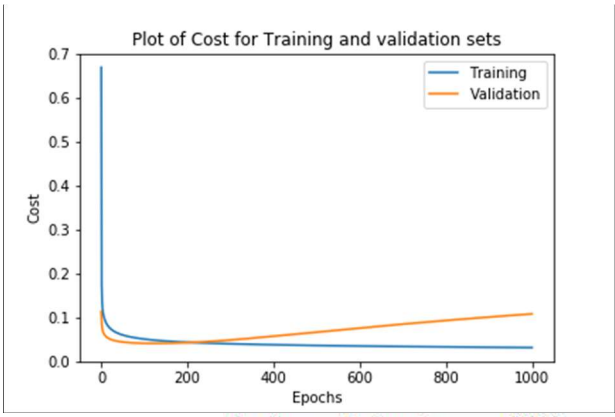
Case 5

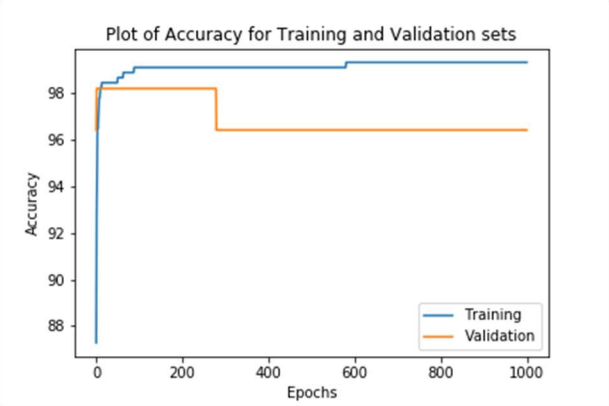
Observation: Very steep learning curve. Model fails to converge later on as learning rate is too high.

Parameters:

Parameter	Value

Learning Rate	1
Epochs	1000





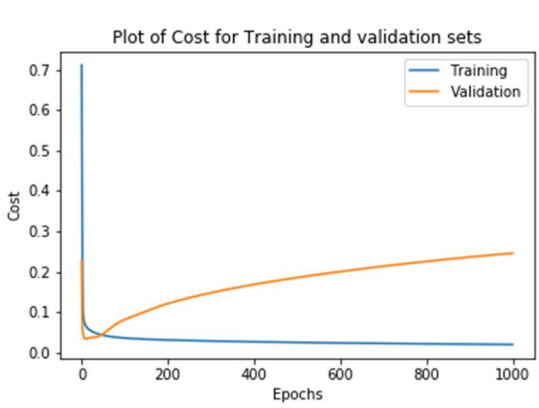

```
Number of Epochs =  1000
Learning rate =  1

-----------------------Training Set----------------------

Confusion Matrix for Training set:
 [[289   0]
 [  3 164]]

Training Accuracy   =  99.3421052631579 %
Training Cost   =  0.03219909340371714
Training Precision   =  1.0
Training Recall   =  0.9760479041916168

-----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[34  0]
 [ 2 20]]

Validation Accuracy =  96.42857142857143 %
Validation Cost =  0.10896060656898791
Validation Precision   =  1.0
Validation Recall   =  0.9760479041916168

------------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 2 21]]

Testing Accuracy    =  96.49122807017544 %
Testing Cost    =  0.17553664922316825
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

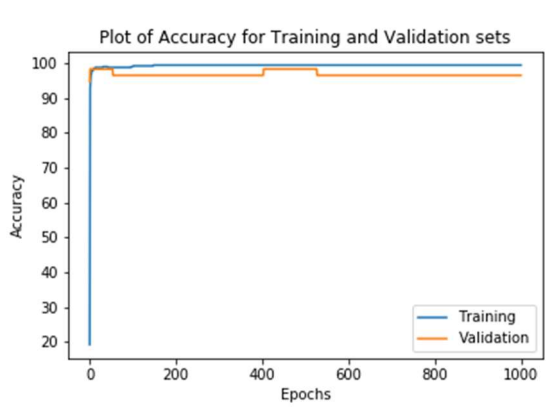
Case 6

Observation: Immediate Steep convergence and model later not being able to converge successfully.

Parameters:

Parameter	Value
Learning Rate	5
Epochs	1000





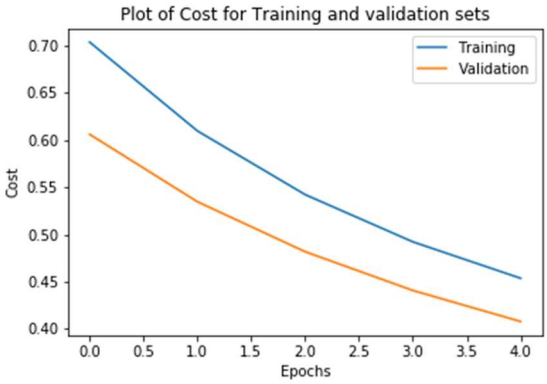

```
Number of Epochs =  1000
Learning rate =  5

----------------------Training Set----------------------

Confusion Matrix for Training set:
 [[289   0]
 [  3 164]]

Training Accuracy   =  99.3421052631579 %
Training Cost   =  0.019916935865990654
Training Precision   =  1.0
Training Recall   =  0.9760479041916168

----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[33  1]
 [ 1 21]]

Validation Accuracy =  96.42857142857143 %
Validation Cost =  0.2459608228411396
Validation Precision   =  0.9545454545454546
Validation Recall   =  0.9760479041916168

------------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 3 20]]

Testing Accuracy    =  94.73684210526315 %
Testing Cost    =  0.22915313815549093
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

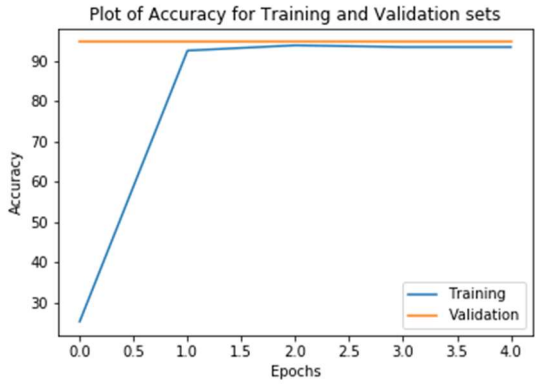
Case 7

Observation: Model is still undergoing the learning process. No of epochs extremely low.

Parameters:

Parameter	Value
Learning Rate	0.05
Epochs	5





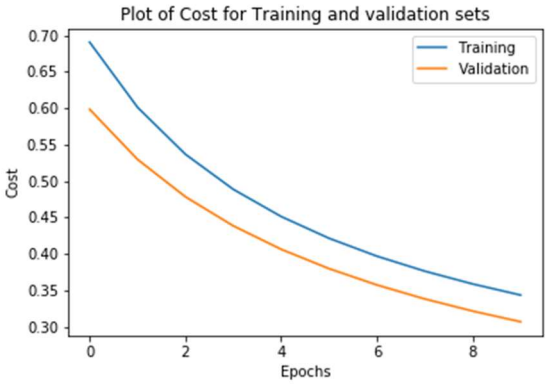

```
Number of Epochs =  5
Learning rate =  0.05

----------------------Training Set----------------------

Confusion Matrix for Training set:
 [[273  16]
 [ 14 153]]

Training Accuracy   =  93.42105263157895 %
Training Cost   =  0.42296862095461396
Training Precision   =  0.9053254437869822
Training Recall   =  0.9760479041916168

----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[33  1]
 [ 2 20]]

Validation Accuracy =  94.64285714285714 %
Validation Cost =  0.4078746343203911
Validation Precision   =  0.9523809523809523
Validation Recall   =  0.9760479041916168

-----------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 2 21]]

Testing Accuracy    =  96.49122807017544 %
Testing Cost    =  0.4115364610811277
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

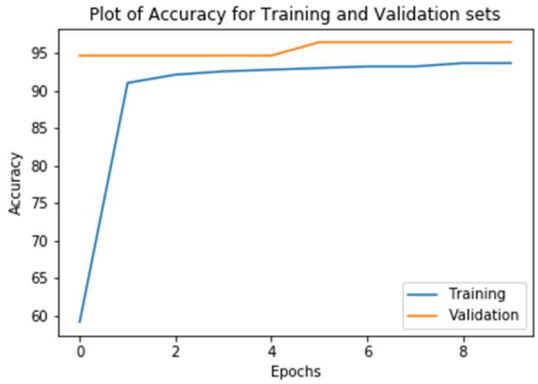
Case 8

Observation: Condition similar to case 7

Parameters:

Parameter	Value
Learning Rate	0.05
Epochs	10



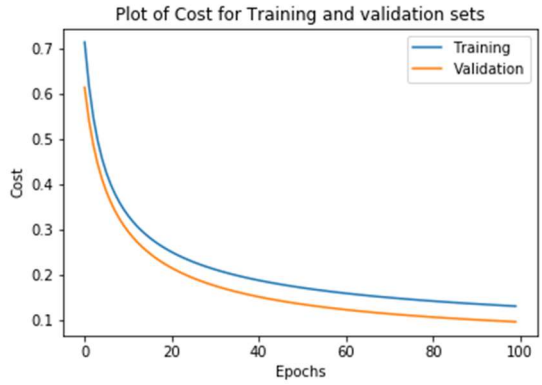


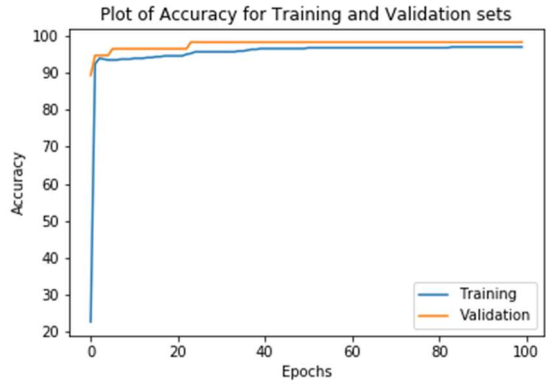

```
Number of Epochs =  10
Learning rate =  0.05

----------------------Training Set----------------------

Confusion Matrix for Training set:
 [[274  15]
 [ 14 153]]

Training Accuracy   =  93.64035087719299 %
Training Cost   =  0.33005317716973903
Training Precision   =  0.9107142857142857
Training Recall   =  0.9760479041916168

----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[34  0]
 [ 2 20]]

Validation Accuracy =  96.42857142857143 %
Validation Cost =  0.3067589571277597
Validation Precision   =  1.0
Validation Recall   =  0.9760479041916168

-----------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 2 21]]

Testing Accuracy    =  96.49122807017544 %
Testing Cost    =  0.3167038920682899
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

Case 9

Observation: Gradual cost curve. Low number of epochs.
Parameters:

Parameter	Value
Learning Rate	0.05
Epochs	100

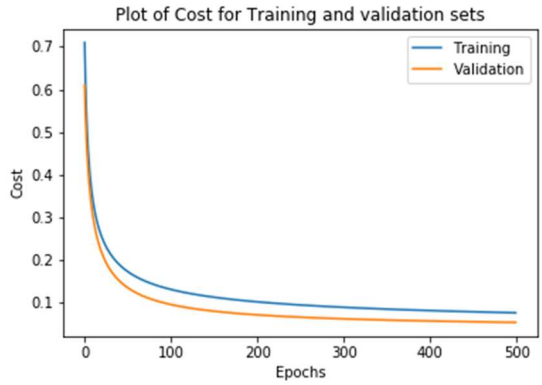


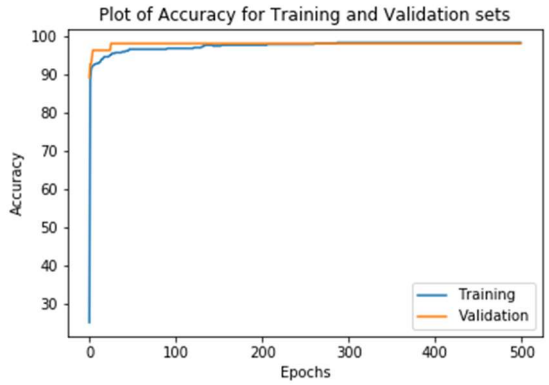



```
Number of Epochs =  100
Learning rate =  0.05

----------------------Training Set----------------------

Confusion Matrix for Training set:
 [[283   6]
 [  8 159]]

Training Accuracy   =  96.9298245614035 %
Training Cost   =  0.1298207718320263
Training Precision   =  0.9636363636363636
Training Recall   =  0.9760479041916168

----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[34  0]
 [ 1 21]]

Validation Accuracy =  98.21428571428571 %
Validation Cost =  0.09568644334058089
Validation Precision   =  1.0
Validation Recall   =  0.9760479041916168

-----------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 1 22]]

Testing Accuracy    =  98.24561403508771 %
Testing Cost    =  0.15078099524669225
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

Case 10
Observation: Low number of epochs.
Parameters:

Parameter	Value
Learning Rate	0.05
Epochs	500

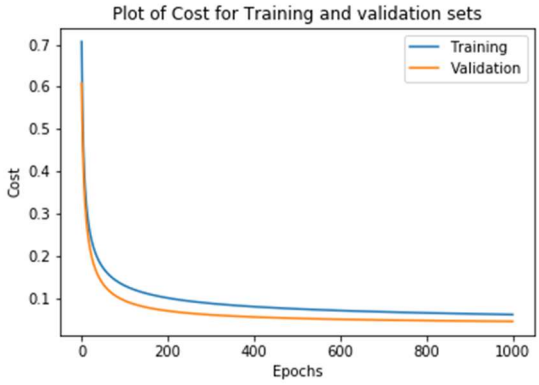


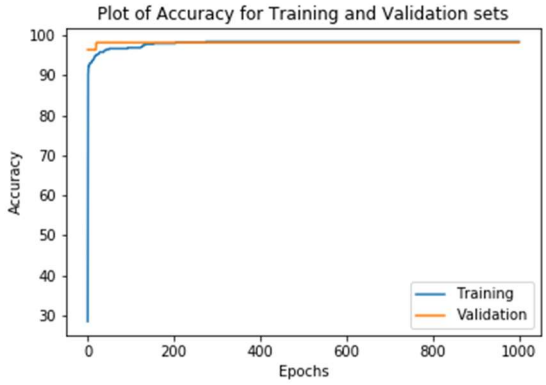


```
Number of Epochs =  500
Learning rate =  0.05

----------------------Training Set----------------------

Confusion Matrix for Training set:
 [[287   2]
 [  5 162]]

Training Accuracy   =  98.46491228070175 %
Training Cost   =  0.07537916460838866
Training Precision   =  0.9878048780487805
Training Recall   =  0.9760479041916168

----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[34  0]
 [ 1 21]]

Validation Accuracy =  98.21428571428571 %
Validation Cost =  0.05293496696212593
Validation Precision   =  1.0
Validation Recall   =  0.9760479041916168

------------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 1 22]]

Testing Accuracy    =  98.24561403508771 %
Testing Cost    =  0.1316852426283209
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

Case 11
Parameters:

Parameter	Value
Learning Rate	0.05
Epochs	1000

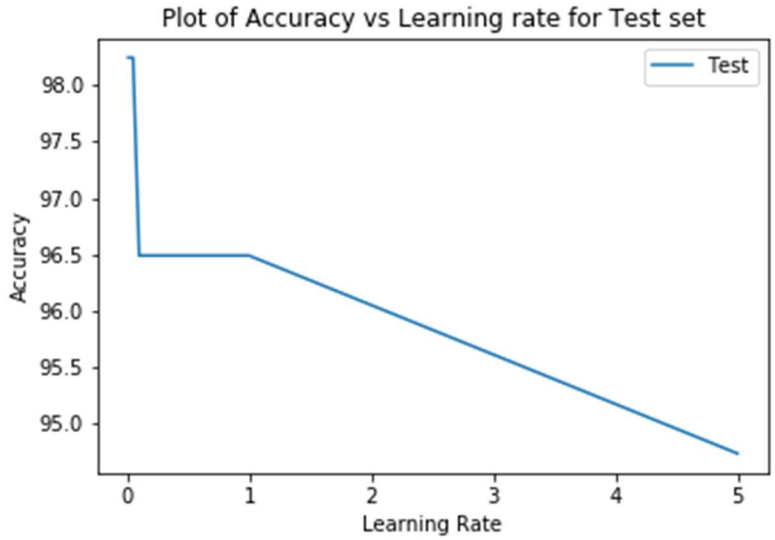




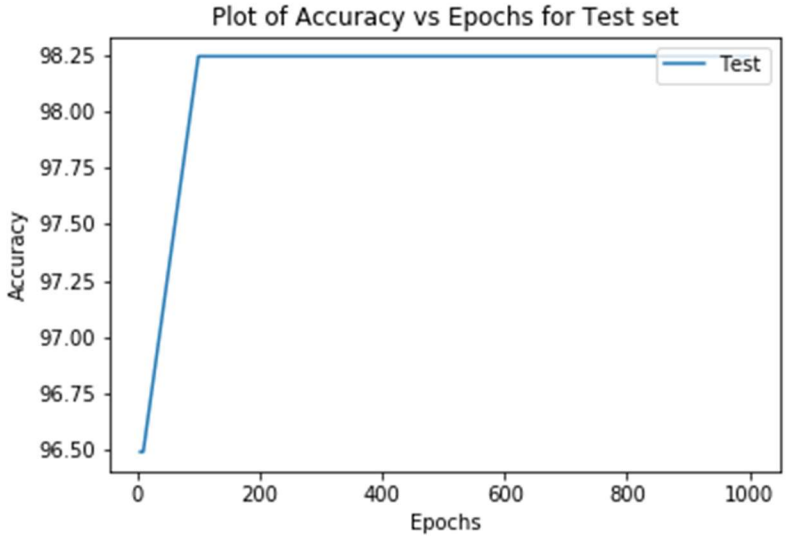

```
Number of Epochs =  1000
Learning rate =  0.05

-----------------------Training Set----------------------

Confusion Matrix for Training set:
 [[287   2]
 [  5 162]]

Training Accuracy   =  98.46491228070175 %
Training Cost   =  0.06203564920176746
Training Precision   =  0.9878048780487805
Training Recall   =  0.9760479041916168

-----------------------Validation Set--------------------

Confusion Matrix for Validation set:
 [[34  0]
 [ 1 21]]

Validation Accuracy =  98.21428571428571 %
Validation Cost =  0.045971273750495145
Validation Precision   =  1.0
Validation Recall   =  0.9760479041916168

------------------------Test Set-------------------------

Confusion Matrix for Test set:
 [[34  0]
 [ 1 22]]

Testing Accuracy    =  98.24561403508771 %
Testing Cost    =  0.1352464449922456
Test Precision   =  1.0
Test Recall   =  0.9760479041916168
```

CONCLUSION/ INFERENCES

1. The Hyper-Parameters tuned are the number of epochs and the learning rate.
2. It is Observed that as the learning rate is increased, the accuracy tends to decrease. This can be depicted by the following graph.



3. As can be seen from the graph as well as all the test cases run, the maximum accuracy occurs for **learning rate** lying in the range **0.01** to **0.05**
4. The accuracy decreases for greater values of learning rate since for larger values the steps taken are bigger and the convergence is difficult to achieve. Hence the accuracy decreases.
5. It is observed that as the number of Epochs are increased the accuracy increases and accuracy is low for low values of number of epochs. As can be seen from the graph:



The accuracy is low for epochs less than 10 and then becomes constant as the number of epochs are increased.

6. This behavior is observed since due to less number of epochs, the convergence is not achieved and the weights obtained are not optimal, resulting in bad accuracy.

7. The decision for the value of the threshold value is majorly affected by the values of precision and recall. Ideally, we want both precision and recall to be.

The best accuracy is obtained for both **Case 1** and **Case 2** where the learning rate is 0.01 and 0.05 and the number of epochs are 1000.

Final Accuracy: 98.25 %

Acknowledgment

Grateful to Professor Srihari and the wonderful TA's for working on this, with us!

References

1. https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html
2. https://www.kaggle.com/keyurparalkar/breast-cancer-prediction-using-logistic-regression
3. https://www.kaggle.com/leemun1/predicting-breast-cancer-logistic-regression
4. https://medium.com/greyatom/logistic-regression-89e496433063