Classification using ML (Logistic Regression and Neural Networks)

Rushabh Shah
(50375759)

1 Abstract:

The goal of this project is to build a machine learning model using i) Logistic Regression and ii) Neural Networks and use it to predict if a patient has diabetes or not. The problem takes 8 features as input and classifies if a patient has diabetes(class 1) or not (class 0), based on the diagnostic measurements provided in the dataset. This can be viewed as a classification problem.

2 Dataset:

Pima Indians diabetes data. The total number of training examples are 768 which have been divided into train, validation and test datasets in the proportion of 60%,20% and 20% respectively.

3 Introduction

3.1 Linear Regression

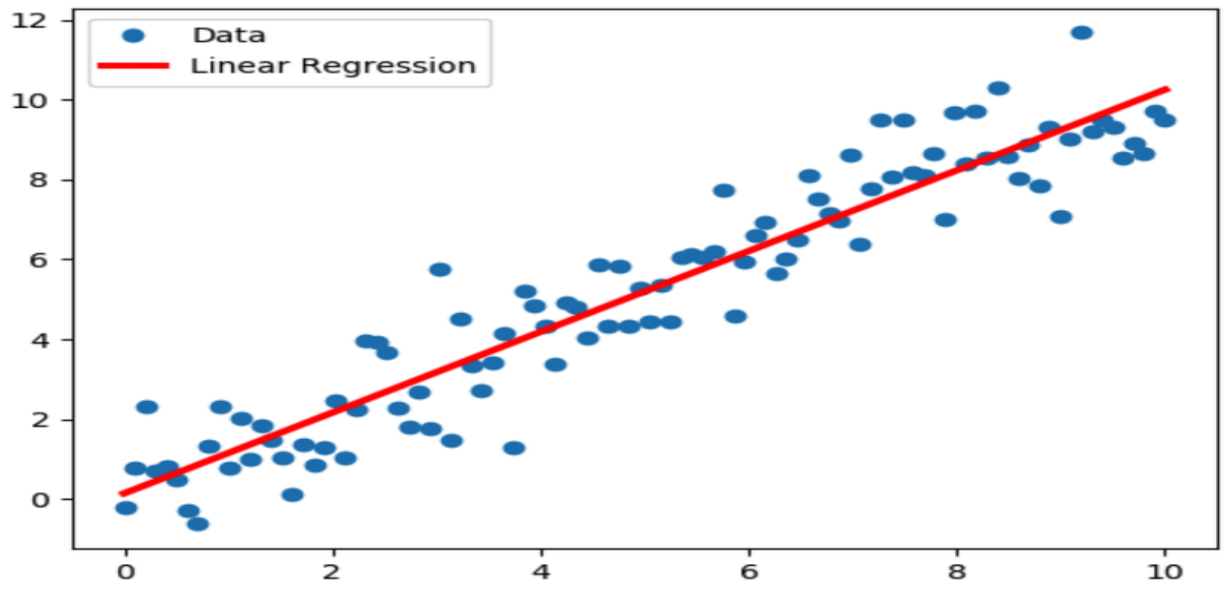
Linear Regression is a form of supervised learning which is used to predict a continuous value based on the input features. The model tries to fit a straight line. The hypothesis function is given by-

$$h_\theta(X) = W0 + W(\text{Transpose}) * X$$

where W0 and W are the weights.

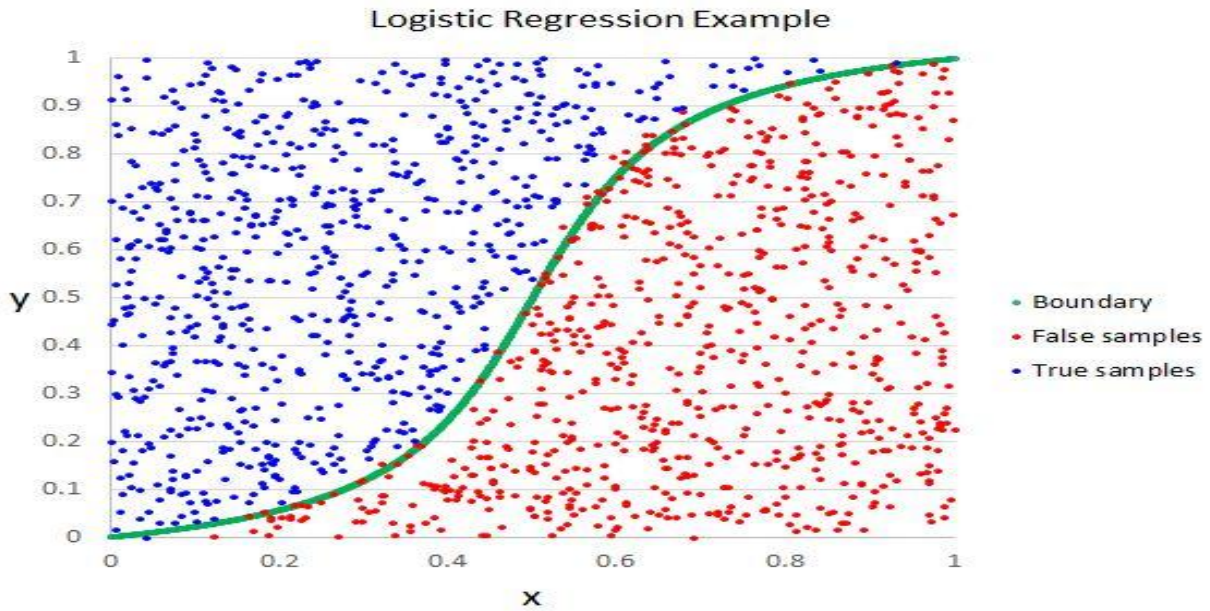
The cost function is

J(W0 , W) = 1/ 2m ∑($h\theta$ (X^i)− Y^i)^2

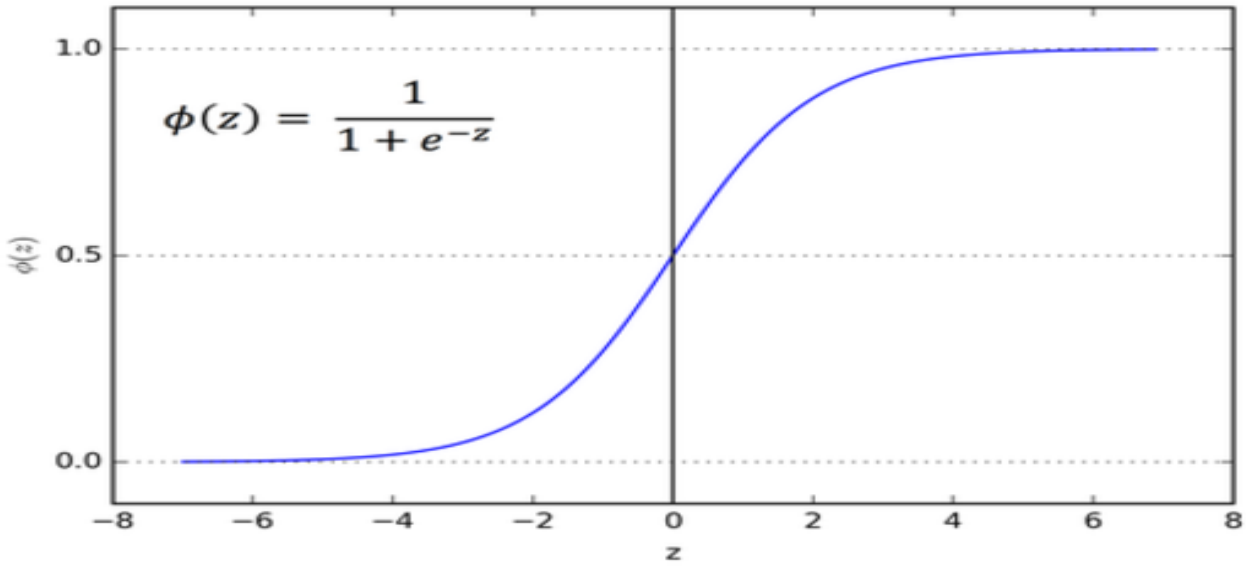


3.2 Logistic Regression

Logistic regression can be used as a classification algorithm that allows us to predict results and categorize them into certain classes. For example, based on scores of students on two exams, we can predict whether the student has passed the exam or not(0 or 1)



In logistic regression, we make use of the sigmoid function which limits the value between 0 and 1.



The Hypothesis Equation is given by:

$\hbar\theta$(X) = σ(W0 + WT $*$ X)

σ = 1/(1 + e^−z)

The cost function for logistic regression is defined as:

−log ($\hbar\theta$ (X), if y = 1

−log (1 − $\hbar\theta$ (X), if y = 0

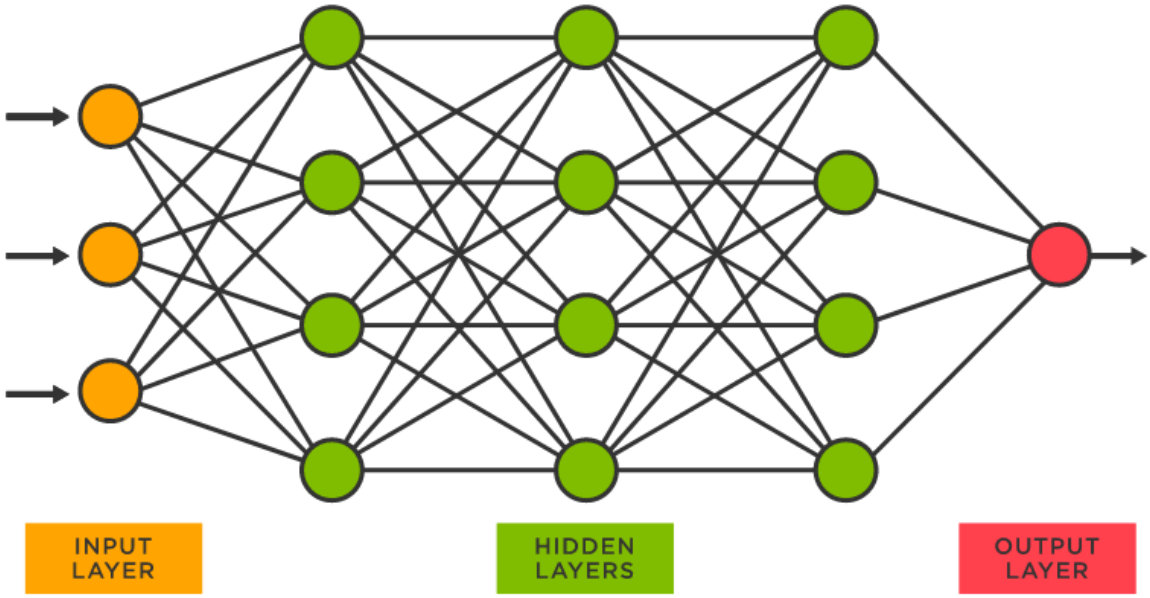
J(W0 , W) = − 1/m ∑y^i log($h\theta$X^i) + (1 −y^i) log(1 − $h\theta x$^i)

In order to make predictions we need to set a threshold value above which the output will be predicted to 1.

3.3 Neural Networks

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus a neural network is an artificial neural network, for solving artificial intelligence problems.

The neural network makes use of units called neurons and hidden layers for training the model.



Neural networks make use of the relu (Rectified Linear Unit) activation function and softmax or sigmoid activation function for the output layer depending on whether the problem is multi-class or not.

4 Implementation

4.1 Preprocessing

The dataset used in this project is Pima Indians diabetes. The total number of training examples is 768. There are total 8 features :-

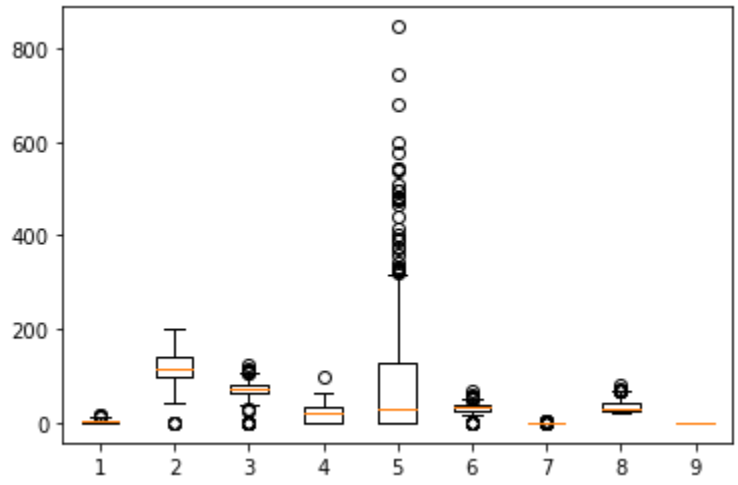
1) Pregnancies
2) Glucose
3) Blood Pressure
4) Skin Thickness
5) Insulin
6) Diabetes Pedigree function
7) BMI
8) Age

The data has been divided into train, validation and test datasets in the proportion of 60%,20% and 20% respectively.

Also, we scale the data to a normalized scale to get better results.

4.2 Logistic Regression Implementation(Part 1)

The implementation starts with importing necessary libraries for data preprocessing and plotting graphs. The data is loaded using pandas dataframe. We move forward with data preprocessing and as an additional step we get rid of the outliers which may skew the final result.



We can see that column 5 contains a few outliers as seen from the boxplot.

The data has been scaled using StandardScaler() and converted to a numpy array for further processing.

The weights b,w are then initialized. w has the dimension 8 * 1, where 8 denotes the number of features used in the dataset.

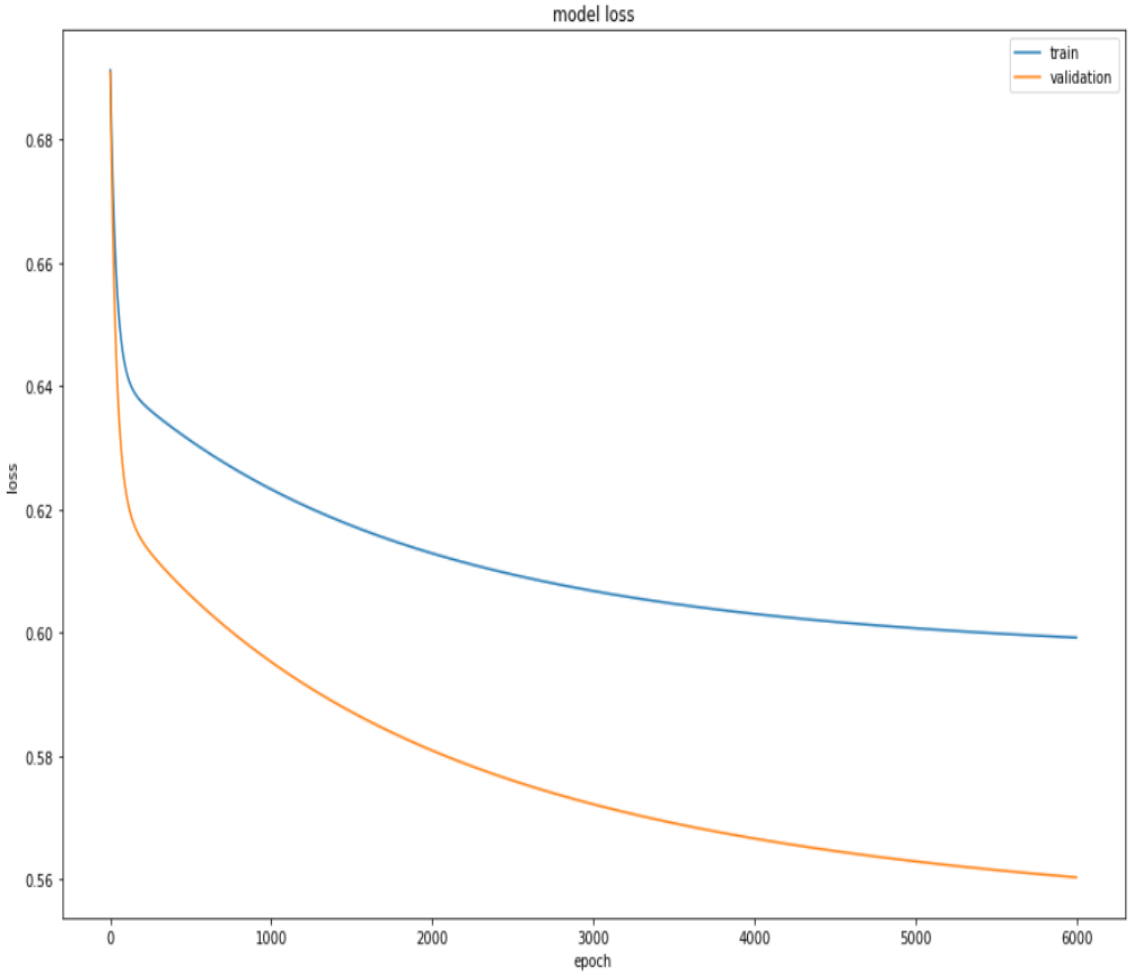
We then implement the function to train our model wherein we make use of the gradient descent which is used to find the global minimum. On running gradient descent for a certain number of epochs we get our learning parameters.

```
w = w - learning_rate*dw
b = b - learning_rate*db
```

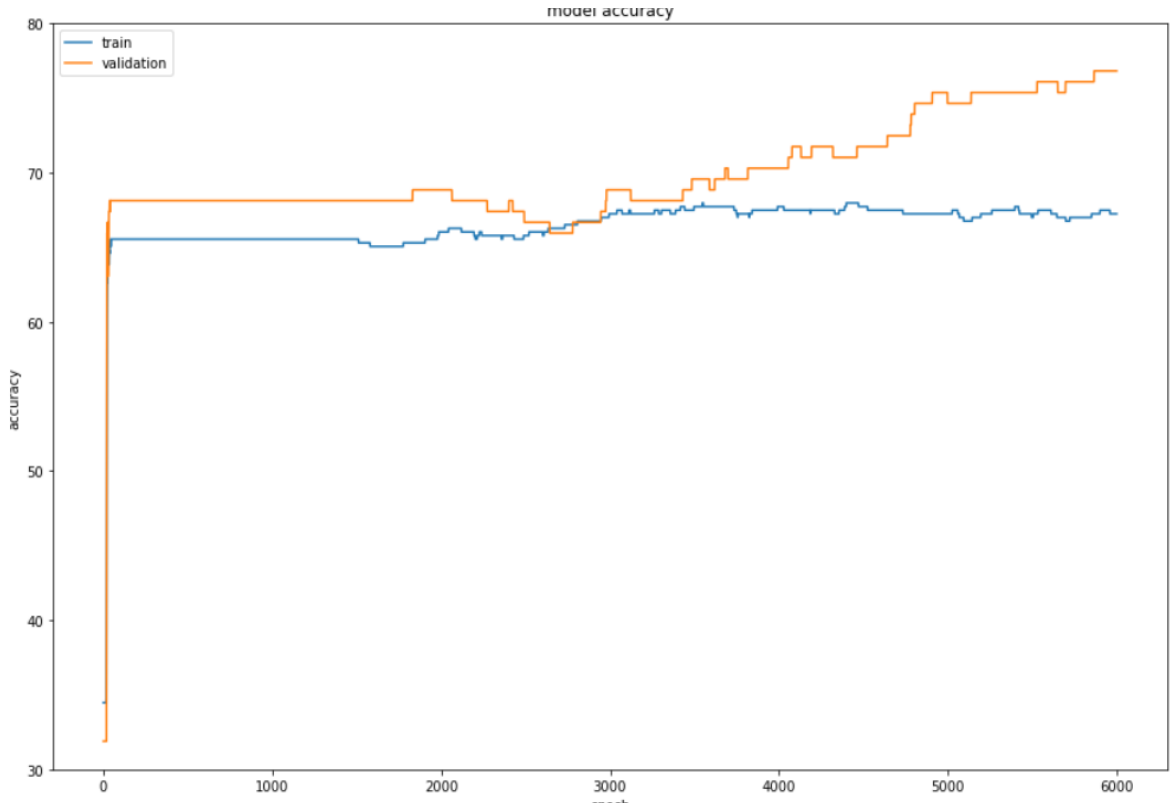
We can then make use of the validation dataset to tune the hyper parameters for optimal results.

Results:

Based on the results obtained from the validation data set, we tune the learning rate to 0.01 and number of epochs to 6000. The train vs validation cross entropy loss graph is as follows:



The training and validation accuracy is shown below:



We get test accuracy of around 76.087% using the trained model with the learning rate of 0.01 and 6000 epochs

4.3 Neural Networks Implementation(Part 2)

In this part of the assignment, we use neural networks with one hidden layer to make predictions. Here, we make use of a popular python library called Tensorflow. As before, we will preprocess and scale the data using StandardScaler().

Sequential model is used to construct the model. We have one input layer, a hidden layer and an output layer. The hidden layer has 8 neurons. We have made use of the Adam optimizer.

Types of Regularization used in Neural Networks:-

L1 Regularization:-

L1 regularization effect on the neural network weight values is that it penalizes weight values that are close to 0 by making them equal to 0. Negative weight values also take a value of 0; so if a weight value is -2, under the effect of L1 regularization, it becomes 0.

The general intuition with L1 regularization is that, if a weight value is close to 0 or very small, then it's negligible when it comes to the overall performance of the model, therefore making it 0 does not affect the model's performance and can reduce the memory capacity of the model.

'*i*' in the mathematical notation represents the index of the current weight, and '*n*' represents the total number of weights values within the layer. '*W*' represents the weight value.

MSE + lambda * ∑ |w| (summation from i=1 to n)

L2 Regularization:-

L2 regularization also penalizes weight values. For both small weight values and relatively large ones, L2 regularization transforms the values to a number close to 0, but not quite 0.

- l2 penalizes the sum of the square of the weights (weight²)

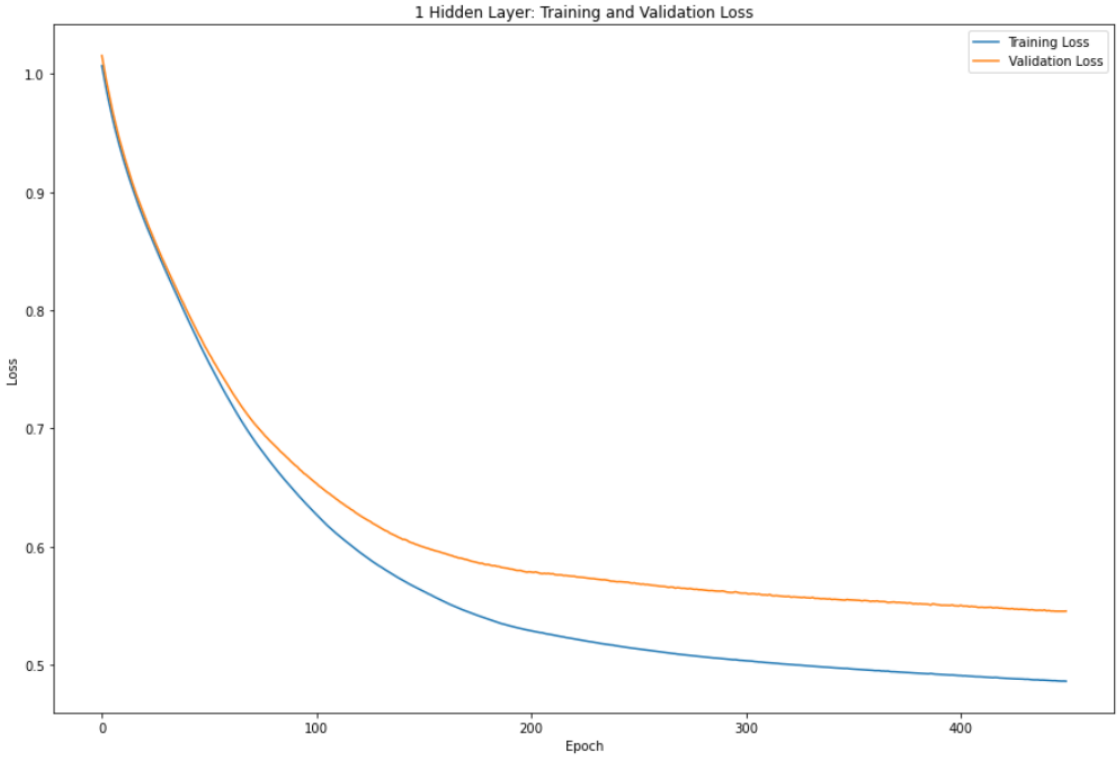
MSE + lambda * ∑ |w|^2 (summation from i=1 to n)

Results:

Using leaning rate as 0.0001, epochs as 450 and batch size as 10, the results obtained are:-

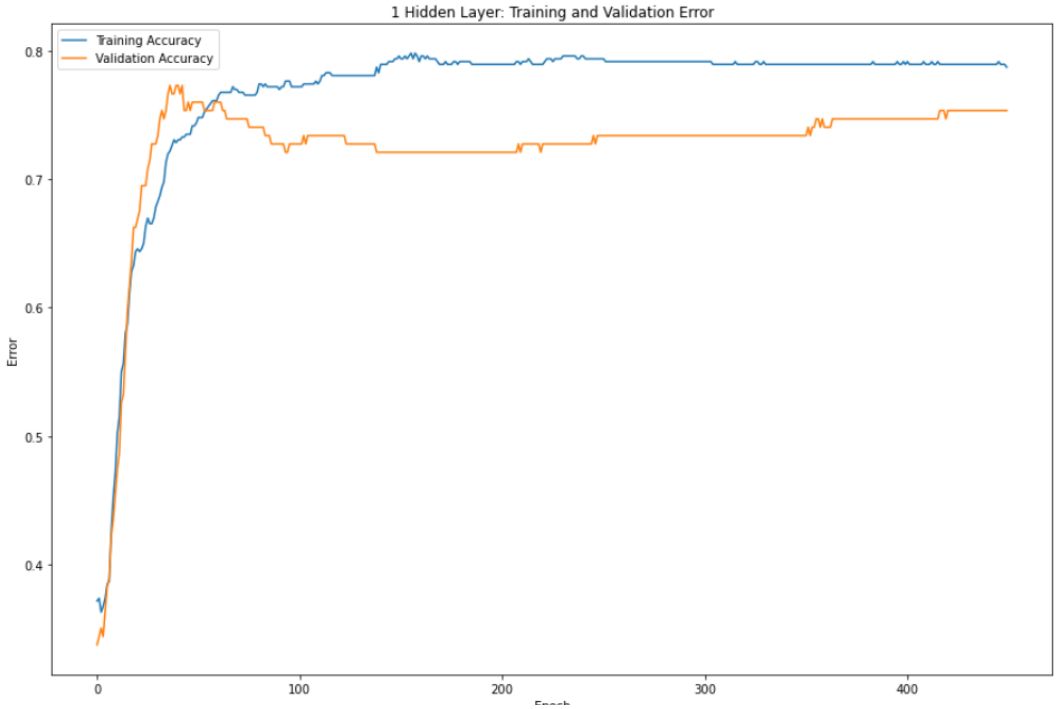
For L1 Regularization

The train vs validation loss graph is as follows:



The graph for Validation vs Training accuracy:-

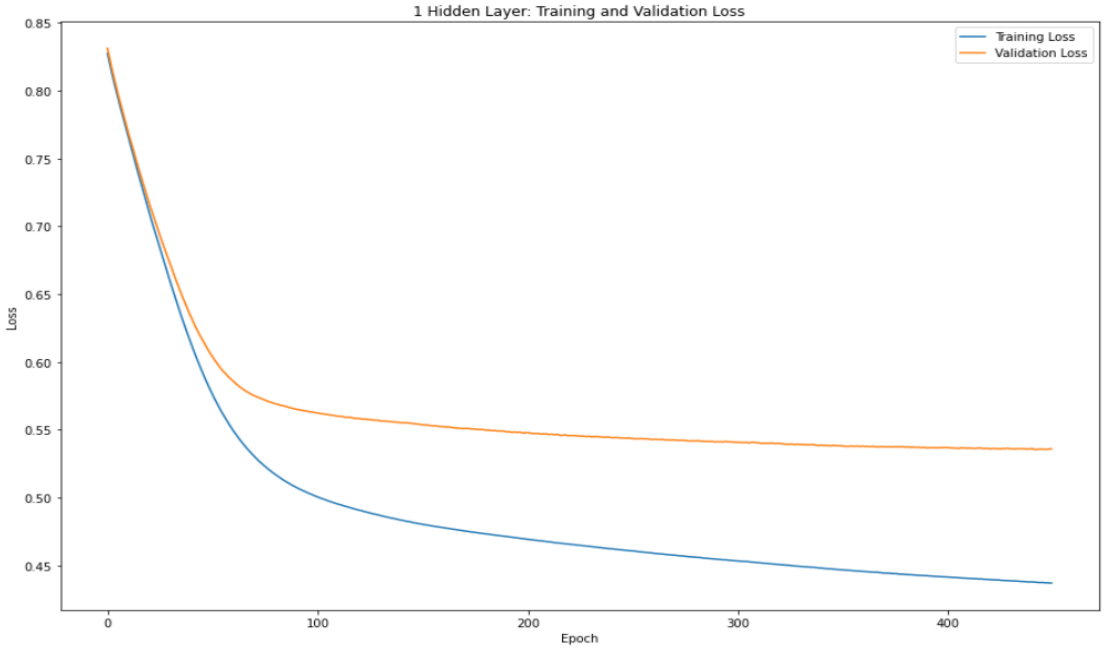
```
Loss 0.5031039118766785 Test Accuracy: 79.22
```



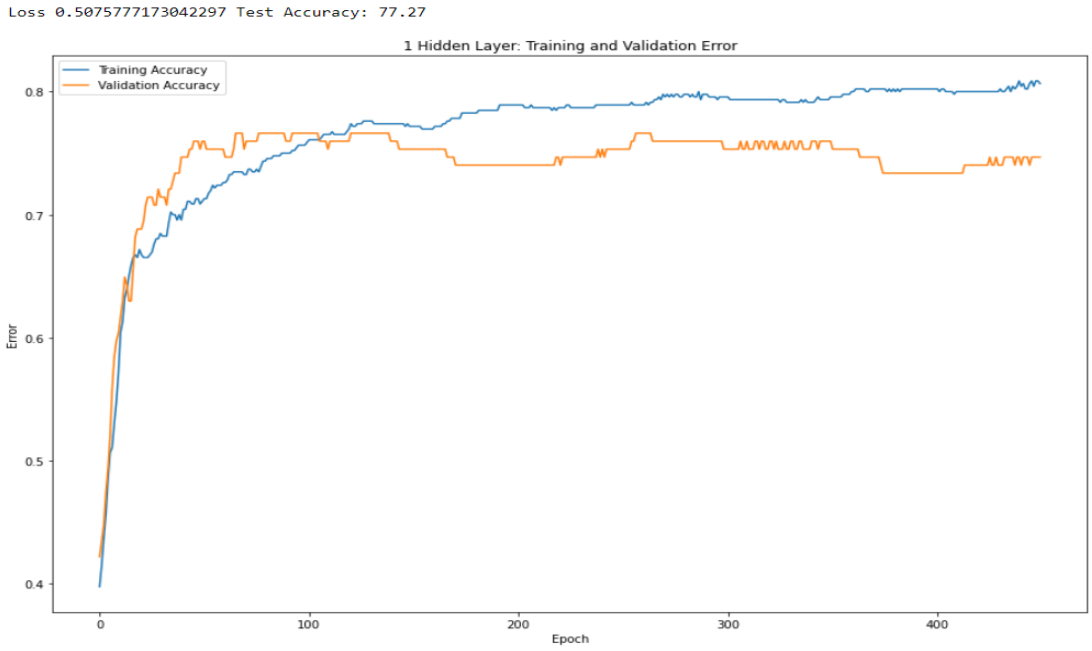
The test accuracy using L1 Regularization turns out to be 79.22%

For L2 Regularization:-

The train vs validation loss graph is as follows:



The graph for Validation vs Training accuracy:-



The test accuracy using L1 Regularization turns out to be 77.27%

We can infer from the above results that L1 Regularization predicts the output more accurately.

4.4 Neural Networks Implementation with Dropout(Part 3)

Dropout is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporarily removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.

As a neural network learns, neuron weights settle into their context within the network. Weights of neurons are tuned for specific features providing some specialization. Neighboring neurons rely on this specialization, which if taken too far can result in a fragile model too specialized to the training data. This reliance on context for a neuron during training is referred to as complex ***co-adaptations***.

Results:-

On running the same dataset with dropout regularization, the test accuracy turns out to be 80.1%. Thus, we can fairly say that using dropout in our model can help in getting the optimum performance on our model. However, we should be wary of the fact that due to the stochastic nature of the algorithm, the results may vary. So, we should run the model a few times to make sure that our conclusion follows correctly.

5 References

1) https://towardsdatascience.com/regularization-techniques-and-their-implementation-in-tensorflow-keras-c06e7551e709
2) https://www.machinecurve.com/index.php/2020/01/23/how-to-use-l1-l2-and-elastic-net-regularization-with-keras/
3) https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/
4) https://www.coursera.org/learn/machine-learning/home/welcome
5) https://en.wikipedia.org/wiki/Gradient_descent
6) https://www.geeksforgeeks.org/box-plot-in-python-using-matplotlib/
7) https://towardsdatascience.com/https-medium-com-chayankathuria-regression-why-mean-square-error-a8cad2a1c96f

8) https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance