Assignment 1

Sanyam Gupta (50359154)

Oct 10 2021

1. Overview

The goal here is to train a supervised machine learning model for a two class classification problem. The same has been achieved using logistic regression and neural network as the classifier. Leveraging the diagnostic measurements provided in the dataset, we aim to predict whether a patient has diabetes(class 1) or not (class 0). On the test Dataset, an accuracy of **79.8 %** is achieved using Logistic regression and accuracy of **77.9%** using a neural network with l2 regularization.

2. Dataset

The Pima Indians Diabetes Database has 768 samples. Here the diagnostic measurements include - Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction and Age. The data is partitioned 3 ways. Choosing 60% of the data for training, 20% for validation and remaining 20% for testing.

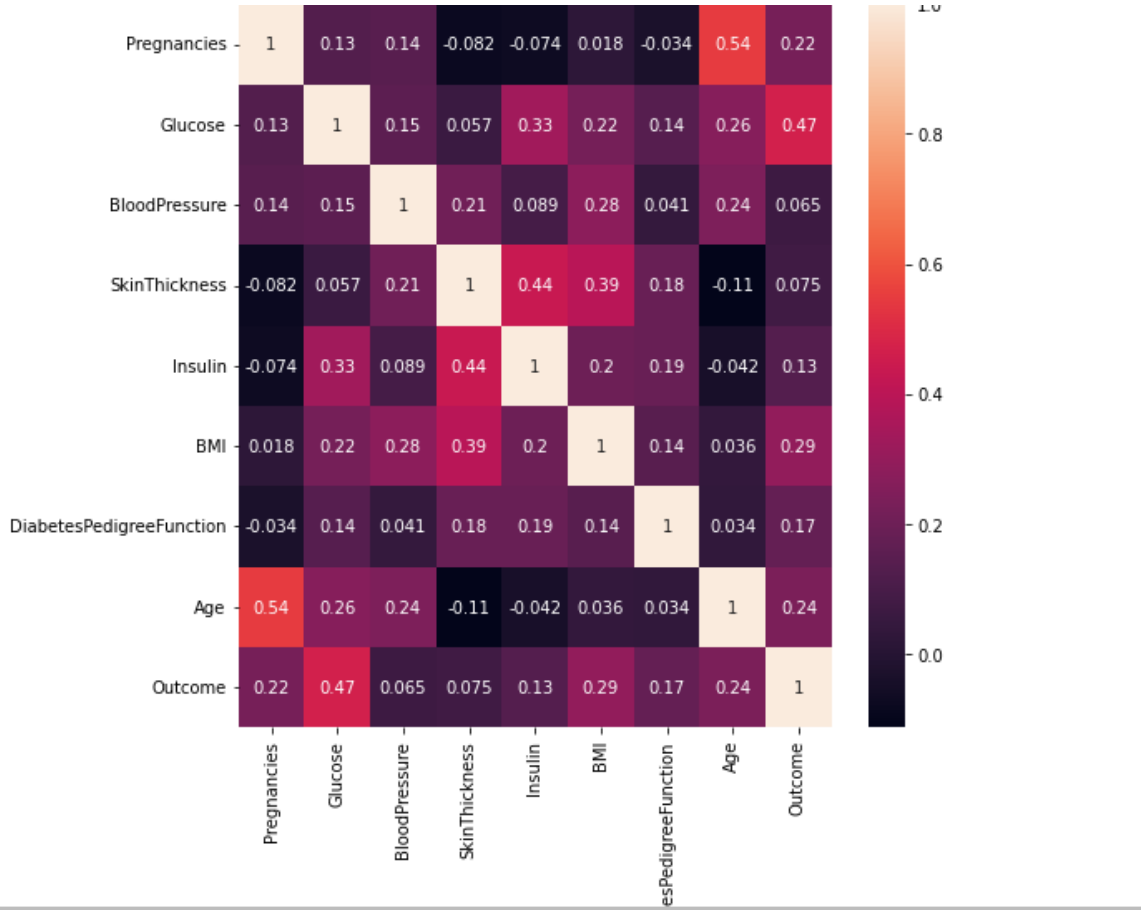

Figure 1 - Correlation matrix

3. Data Preprocessing

In order to normalize the dataset, I have calculated mean and standard deviation. From the entire dataset, I have subtracted the mean and divided it by the standard deviation.

4. Logistic Regression

4.1 Training the model

I have used different learning rates starting from .05 all the way down to 1e-12. For higher learning rates the number of iterations is significantly less as compared to the ones for lower learning rates. The rationale here is to take bigger steps quickly during initial phases and spend more time with small learning rates.

The bias is calculated separately from the weight vector. For the weight vector, I am initializing it to random values and bias is initialized to -6. This value of bias has been observed to give highest accuracy during the validation phase.

```
passInfo = [[.05, 500], [.005, 1000], [.001, 1000], [5e-4, 1000], [5e-6, 2000], [5e-8, 4000], [1e-10, 7000], [1e-12,
total_samples = train_x.shape[0]

#initial values of weights and bias
weights = np.random.randn(train_x.shape[1], 1)*.01
b =-6

#using differrent itration values for different learning rates
for info in passInfo:
    alpha = info[0]
    iterations = info[1]
    for i in range(iterations):
        p = sigmoid(train_x)

        #caluculating the gradient
        dz = p-train_y
        dw = (1 / total_samples) * np.dot(train_x.T, dz)
        db = (1 / total_samples) * np.sum(dz)

        #updating the parametes
        weights -= (alpha * dw)
        b -= (alpha * db)
```

Figure 2 - Training Logistic regression model using gradient descent.

4.2. Outcome

After training and hyper-parameter tuning the model has an accuracy of **79.87%** on the test dataset.

Step 6 - Verifying accuracy (79.87 %)

```
predicted_y = predict(test_x)
test_acc = check_accuracy(test_y)
print("accuracy on test data", test_acc)
```

```
accuracy on test data 79.87 %
```

Figure 3 - Accuracy on testing dataset

5. Netral Network

5.1 Training the model

I have constructed a 3 layer Neural Network with :

- regularization = l2
- lambda_val = .5
- learning_rate=.005
- hidden_layer_act='relu'
- output_layer_act='sigmoid'

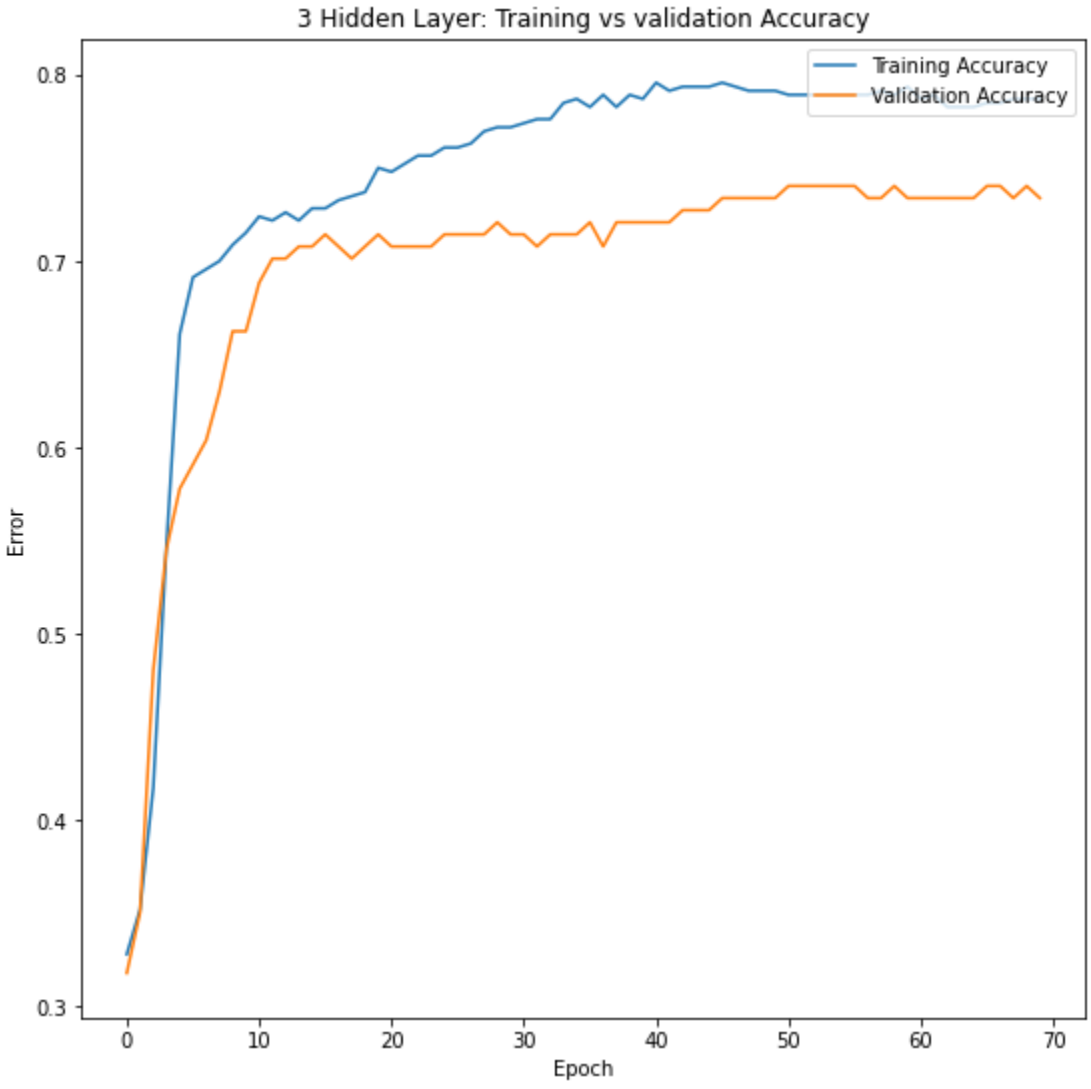

Figure 4 - Training vs validation Accuracy

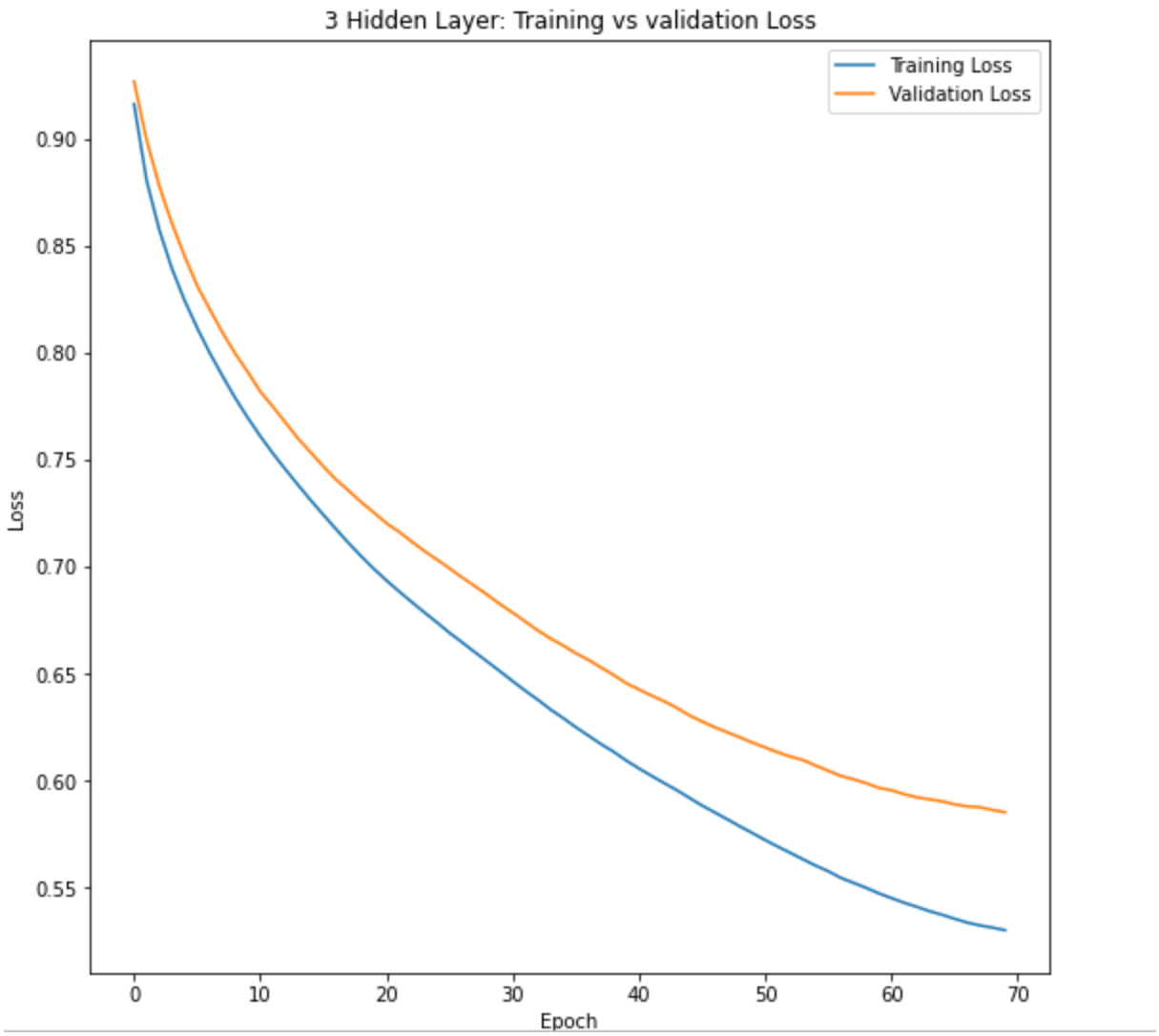

Figure 5 - Training vs Validation Loss

5.2 Result

After training and hyper-parameter tuning the model has an accuracy of **77.9%** on the testing dataset.

Step 4 - Result

```
prediction = model_l2.predict(test_x)
accuracy_l2 = accuracy_score(test_y, prediction.round())
print(f"Accuracy of test set is :", "{:.2f} %".format(accuracy_l2 * 100))
```

```
Accuracy of test set is : 77.92 %
```

Figure 6 - Accuracy on test dataset

6. Comparing different regularization methods.

On a comparison of the three regularization methods - l2, l1 and dropout. We can observe that Dropout has the lowest loss across all epochs.

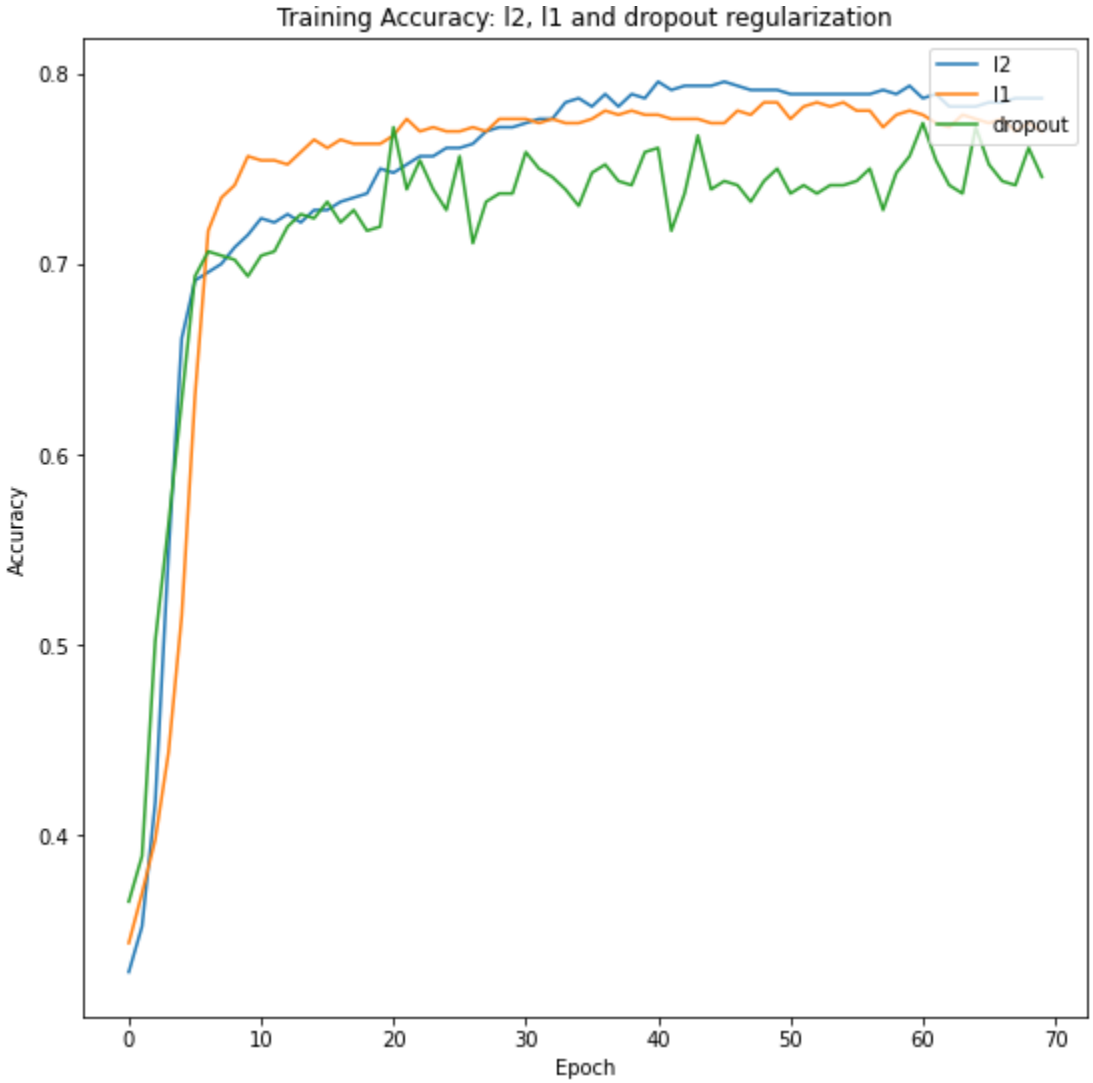

Figure 6 - Accuracy on Training dataset

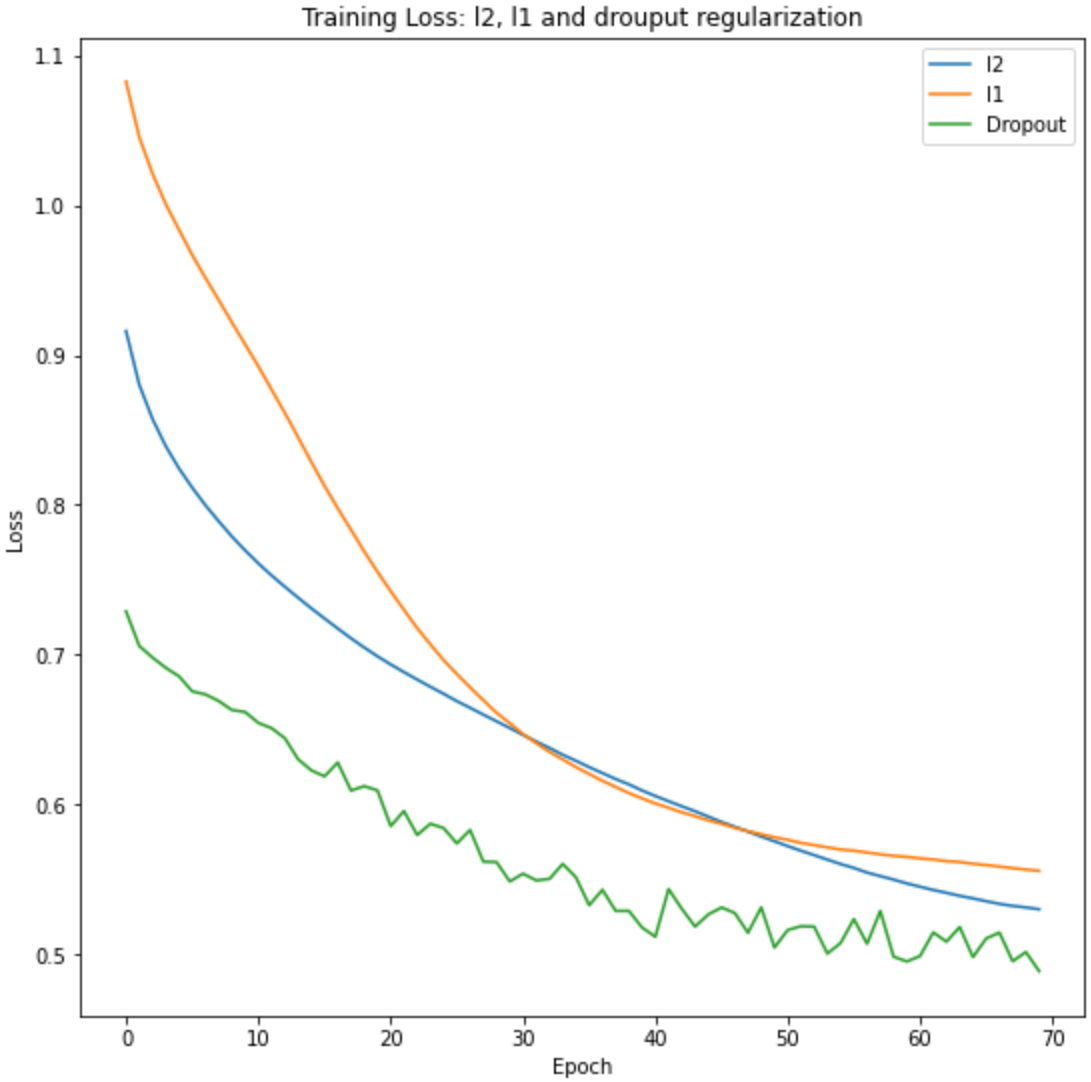

Figure 7 - Loss on training Dataset