Neural Networks

Group Members:

Sarthak Gaur - 2017A7PS0250H
Smit D Sheth - 2017A7PS1666H
Sourav Sanganeria - 2017A7PS1625H

Introduction:

This report focusses on the use of the neural network on predicting whether the house prices of a given dataset are above or below the median price given 10 features of the house. The neural network is built from scratch and supports a variable number of layers with a variable number of neurons.

Dataset:

The dataset *housepricedata.csv* consists of 1460 data points containing **10** input features and **1** binary output feature. These 10 input features are features of a house and are either **categorical** or **continuous**. The output feature is either 1 (denoting that the house price is above the median price) or 0 (denoting that the house price is below the median price). The dataset is ***normalized*** using the formula:

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Further, the dataset is split into **80% training data** and **20% test data.**

Architecture:

We tried to train the dataset using a **2-layered** neural network model and a **3-layered** neural network model. The models were trained using **stochastic gradient descent**.

Activation Functions Used:

1. Sigmoid: $y = \frac{1}{(1 + e^{-x})}$
2. Tanh: $y = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
3. Linear: $y = x$

4. ReLU: $y = max(0, x)$

Loss Function Used:

We used binary cross-entropy as the loss function for our classification model. The reason to use this loss function is that binary cross-entropy gives a convex loss curve in case of classification and reaching the minima is possible via gradient descent.
Binary Cross-entropy formula is given below:

$$L_n = -t_n \, log(y_n) - (1 - t_n) \, log(1 - y_n)$$

2-layer Neural Network Model:

We tried various combinations of activation functions, initialization of weights, and the number of neurons for the 2-layer neural network model. Some of the best results that we got are mentioned below:

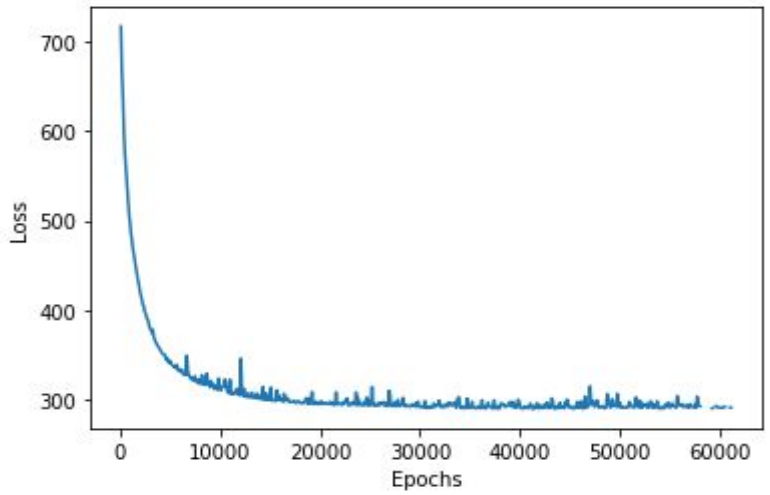
Model 1:
A NN with **5 neurons** in the hidden layer, initialization of weights as **uniform**, **ReLU** activation for the input layer, and **sigmoid** activation for the hidden layer.
After many trials, we found **learning rate = 0.01** to be the best for the model over **60000 iterations.**

This model provided good results as shown below:
Test set Accuracy in the range: **0.89-0.91**
Test set F-Score in the range: **0.90-0.91**

Result for a run using the above model:

```
Training Accuracy:  0.9016465
Test Accuracy :  0.90753424657534246
Test F-score:  0.9137380191693291
```

Loss vs. Epoch Graph:



Model 2:

A NN with **5 neurons** in the hidden layer, initialization of weights as **normal**, **ReLU** activation for the input layer, and **sigmoid** activation for the hidden layer.

After many trials, we found **learning rate = 0.01** to be the best for the model over **100000 iterations.**

This model provided results as shown below:

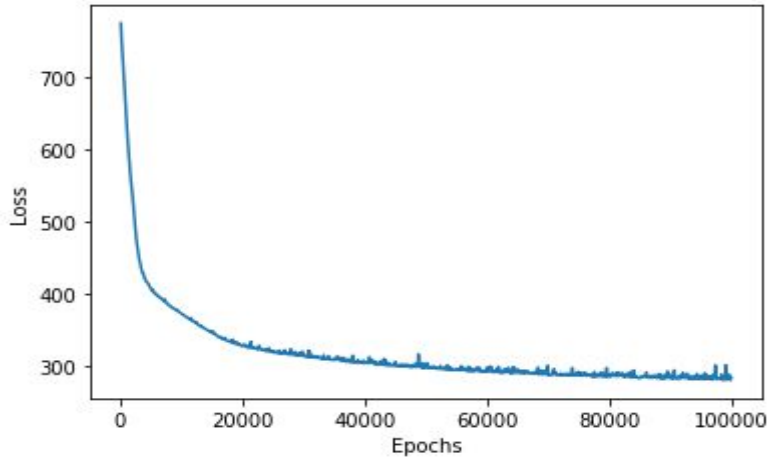
Test set Accuracy in the range: **0.88-0.90**

Test set F-Score in the range: **0.89-0.91**

Result for a run using the above model:

```
Epoch:  100000 Loss:  [302.91307187]
Training Accuracy:  0.904827
Test Accuracy :  0.8972602739726028
Test F-score:  0.9006622516556291
```

Loss vs. Epoch Graph:



3-layer Neural Network Model:

Next, we built 3-layer neural network models with 5 neurons in each hidden layer. We did not get as satisfying results as the 2-layer models. Some of the best results that we got are mentioned below:

Model 1:

A NN with **5 neurons** in each hidden layer, initialization of weights as **uniform**, **ReLU** activation after the input layer, and **sigmoid** activation for both the hidden layers.

After many trials, we found **learning rate = 0.004** to be the best for the model over **100000 iterations.**

This model provided results as shown below:

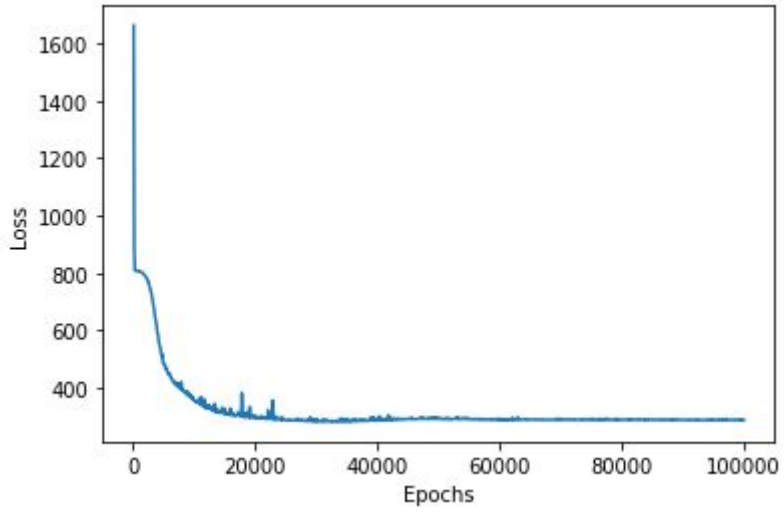
Test set Accuracy in the range: **0.87-0.90**

Test set F-Score in the range: **0.89-0.91**

Result for a run using the above model:

```
Epoch:  100000 Loss:  [287.4598713]
Training Accuracy:  0.88778
Test Accuracy :  0.8972602739726028
Test F-score:  0.8993288590604027
```

Loss vs. Epoch Graph:



Model 2:

A NN with **5 neurons** in the hidden layer, initialization of weights as **normal**, **ReLU** activation after the input layer, and **sigmoid** activation for both the hidden layers.

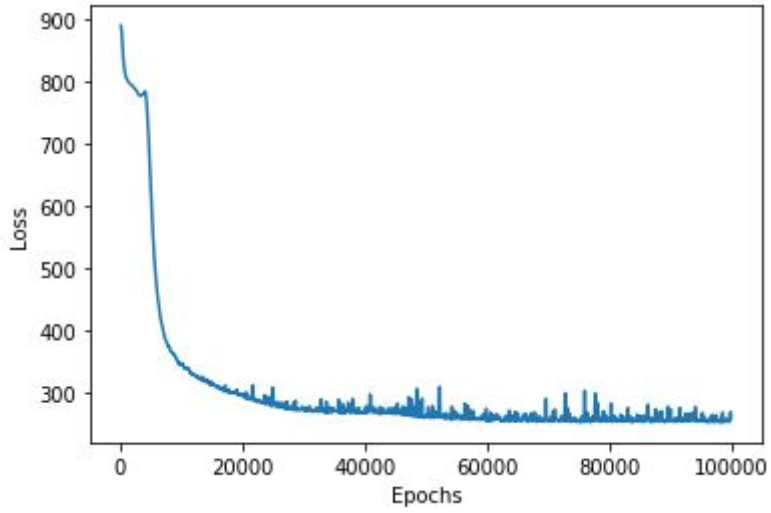
After many trials, we found **learning rate = 0.004** to be the best for the model over **100000 iterations.**

This model provided good results as shown below:
Test set Accuracy in the range: **0.86-0.88**
Test set F-Score in the range: **0.87-0.89**

Result for a run using the above model:

```
Epoch:  100000 Loss:  [260.74160868]
Training Accuracy:  0.8914385
Test Accuracy :  0.87671232876712328
Test F-score:  0.8867924528301888
```

Loss vs. Epoch Graph:



Conclusion:

Different initialization of weights led to the convergence of the model at different local minima. The best results were provided by the 2-layer NN model with the weights initialized uniformly. The reason for 3-layer to not work as expected maybe overfitting. Also, we had to decrease the learning rate in order to get better results from the 3-layer NN.