NEURAL NETWORKS

RESULTS -

1) 1 Hidden Layer

S.No.	Weight Initialization	Activation Function	Hidden Neurons	Epochs	Learning Rate	Accuracy (in %)	F-Score
1	Gaussian	Sigmoid	20	350	0.0005	86.64	0.8707
2	**Gaussian**	**Sigmoid**	**25**	**350**	**0.0005**	**86.98**	**0.8698**
3	Gaussian	tanh	20	300	0.0005	85.96	0.8591
4	Gaussian	tanh	15	300	0.0005	86.30	0.8631
5	Uniform	Sigmoid	15	600	0.0001	84.59	0.8524
6	Uniform	Sigmoid	15	600	0.0005	86.64	0.8687
7	Uniform	tanh	15	300	0.0005	86.99	0.8707
8	Uniform	tanh	20	300	0.0005	85.96	0.8619

2) 2 Hidden Layers

S.No.	Weight Initialization	Activation Functions	Hidden Neurons	Epochs	Learning Rate	Accuracy (in %)	F-Score
1	Gaussian	Sigmoid,Sigmoid	40,30	300	0.0001	85.96	0.8619
2	Gaussian	tanh, tanh	15,10	300	0.0001	85.96	0.8601
3	**Gaussian**	**tanh, Sigmoid**	**20,10**	**300**	**0.0001**	**86.64**	**0.8696**
4	Gaussian	Sigmoid, tanh	20,12	300	0.0001	84.93	0.8503
5	Uniform	Sigmoid,Sigmoid	20,15	300	0.0005	83.87	0.8456
6	Uniform	tanh, tanh	12,6	600	0.0005	85.62	0.8571
7	Uniform	tanh, Sigmoid	20,15	300	0.0003	84.58	0.8553
8	Uniform	Sigmoid, tanh	20,12	500	0.00005	84.38	0.8461

Algorithm Flow and Design Decisions -

We took the dataset of house prices and had to classify them into whether a house's price will be above or below the market's median price (binary classification problem).

We splitted the data into 80:20 ratio where training data contains 80% of the whole dataset while the testing data contains the rest 20%.

The class attribute was converted to one hot encoded vector with two columns, one of them is for class 1 and other is for class 0(depending upon the class label).

Weights were initialized in two ways - either Gaussian (Mean 0, Variance 1) or Uniform(0-1).

Activation functions (sigmoid, tanh and softmax) were defined separately.

```
def sigmoid(x):
  return 1/(1+np.exp(-x))

def softmax(x,y):
  return np.exp(x)/(np.exp(x)+np.exp(y))

def tanh(x):
  return 2*(sigmoid(2*x)) - 1
```

Then, training of the neural network started with feed forward and back propagation. We have used the Gradient descent algorithm for training the neural network model.

The Loss function used here is Cross-Entropy Loss Function, as our task is Binary Classification.

The graph of Loss vs Epochs has been plotted, which can be seen under the visualizations section.

And at last the model was evaluated on the basis of accuracy and F-score.

CONCLUSIONS-

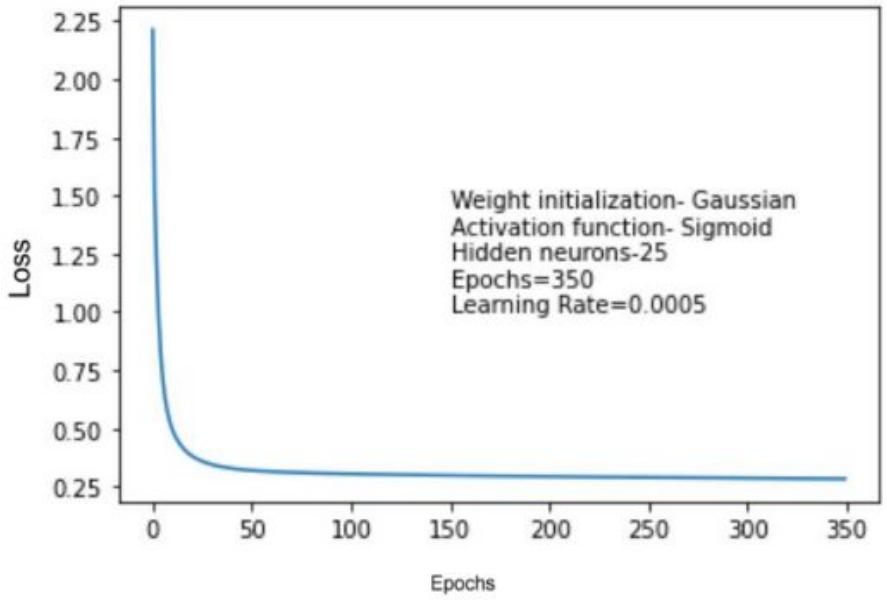
We trained different models and got the accuracy between 84% to 87%.

We also observed that uniform weights initialization required more iterations to train as compared to gaussian weights initialization. So, gaussian weights initialization suited our dataset.
We didn't get significant improvement in the accuracy by changing the number of hidden neurons, number of hidden layers or activation functions.

VISUALIZATIONS-

Graphs illustrating how the loss function's value decreases as training proceeds-

1) Single Hidden layer



2) Two Hidden Layers