# Body Fat Prediction Using Artificial Neural Networks in MATLAB

Mehuli Lahiri

***Abstract*—This paper presents a MATLAB-based implementation of an artificial neural network (ANN) to estimate body fat percentage from a set of physiological measurements. The bodyfat dataset built into MATLAB was utilized to train and test a feedforward neural network model. Key performance metrics, such as Mean Squared Error (MSE), regression analysis, and error histograms, were analyzed to evaluate model effectiveness. The trained model demonstrated strong prediction capabilities, indicating its potential utility in biomedical and health-related applications.**

***Index Terms*—Body Fat Prediction, Neural Network, MATLAB, Regression, Feedforward Network.**

## I. Introduction

Accurate body fat estimation is essential in healthcare and fitness industries. Traditional methods are either invasive or require expensive equipment. Hence, computational models like ANNs provide a non-invasive, low-cost alternative. This project explores the use of a feedforward neural network trained on anthropometric data to predict body fat percentage.

## II. Dataset Description

The bodyfat dataset in MATLAB contains:

- Total Samples: 252
- Input Features: 13 anthropometric measurements (e.g., age, height, abdomen circumference)
- Target Output: Body fat percentage
- The dataset is split internally during training using default MATLAB behavior: 70 training, 15 validation, 15 testing.

## III. Methodology

### A. Network Configuration

- Type: Feedforward Neural Network (fitnet)
- Hidden Layers: 1
- Number of Hidden Neurons: 10
- Training Algorithm: Levenberg–Marquardt (default)
- Activation Functions: tan-sigmoid (hidden), linear (output)

### B. MATLAB Implementation

```
[x, t] = bodyfat_dataset;
net = fitnet(10);
[net, tr] = train(net, x, t);
y = net(x);
view(net);
figure, plotperform(tr);
figure, plotregression(t, y);
figure, ploterrhist(t - y);
```

```
>> load bodyfat_dataset.mat
>> [x, t] = bodyfat_dataset;
>> net = fitnet(10);                       % 10 neurons in hidden layer
net.divideParam.trainRatio = 0.7;       % 70% training
net.divideParam.valRatio = 0.15;        % 15% validation
net.divideParam.testRatio = 0.15;       % 15% testing

[net, tr] = train(net, x, t);           % Train the network
y = net(x);                             % Predict using trained model
```

## IV. Performance Evaluation

### A. Performance Plot

- Displays mean squared error (MSE) over training, validation, and testing epochs.
- Shows clear convergence and learning behavior of the neural network.
- Best validation performance indicates the model's generalization ability.
- Differentiates overfitting if validation error starts increasing.

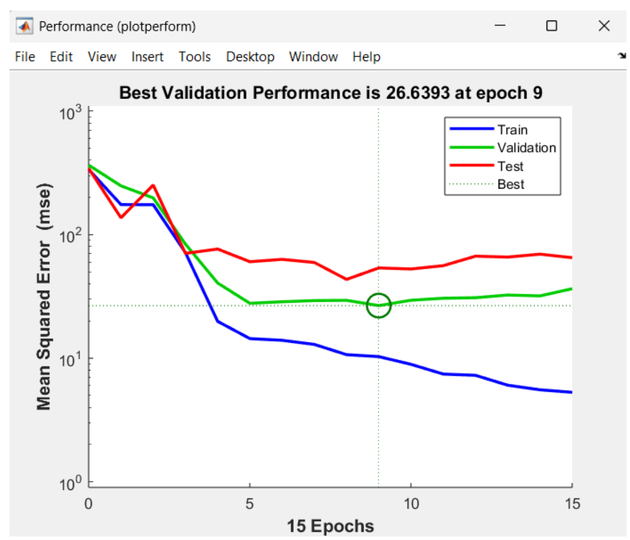


Fig. 1. Performance plot showing MSE over training, validation, and testing

### B. Regression Plot

- Shows predicted vs. actual body fat percentages for all data subsets.
- High R-values (close to 1) indicate strong linear correlation between output and target.
- Indicates the linearity and consistency of model predictions.

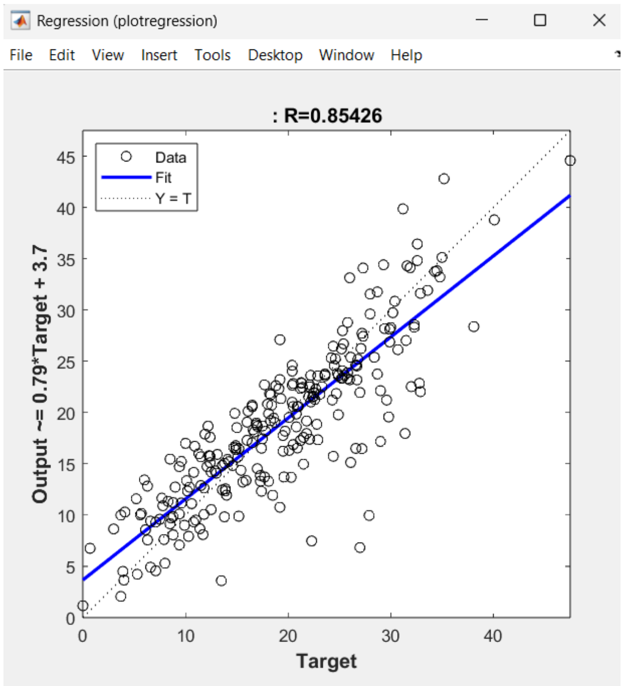


Fig. 2. Regression plot comparing predicted and target body fat percentages

### C. Error Histogram

- Highlights the distribution of prediction errors.
- Most errors centered around zero, indicating high model accuracy.

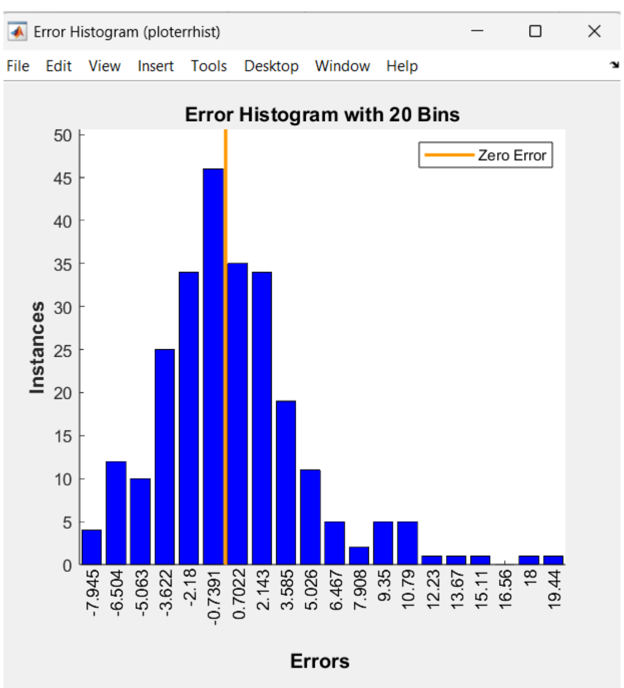


Fig. 3. Histogram of prediction errors across all samples

## V. Results and Discussion

The ANN model demonstrates strong predictive capability with:

- High R-value (correlation coefficient close to 1)
- Low mean squared error across datasets
- Narrow error distribution

This proves the effectiveness of neural networks in approximating complex, nonlinear functions like body fat estimation from anthropometric features.

## VI. Conclusion

This work demonstrates how MATLABs neural network toolbox can be effectively utilized to create a predictive model for body fat estimation. The network trained well, validated effectively, and showed accurate predictions with all major performance metrics confirming its reliability. Future work can include:

- Hyperparameter tuning
- Testing with other datasets
- Feature reduction for real-time applications