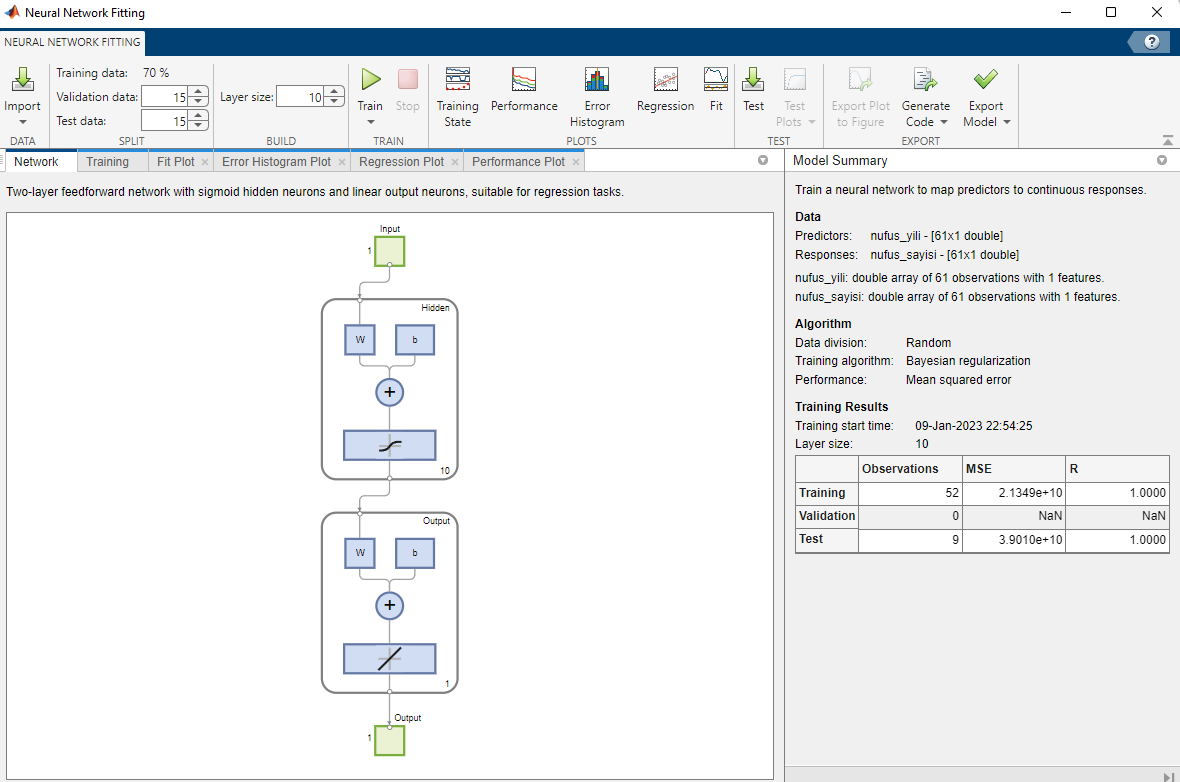
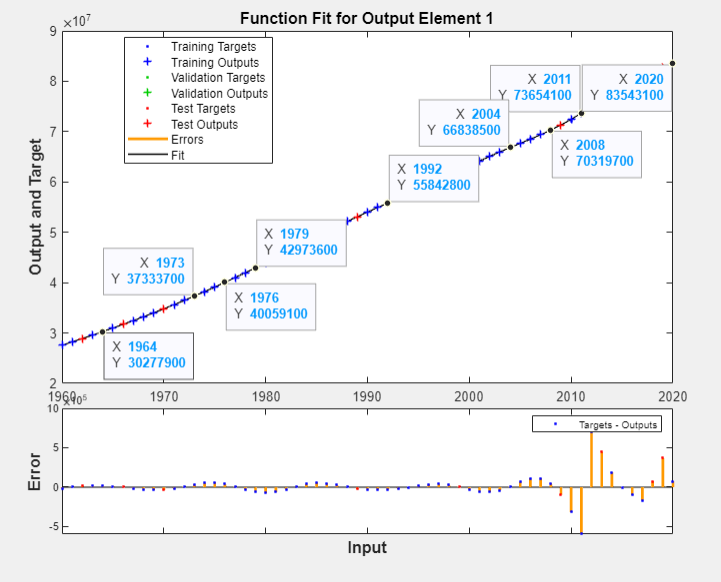
Neural Net Fitting Population Predictor By Years

Mert ERMAN

First of all I made an excel file to add population values over time. After That I opened them at matlab. After that I used Neural Net Fitting Tool to import my variables. I used %70 Trainig,%15 Validation,%15 Test Data to Train.The reason of 1 millon error is because of some datas in excel are stated NA in variables part I manually entered them in my matlab. When I train my function I exported them as function and saved in my script. Also I wrote a little part to take input from user. I add the process and my script down below.



tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**nufus = xlsread('nufus.xlsx');**

**nufus\_yili = nufus(:,2);**

**nufus\_sayisi = nufus(:,1);**

**a = input("Girilmesini İstediğiniz Yılı Giriniz:");**

**b= myNeuralNetworkFunction(a);**

**fprintf("%d Yılındaki Nüfus Miktarı %d",a,b)**

**function [y1] = myNeuralNetworkFunction(x1)**

**%MYNEURALNETWORKFUNCTION neural network simulation function.**

**%**

**% Auto-generated by MATLAB, 09-Jan-2023 22:59:25.**

**%**

**% [y1] = myNeuralNetworkFunction(x1) takes these arguments:**

**% x = Qx1 matrix, input #1**

**% and returns:**

**% y = Qx1 matrix, output #1**

**% where Q is the number of samples.**

**%#ok<\*RPMT0>**

**% ===== NEURAL NETWORK CONSTANTS =====**

**% Input 1**

**x1\_step1.xoffset = 1960;**

**x1\_step1.gain = 0.0333333333333333;**

**x1\_step1.ymin = -1;**

**% Layer 1**

**b1 = [-2.5772520111098069506;-0.053241297557413302943;1.179116981417726695;1.3464236104044895459;0.053242234437804257519;-0.05324225024030872222;0.033228056800464020326;-0.05324243351542445879;0.053242505232753860323;-0.053242206897644105135];**

**IW1\_1 = [3.5654654023846616973;0.28998100392276815462;-2.3409640859152940351;-3.5054497216029152362;-0.28998150882161061181;0.28998151719910220114;1.8398949416498309173;0.28998161400441613278;-0.28998165162604783163;0.28998149425458868977];**

**% Layer 2**

**b2 = 0.06891673047985234779;**

**LW2\_1 = [0.77813013635455641204 0.40163950676888132696 1.711583031229457097 -0.75426789675971295246 -0.40164075972402429882 0.40164078085404125718 0.4734450754169666209 0.40164102590132022552 -0.40164112178264010922 0.40164072290068014714];**

**% Output 1**

**y1\_step1.ymin = -1;**

**y1\_step1.gain = 3.56754618928808e-08;**

**y1\_step1.xoffset = 27553280;**

**% ===== SIMULATION ========**

**% Dimensions**

**Q = size(x1,1); % samples**

**% Input 1**

**x1 = x1';**

**xp1 = mapminmax\_apply(x1,x1\_step1);**

**% Layer 1**

**a1 = tansig\_apply(repmat(b1,1,Q) + IW1\_1\*xp1);**

**% Layer 2**

**a2 = repmat(b2,1,Q) + LW2\_1\*a1;**

**% Output 1**

**y1 = mapminmax\_reverse(a2,y1\_step1);**

**y1 = y1';**

**end**

**% ===== MODULE FUNCTIONS ========**

**% Map Minimum and Maximum Input Processing Function**

**function y = mapminmax\_apply(x,settings)**

**y = bsxfun(@minus,x,settings.xoffset);**

**y = bsxfun(@times,y,settings.gain);**

**y = bsxfun(@plus,y,settings.ymin);**

**end**

**% Sigmoid Symmetric Transfer Function**

**function a = tansig\_apply(n,~)**

**a = 2 ./ (1 + exp(-2\*n)) - 1;**

**end**

**% Map Minimum and Maximum Output Reverse-Processing Function**

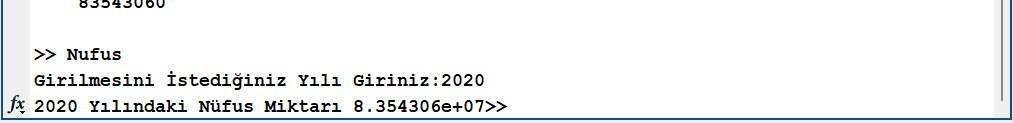
**function x = mapminmax\_reverse(y,settings)**

**x = bsxfun(@minus,y,settings.ymin);**

**x = bsxfun(@rdivide,x,settings.gain);**

**x = bsxfun(@plus,x,settings.xoffset);**

**end**



The script found the 2020’s population value : 83543060

Official Population value of 2020: 83614222

There is 0.08518% error value in this script.