INDIAN INSTITUTE OF TECHNOLOGY

KANPUR

Computational Method

AE 703A

"MACHINE LEARNING BASED EFFICIENT PREDICTION OF THE MECHANICAL RESPONSE IN LAMINATED COMPOSITE"

Under Guidance

Prof. Tanmoy Mukhopadhyay
Assistant Professor
Department of Aerospace Engineering

Submitted by

Anubhav Joshi
21101008
M.Tech (AE)

INDEX

Abstract

This paper is devoted to the development and construction of practical Machine Learning (ML) based model for the prediction of the deflection of the laminated composite rectangular plate. The solution of the deflection of a laminated composite rectangular plate when a uniform distributed load (UDL) is applied on it is given by a closed-form solution which depends on the material and geometric parameters. As the problem of prediction should be accurate, there are three ML models are used to predict the deflection. The three models used are Support Vector Machine (SVM), Neural Network Model for regression (NN) and Gaussian Process regression (GP) Models. The results showed that the ML models are computationally inexpensive and give accurate results. The value of the coefficient of determination (R^2) is 0.99 for the GP model, 0.98 for the NN model and 0.77 for the SVM model for specific input values. Different values of the dimensions and material property will affect the model and thus the model accuracy will change.

1. Introduction

Composite materials have wide application in many industries like aircraft, marine, sports and other engineering fields due to their high stiffness-to-weight ratio and high strength-to-weight ratio. But in some applications due to high load on the materials, the materials may deflect, buckle, bend and deform their shape and in these cases, the materials are unable to perform the desired function. Predicting the values of deflection on a plate for different materials and dimensions are a lengthy problem and computationally expensive to some extent. Therefore a machine learning model can be generated which helps us to predict the value of deflection for various input parameters. The performance of a machine learning model is of low-cost and if developed accurately, by ignoring the outliers and over fitting then highly accurate.

Theoretically, we have the expression of the deflection of the plate. To generate an ML algorithm we need the distribution of the data so that with the help of distribution we can predict the values. The distribution can be generated by discrete data (values) or by continuous data (values). Here we have a closed-form solution of the problem; we can generate the data from the closed-form solution and then see the distribution of the data which helps to predict. The whole data can be divided into 2 sets of data. One which contains around 75% of total data is called training data and the rest 25% data is called test data. The training data can be used to create or model the machine learning model. In our deflection problem the input parameters for the ML model as the material as Young's modulus of elasticity and geometric parameters like depth, width, length. The output parameter for the ML model is the deflection at any point.

Using the closed-form solution of the deflection of the plate, 200 data points are generated. Here are 3 features which are as follows

1. There are 200 random values of Young's modulus of elasticity in the range of 50 to 100GPa.
2. There are 200 random values of the length of the plate in a meter.
3. There are 200 random values of the width of the plate in a meter.

If we plot the random values of length and width, we can see the values of the corresponding deflection. We can plot the scattered data of length and width on the x, y-axis and plot the corresponding value of deflection on the z-axis. The data generated for the length and width are random numbers from 10 to 50. That means the length and width of the plate is any random value from 10 to 50 meters. On plotting the 2 features in the x and y-axis and getting the values on the z-axis we will get a plot like shown in the figure fig.1.

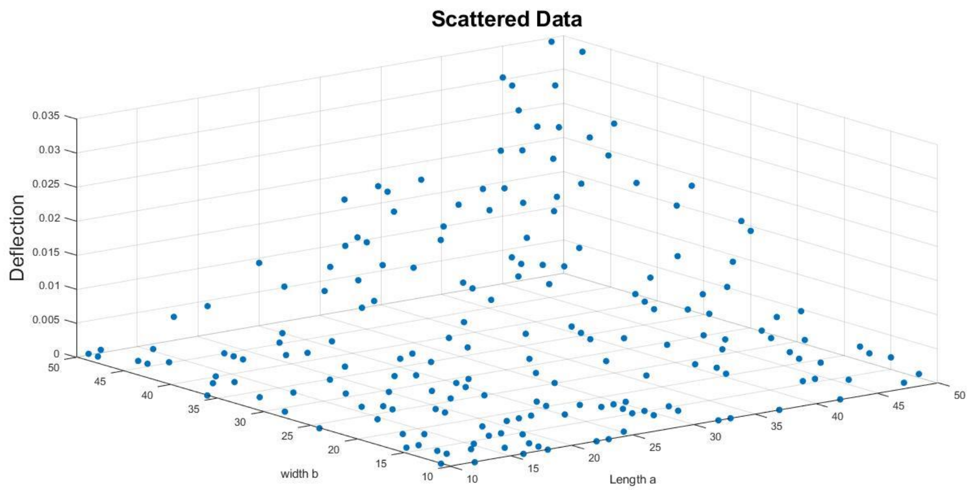

Fig.1.Random values of length and width of the plate with corresponding value of deflection

Data preprocessing is the first and most important step of any machine learning process. This helps us to do scaling of the features so that the model will perform accurately. Feature scaling is a method used to normalize the range of independent variables or features of data. There are various ways of doing the feature scaling the data one of them is standardization (Z-score Normalization). Feature standardization makes the values of each feature in the data have zero-mean and unit variance. This method is widely used for normalization in many machine learning algorithms like support vector machines, logistic regression and neural networks. The general method of calculating is to determine the distribution mean and for each feature.

The machine learning-based prediction has been generated using 3 models which will work to predict the value of the deflection. First one is support vector machine based model to predict the deflection, by the SVM model. In MATLAB Regression SVM is a support vector machine regression model. We can construct the model by using the keyword ‘fitrsvm’. Regression SVM models store data, parameter values, support vectors, and algorithmic implementation information. Later in the stage, we are implementing models like the Neural network and Gaussian process regression model to predict the deflection in the plate.

2. Deflection of Plate

A laminated composite plate can be considered a simple rectangular plate. Let the plate be simply supported and under a sinusoidal load. We can assume that the load distribution over the surface of the plate is given by the expression Eq1. For the calculation we have considered the value of UDL to be 400N/m^2.

$$q = q_0 \sin\frac{\pi x}{a} \sin\frac{\pi y}{b}$$

Equation 1

In the expression 'q_0' represent the intensity of the load at the center of the plate and 'a', 'b' represents the length, width of the plate. The differential equation for the deflection surface in this case becomes,

$$\frac{\partial^4 w}{dx^4} + 2\frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4} = \frac{q_0}{D} \sin\frac{\pi x}{a} \sin\frac{\pi y}{b}$$

Equation 2

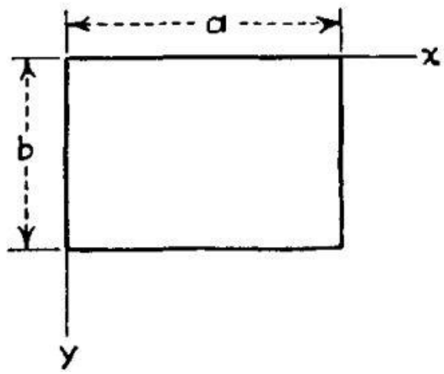

Fig.2. Laminated composite rectangular plate.

Where in the differential equation w represents the deflection and x, y and D can we determined by the equation [3]. Here 'E' represented the Young's modulus of elasticity in N/m^2; 'h' represents the depth of the plate and 'ϑ' represents the Poisson's ratio.

$$D = \frac{E * h^3}{12(1 - \vartheta^2)}$$

Equation 3

The boundary conditions for the simply supported edges are: w=0 M_x=0 for x=0 and x=a

w=0 M_y=0 for y=0 and y=b

w=0 $\frac{\partial y}{\partial x} = 0$ for x=0 and x=a

w=0 $\frac{\partial y}{\partial x} = 0$ for y=0 and y=b

If we put the boundary condition to determine the deflection we will get the final expression as,

$$w = \frac{16q_0}{\pi^6 D} \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} \frac{\sin\frac{m\pi x}{a} \sin\frac{n\pi y}{b}}{mn\left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right)^2}$$

Equation 4

Considering a, b the dimensions of the plate 'q_0' as uniform distributed transverse load and 'w' is the transverse deflection at co-ordinate (x, y). On taking a defined value of 'm' and 'n' we can write a MATLAB code to calculate the value of the deflection at any given coordinates (x, y).

MATLAB CODE: Functions file name deflection_term.m (link) by considering m=n=3 for calculating the deflection using the formula maintained in the equation [4].

```
function [def_val]=deflection_term(a,b,E,h,v,m,n,q,x,y)
                  % a,b,h are dimension of plate
                  % E is Youngs modulus,v is poissions ratio,q is UDL
                  % m=n=3
                  % x,y are data
D=(E*h^3)/(12*(1-v));
def=0;
sum=0;
% Calculation of A and B for the calculation of deflection
for i =1:n
    val_n=i;
    for j=1:m
        val_m=j;
        A=sin((val_m.*pi.*x)./a).*sin((val_n.*pi.*y)./b);
        B=m*n*((val_m^2)./(a.^2)+(val_n^2)./(b.^2)).^2;

        % Deflection
        def=(16*q.*A)./(pi^6*D.*B);
        sum=sum+def;
    end
end
def_val=sum;
end
```

Here a, b, E is the 3 feature and thus they are vector of size 200*1 containing the training and test data. These features are the input parameters and the output parameter is the deflection which is calculated at the specified co-ordinate x and y. The def_val is an output variable which is a deflection of the plate in meter for the corresponding input parameter. It is also a 200*1 vector for the given case.

3. Machine Learning Models

There are three models of machine learning for regression that is used to predict the deflection at specified co-ordinate of the laminated composite plate. The data which is generated from the random values of the dimension of the plate is first pre-processed to get normalized data or standard data for further calculation in the ML model. The technique is to rescale the feature value with the distribution value between 0 and 1 is useful for the optimization algorithm. The equation for standardization the data is,

$$x_{stand} = \frac{x - mean(x)}{standard\ deviation\ (x)}$$

Equation 5

3.1 Preparing data set

- Generating data: There are 3 features which are randomly generated, length and width of the plate is any value from 10 to 50 meter. The value of 'E' is random value from 50 to 100GPa.

MATLAB CODE:

```
E=randi([50,100]*10^9,200,1);     % Young's modulus
% Taking random value of a and b which are dimensions of plate
a=randi([10,50],200,1);
b=randi([10,50],200,1);
```

- Feature scaling: Feature scaling of data, the data is first standardized and then the total data is split into the training and test set. There is no missing data here because we have generated the data from the closed form solution of deflection.

MATLAB CODE:

```
%% Feature scaling
def=(def-mean(def))./(std(def));% Feature scaling for deflection
a=(a-mean(a))./(std(a));         % Feature scaling for dimension a
b=(b-mean(b))./(std(b));         % Feature scaling for dimension b
E=(E-mean(E))./(std(E));
```

- Splitting dataset: Splitting the data into the training data and test data. About 75% of the data is the training data and the 25% of the data is the test data.

  MATLAB CODE:

```
%% Spliting the data
train_x=[a(1:150) b(1:150) E(1:150)];% 2 features(dimension of
plate), 150 training data
test_x=[a(151:200) b(151:200) E(151:200)];% 50 test data
train_y=def(1:150);  %  150 training data (deflection)
test_y=def(151:200); % Test data (True value)
```

3.2 Regression models

3.2.1 SVM Regression model: - Support Vector Regression is a supervised learning algorithm that is used to predict discrete values. Support Vector Regression uses the same principle as the SVMs. The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyper plane that has the maximum number of points. In MATLAB there is a predefined keyword 'fitrsvm' which fits the support vector regression model to the moderate-dimensional prediction dataset. 'fitrsvm' supports mapping the predictor data using kernel functions, and supports $L1$ soft-margin minimization via quadratic programming for objective-function minimization.

Later we have also checked whether the model has converged or not. If the model converges then it predicts 1 or else it predicts 0.

MATLAB CODE:

```
% SVM model
Md_SVM=fitrsvm(train_x,train_y);
Md_SVM.ConvergenceInfo.Converged;
... cross vaidation (KFold using 5 folds)
Md_svm_CV=fitrsvm(train_x,train_y,'Standardize',true,'KFold',10);
```

Cross-validation is a process of resembling the data set and predicting the results for number for time to get the generalized solution of the problem. Train data is used to train the model and the unseen test data is used for prediction. If the model performs well over the test data and gives good accuracy, it means the model hasn't over fitted the training data and can be used for prediction.

3.2.2 Neural Network model: - The Neural Network model can work on both supervised and unsupervised learning problems. Here we have a supervised learning problem, and the NN model is a type of deep learning model which works with the help of nets (neural nets). In MATLAB, we need to use the predefined keyword 'fitrnet' to predict deflection using neural networks. In the neural nets, there are connections between the inputs (predictor input) in this case, inputs are the dimensions of the plate and the material of the plate, to the functions at the different layers which are learned itself (implicitly).

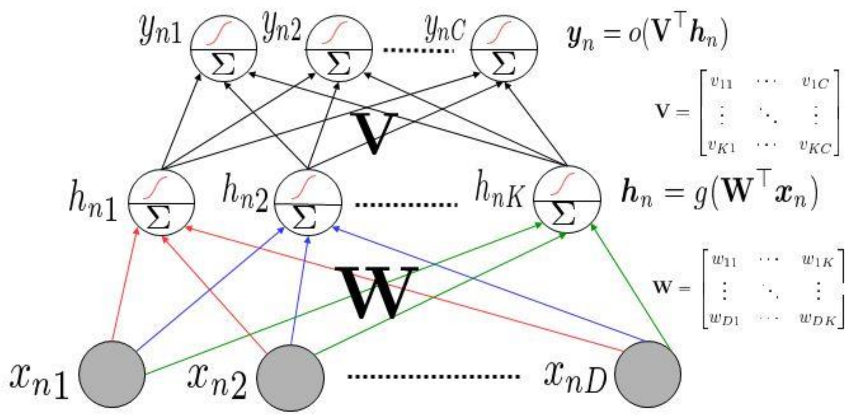

Fig.3. Deep Learning: Neural Network

MATLAB CODE:

```
% Neural Network Model
Md_NN=fitrnet(train_x,train_y,'Standardize',true);
```

3.2.3 Gaussian Process Regression model: - Gaussian process regression (GPR) models are nonparametric kernel-based probabilistic models. In which we can train a GPR model using the 'fitrgp' function in MATLAB. In the GP model the value of deflection which is unknown is determined by the linear regression model

$$y = x^T \beta + \epsilon$$

Where $\epsilon \sim N(0, \sigma^2)$, the error variance σ^2 and the coefficients β are estimated from the data. 'fitrgp' estimates the basis function coefficients, β, noise variance, σ^2 and the hyper parameter θ of the kernel function from the data while training the GPR model.

MATLAB CODE:

```
% GP model
Md_gp=fitrgp(train_x,train_y);
```

4. Results and discussion

In this paper, criteria involving RSS, R^2 are used to evaluate the performance of the algorithm. These terms can be defined as follow:

Residual sum of Squares (RSS):- It is a statistical technique used to measure the amount of variance in a data set that is not explained by a regression model itself. Instead, it estimates the variance in the residuals, or error term. Linear regression is a measurement that helps determine the strength of the relationship between a dependent variable and one or more other factors, known as independent or explanatory variables. The smaller the residual sum of squares, the better is the model fits the data; the greater the residual sum of squares, the poorer the model fits the data.

Coefficient of determination (R^2):- R^2 value gives the information about the goodness of fit of a model in regression. An R^2 of 1 indicates that the regression predictions perfectly fit the data. If a wrong model is chosen to predict the value of unknown then the R^2 value may exceed to 1.

R^2 value for SVM= 0.7333

R^2 value for NN= 0.9941

R^2 value for GPR= 0.9999

4.1 Predictions

Once the model of the deflection data prepared we can predict the unseen data known as the test data. So now we do have 3 models and now we want to predict the value of deflection. We can use a function 'predict()' in the MATLAB which will predict the value of the test data. With the prediction of the test data we are also calculating the RSS and R^2 values to define the accuracy of the model prediction.

4.1.1 SVM Regression model: - We are predicting the values of the test set which contain 50 data points. We can plot the graph between the true deflection value which are in the 'test_y' and the predicted deflection value. The true model should have the values at line$y = x$. Which mean for the true model we will have the predicted value is equal to the true value and get a line on x , y plan with 45^0 angle. For SVM model we have R^2 value 0.733.

MATLAB CODE:

```
ypred_SVM=predict(Md_SVM,test_x);
RSS_SVM=sum((test_y-ypred_SVM).^2);
RMSE_SVM=sqrt(RSS_SVM/length(ypred_SVM));
R2_SVM=1-(sum((test_y-ypred_SVM).^2))/(sum((test_y-
mean(test_y)).^2))
```

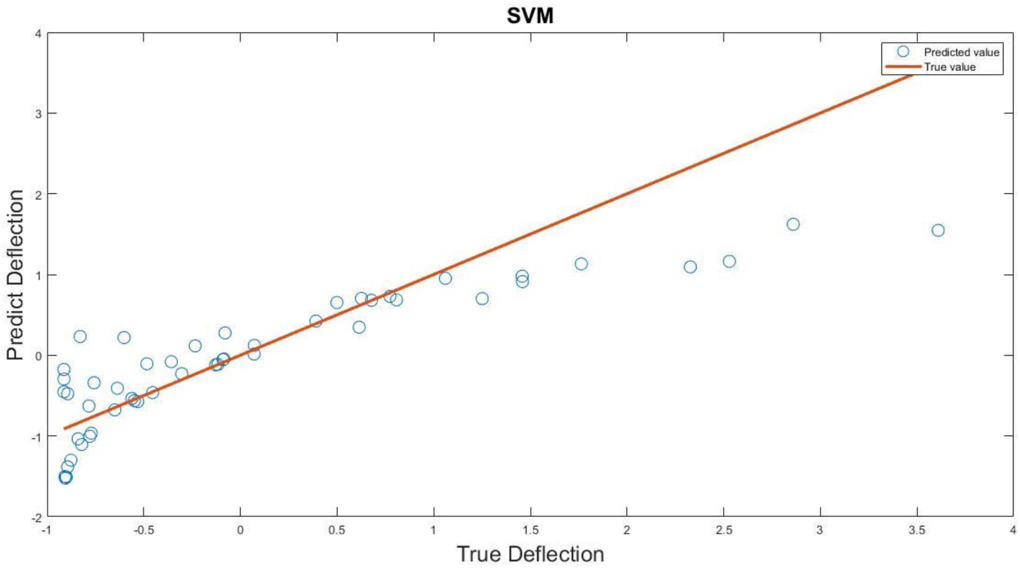

Fig.3.Plot: predicting the deflection by SVM model

4.1.2 Neural Network model: - We are predicting the values of the test set which contain 50 data points. We can plot the graph between the true deflection value which are in the 'test_y' and the predicted deflection value. The true model should have the values at line$y = x$. Which mean for the true model we will have the predicted value is equal to the true value and get a line on x , y plan with 45^0 angle. For NN model we have R^2 value 0.9941.

MATLAB CODE:

```
ypred_NN=predict(Md_NN,test_x);
RSS_NN=sum((test_y-ypred_NN).^2);
RMSE_NN=sqrt(RSS_NN/length(ypred_NN));
R2_NN=1-(sum((test_y-ypred_NN).^2))/(sum((test_y-
mean(test_y)).^2))
```

4.1.3 Gaussian Process Regression model: - We are predicting the values of the test set which contain 50 data points. We can plot the graph between the true deflection value which are in the 'test_y' and the predicted deflection value. The true model should have the values at line$y = x$. Which mean for the true model we will have the predicted value is equal to the true value and get a line on x , y plan with 45^0 angle. For NN model we have R^2 value 0.9999.

MATLAB CODE:

```
ypred_gp=predict(Md_gp,test_x);
RSS_gp=sum((test_y-ypred_gp).^2);
RMSE_gp=sqrt(RSS_gp/length(ypred_gp));
R2_gp=1-(sum((test_y-ypred_gp).^2))/(sum((test_y-
mean(test_y)).^2))
```

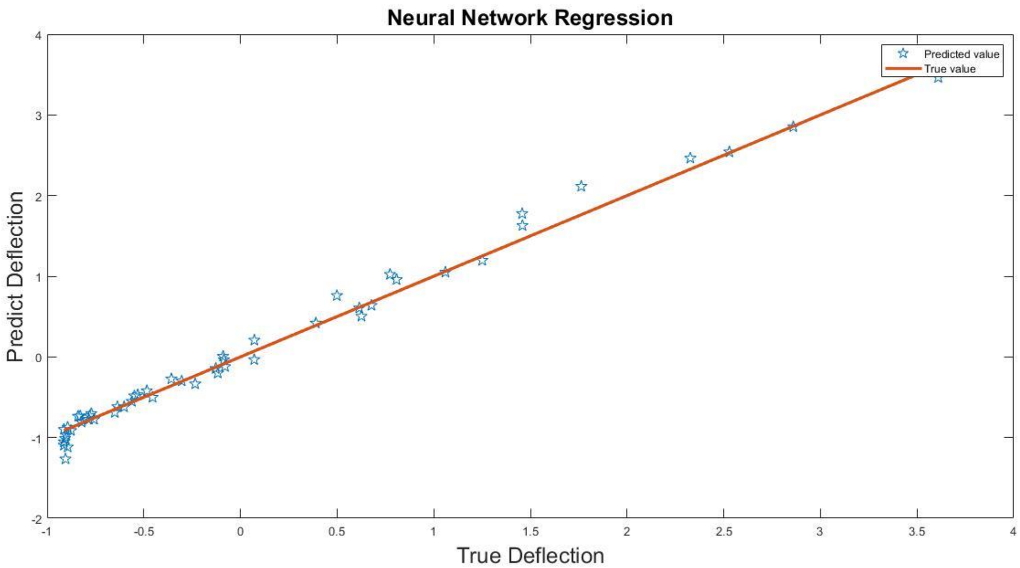

Fig.4. Plot: predicting the deflection by Neural Network model

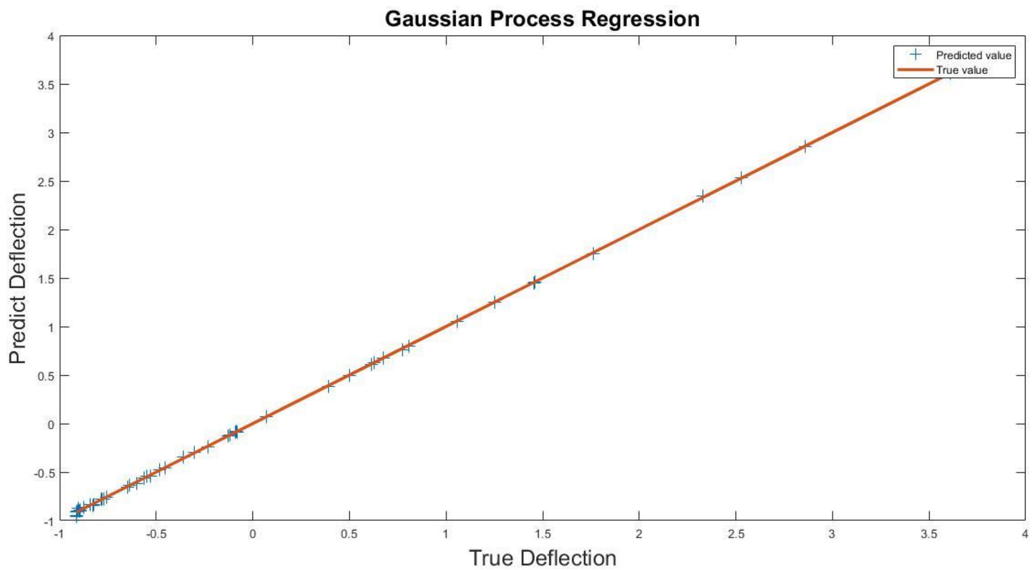

Fig.5. Plot: Predicting the deflection by Gaussian Process Regression

5. Conclusions

This study presents the machine learning-based prediction of laminated composite plates of different sizes. There are 3 models used to predict the deflection where the Gaussian process regression model gives the best result for the solution. The results showed that the ML models are computationally inexpensive and give accurate results. The value of the coefficient of determination (R2) is 0.99 for the GP model, 0.98 for the NN model and 0.77 for the SVM model for specific input values. Different values of the dimensions and material properties will affect the model and thus the model accuracy will change a little. By extending the sample points from 200 to 2000 and dividing 80% of the training data and the rest of the test data the results are more accurate and the Gaussian process (GP) model is still the best model to predict the deflection.

Reference

1. A zip file with compete code is uploaded on Google drive (Link).
   https://drive.google.com/file/d/1wsFrwUpMHpjWHU0c87Ge5ojmew4W85lY/view?usp=sharing
2. Oksana Choporova, Andrey Lisnyak: Using Machine Learning to Predict the Stress-strain State of a Rectangular Plate with a Circular Cut-out.
3. O.V. CHOPOROVA, S.V. CHOPOROV: Using machine learning to predict the stress-strain state of a circular plate. 2021-10-04.
4. Huang Zhongmin, Xie Zhen, Zhang Yishen, Peng Linxin: Deflection-bending moment coupling neural network method for the bending problem of thin plates with in-plane stiffness gradient. Chinese Journal of Theoretical and Applied Mechanics, 2021, 53(9): 2541-2553 doi: 10.6052/0459-1879-21-273
5. Sy-Ngoc Nguyen, Chien Truong-Quoc, Jang-woo Han, Sunyoung Im & Maenghyo Cho: Neural network-based prediction of the long-term time-dependent mechanical behavior of laminated composite plates with arbitrary hygrothermal effects. Published: 02 October 2021.

Complete MATLAB code

```
clear all;
clc;

%% Name: Anubhav Joshi
% Roll No: 21101008

%% Data
E=2*10^9;     % Youngs modulus
h=1;          % depth
v=0.25;       % poissons ratio
q=4000;       % UDL in N/m^2.
x=10;         % x coordinate
y=10;         % y coordinate
m=3;          % Given m=n=3
n=3;
% Taking random value of a and b which are dimensions of plate
a=randi([10,50],200,1);
b=randi([10,50],200,1);

% Calcuating the value of deflection
def=deflection_term(a,b,E,h,v,m,n,q,x,y);

% Plot the scattered data dimensions of plate and deflection
figure(4);
scatter3(a,b,def,'filled');
xlabel('Length a');
ylabel('width b');
zlabel('Deflection','Fontsize',18);
title('Scattered Data','Fontsize',20);

%% Feature scaling
def=(def-mean(def))./(std(def));   % Feature scaling for deflection
a=(a-mean(a))./(std(a));           % Feature scaling for dimension a
b=(b-mean(b))./(std(b));           % Feature scaling for dimension b

%% Spliting the data
train_x=[a(1:150) b(1:150)];      % 2 features(dimension of plate), 150
training data
test_x=[a(151:200) b(151:200)];   % 50 test data
train_y=def(1:150);               %  150 training data (deflection)
test_y=def(151:200);              % Test data (True value)

%% Models

% SVM model
Md_SVM=fitrsvm(train_x,train_y);
Md_SVM.ConvergenceInfo.Converged;
... cross vaidation (KFold using 5 folds)
Md_svm_CV=fitrsvm(train_x,train_y,'Standardize',true,'KFold',10);

% GP model
Md_gp=fitrgp(train_x,train_y);
```

```
% Neural Network Model
Md_NN=fitrnet(train_x,train_y,'Standardize',true);

%% Prediction and RMSE Calculation

%SVM
ypred_SVM=predict(Md_SVM,test_x);
RSS_SVM=sum((test_y-ypred_SVM).^2);
RMSE_SVM=sqrt(RSS_SVM/length(ypred_SVM));
R2_SVM=1-(sum((test_y-ypred_SVM).^2))/(sum((test_y-mean(test_y)).^2))

% GP
ypred_gp=predict(Md_gp,test_x);
RSS_gp=sum((test_y-ypred_gp).^2);
RMSE_gp=sqrt(RSS_gp/length(ypred_gp));
R2_gp=1-(sum((test_y-ypred_gp).^2))/(sum((test_y-mean(test_y)).^2))

% NN
ypred_NN=predict(Md_NN,test_x);
RSS_NN=sum((test_y-ypred_NN).^2);
RMSE_NN=sqrt(RSS_NN/length(ypred_NN));
R2_NN=1-(sum((test_y-ypred_NN).^2))/(sum((test_y-mean(test_y)).^2))

%% Plot

% Plot predicted value by SVM model
figure(1);
plot(test_y,ypred_SVM,'o');
hold on;
plot(test_y,test_y);
xlabel('True Deflection','Fontsize',18);
ylabel('Predict Deflection','Fontsize',18);
legend('Predicted value','True value');
title('SVM','Fontsize',18);

% Plot predicted value by Gaussian process Regression
figure(2);
plot(test_y,ypred_gp,'+');
hold on;
plot(test_y,test_y);
xlabel('True Deflection','Fontsize',18);
ylabel('Predict Deflection','Fontsize',18);
legend('Predicted value','True value');
title('Gaussian Process Regression','Fontsize',18);

% Plot predicted value by Neural Network Regression
figure(3);
plot(test_y,ypred_NN,'p');
hold on;
plot(test_y,test_y);
xlabel('True Deflection','Fontsize',18);
ylabel('Predict Deflection','Fontsize',18);
legend('Predicted value','True value');
title('Neural Network Regression','Fontsize',18);
```

```
%% Function for calculating the deflection
function [def_val]=deflection_term(a,b,E,h,v,m,n,q,x,y)
                              % a,b,h are dimension of plate
                              % E is Youngs modulus,v is poissions ratio,q is
UDL
                              % m=n=3
                              % x,y are data
D=(E*h^3)/(12*(1-v));
def=0;
sum=0;
% Calculation of A and B for the calculation of deflection
for i =1:n
    val_n=i;
    for j=1:m
        val_m=j;
        A=sin((val_m.*pi.*x)./a).*sin((val_n.*pi.*y)./b);
        B=m*n*((val_m^2)./(a.^2)+(val_n^2)./(b.^2)).^2;

        % Deflection
        def=(16*q.*A)./(pi^6*D.*B);
        sum=sum+def;
    end
end
def_val=sum;
end
```