Experiment-1

Aim: Implement the Naive Bayes learning algorithm on Balance Scale Dataset.

Code:

```
Mdl = fitcnb(BCDE,A)
[label,Posterior,Cost] = predict(Mdl,row)
[xx1, xx2] = meshgrid(4:.01:8,2:.01:4.5);
XGrid = [xx1(:) xx2(:)];
figure('Units','Normalized','Position',[0.25,0.55,0.4,0.35]);
sz = size(xx1);
```

Output:

```
Mdl =

ClassificationNaiveBayes
PredictorNames: {'VarName2'  'VarName3'  'VarName4'  'VarName5'}
ResponseName: 'B'
CategoricalPredictors: []
ClassNames: [B    L    R]
ScoreTransform: 'none'
NumObservations: 625
DistributionNames: {'normal'  'normal'  'normal'  'normal'}
DistributionParameters: {3×4 cell}


  Properties, Methods


label =

  2×1 categorical array

     L
     R


Posterior =

    0.2723    0.3638    0.3638
    0.2108    0.2291    0.5601


Cost =

    0.7277    0.6362    0.6362
    0.7892    0.7709    0.4399
```

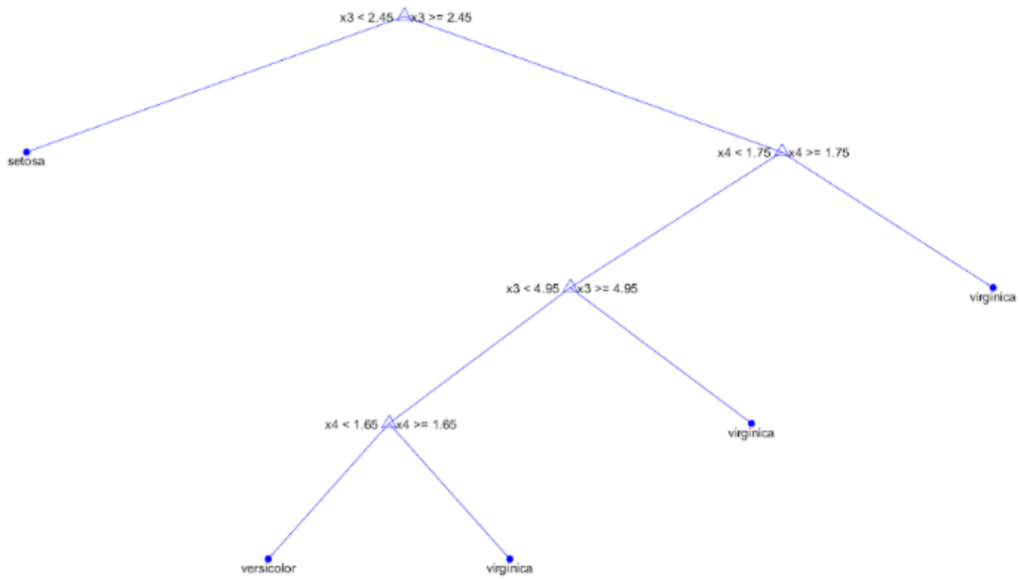
Experiment-2

Aim: Classify the iris dataset using Decision Tree learning algorithm. Calculate the classification accuracy using 5-fold cross validation.

Code:

```
load fisheriris
Mdl = fitctree(meas,species);
CVMdl = crossval(Mdl,'Kfold',5);
L = kfoldLoss(CVMdl);
%view(Mdl)
view(Mdl,'Mode','graph');
```

Output:



Command Window

```
>> decision_tree
>> Mdl

Mdl =

  ClassificationTree
             ResponseName: 'Y'
    CategoricalPredictors: []
               ClassNames: {'setosa'  'versicolor'  'virginica'}
           ScoreTransform: 'none'
          NumObservations: 150


  Properties, Methods

>> CVMdl

CVMdl =

  classreg.learning.partition.ClassificationPartitionedModel
    CrossValidatedModel: 'Tree'
         PredictorNames: {'x1'  'x2'  'x3'  'x4'}
           ResponseName: 'Y'
        NumObservations: 150
                  KFold: 5
              Partition: [1x1 cvpartition]
             ClassNames: {'setosa'  'versicolor'  'virginica'}
         ScoreTransform: 'none'


  Properties, Methods

>> L

L =

    0.0533
```

Experiment-3

Aim: Implement the K-means algorithm and apply it on any two datasets. Evaluate performance by measuring the Euclidean distance of each example from its class center. Test the performance of algorithm as a function of k.

Code:

1) Fisheriris Dataset

```
load fisheriris
X = meas(:,3:4);
figure;
plot(X(:,1),X(:,2),'k*','MarkerSize',5);
title 'Fisher''s Iris Data';
xlabel 'Petal Lengths (cm)';
ylabel 'Petal Widths (cm)';
rng(1); % For reproducibility
eva = evalclusters(meas,'kmeans','CalinskiHarabasz','KList',
[1:4]);
[idx,C,sumD,D] = kmeans(X,3);
x1 = min(X(:,1)):0.01:max(X(:,1));
x2 = min(X(:,2)):0.01:max(X(:,2));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:),x2G(:)]; % Defines a fine grid on the plot
idx2Region = kmeans(XGrid,3,'MaxIter',1,'Start',C);
figure;
gscatter(XGrid(:,1),XGrid(:,2),idx2Region,...
    [0,0.75,0.75;0.75,0,0.75;0.75,0.75,0],'..');
hold on;
plot(X(:,1),X(:,2),'k*','MarkerSize',5);
title 'Fisher''s Iris Data';
xlabel 'Petal Lengths (cm)';
ylabel 'Petal Widths (cm)';
legend('Region 1','Region 2','Region
3','Data','Location','SouthEast');
hold off;
```

Output:

```
Command Window
>> D

D =

   0.005960000000000  20.981423611111104   9.537352071005916
   0.005960000000000  20.981423611111104   9.537352071005916
   0.028360000000000  21.830590277777770  10.121198224852071
   0.003560000000000  20.152256944444435   8.973505917159763
   0.005960000000000  20.981423611111104   9.537352071005916
   0.080360000000000  17.858923611111102   7.488890532544377
   0.006760000000000  20.623923611111103   9.318890532544378
   0.003560000000000  20.152256944444435   8.973505917159763
   0.005960000000000  20.981423611111104   9.537352071005916
   0.022760000000000  20.529756944444436   9.211967455621300
   0.003560000000000  20.152256944444435   8.973505917159763
   0.021160000000000  19.343090277777769   8.429659763313609
   0.025160000000000  21.358923611111102   9.775813609467454
   0.152360000000000  23.966423611111097  11.587352071005915
   0.070760000000000  22.699756944444434  10.725044378698223
   0.025160000000000  19.457256944444435   8.556582840236684
   0.049960000000000  21.135590277777769   9.704275147928994
   0.006760000000000  20.623923611111103   9.318890532544378
   0.059560000000000  18.196423611111101   7.687352071005915
   0.004360000000000  19.794756944444437   8.755044378698223
   0.058760000000000  18.553923611111102   7.905813609467454
   0.025160000000000  19.457256944444435   8.556582840236684
   0.215560000000000  24.498090277777766  11.992736686390531
   0.121160000000000  17.541423611111099   7.310428994082838
   0.193960000000000  17.035590277777771   6.918121301775147
   0.021160000000000  19.343090277777769   8.429659763313609
   0.042760000000000  18.648090277777769   8.012736686390531
   0.003560000000000  20.152256944444435   8.973505917159763
   0.005960000000000  20.981423611111104   9.537352071005916
   0.021160000000000  19.343090277777769   8.429659763313609
   0.021160000000000  19.343090277777769   8.429659763313609
   0.025160000000000  19.457256944444435   8.556582840236684
   0.022760000000000  20.529756944444436   9.211967455621300
   0.005960000000000  20.981423611111104   9.537352071005916
   0.003560000000000  20.152256944444435   8.973505917159763
   0.070760000000000  22.699756944444434  10.725044378698223
fx
```

```
Command Window
   0.028360000000000  21.830590277777770  10.121198224852071
   0.003560000000000  20.152256944444435   8.973505917159763
   0.029160000000000  21.473090277777771   9.902736686390533
   0.029160000000000  21.473090277777771   9.902736686390533
   0.028360000000000  21.830590277777770  10.121198224852071
   0.144360000000000  18.033090277777767   7.675813609467454
   0.215560000000000  16.340590277777771   6.501198224852071
   0.006760000000000  20.623923611111103   9.318890532544378
   0.021160000000000  19.343090277777769   8.429659763313609
   0.005960000000000  20.981423611111104   9.537352071005916
   0.003560000000000  20.152256944444435   8.973505917159763
   0.005960000000000  20.981423611111104   9.537352071005916
  11.816360000000000   1.208923611111109   0.188890532544379
  10.801959999999999   1.489756944444442   0.078121301775148
  13.392360000000002   0.773090277777776   0.422736686390533
   7.552359999999998   3.090590277777774   0.074275147928994
  11.419559999999997   1.280590277777776   0.134275147928994
  10.340359999999999   1.744756944444442   0.055044378698225
  12.317959999999999   0.993923611111109   0.251967455621302
   3.946759999999999   6.347256944444441   1.056582840236686
  10.957959999999996   1.535590277777776   0.111198224852071
   7.275559999999999   3.282256944444441   0.139659763313610
   4.721959999999999   5.468923611111107   0.708890532544378
   9.069160000000000   2.237256944444441   0.029659763313610
   7.009959999999999   3.623090277777774   0.189659763313609
  11.816360000000000   1.208923611111109   0.188890532544379
   5.681960000000000   4.527256944444440   0.449659763313609
   9.963560000000001   1.836423611111107   0.020428994082840
  10.801959999999999   1.489756944444442   0.078121301775148
   7.527559999999997   3.313923611111108   0.145813609467456
  10.801959999999999   1.489756944444442   0.078121301775148
   6.673159999999999   3.754756944444440   0.195044378698225
  13.557159999999998   0.689756944444443   0.491198224852071
   7.552359999999998   3.090590277777774   0.074275147928994
  13.392360000000002   0.773090277777776   0.422736686390533
  11.394760000000000   1.503923611111108   0.205813609467456
   9.165159999999997   2.223090277777775   0.002736686390533
   9.963560000000001   1.836423611111107   0.020428994082840
  12.473959999999998   1.039756944444443   0.285044378698225
  14.631559999999999   0.468923611111110   0.661967455621302
fx
```

```
>> eva

eva =

  CalinskiHarabaszEvaluation with properties:

    NumObservations: 150
         InspectedK: [1 2 3 4]
    CriterionValues: [NaN 5.139245459802769e+02 5.616277566296201e+02 5.304871420421677e+02]
           OptimalK: 3
```

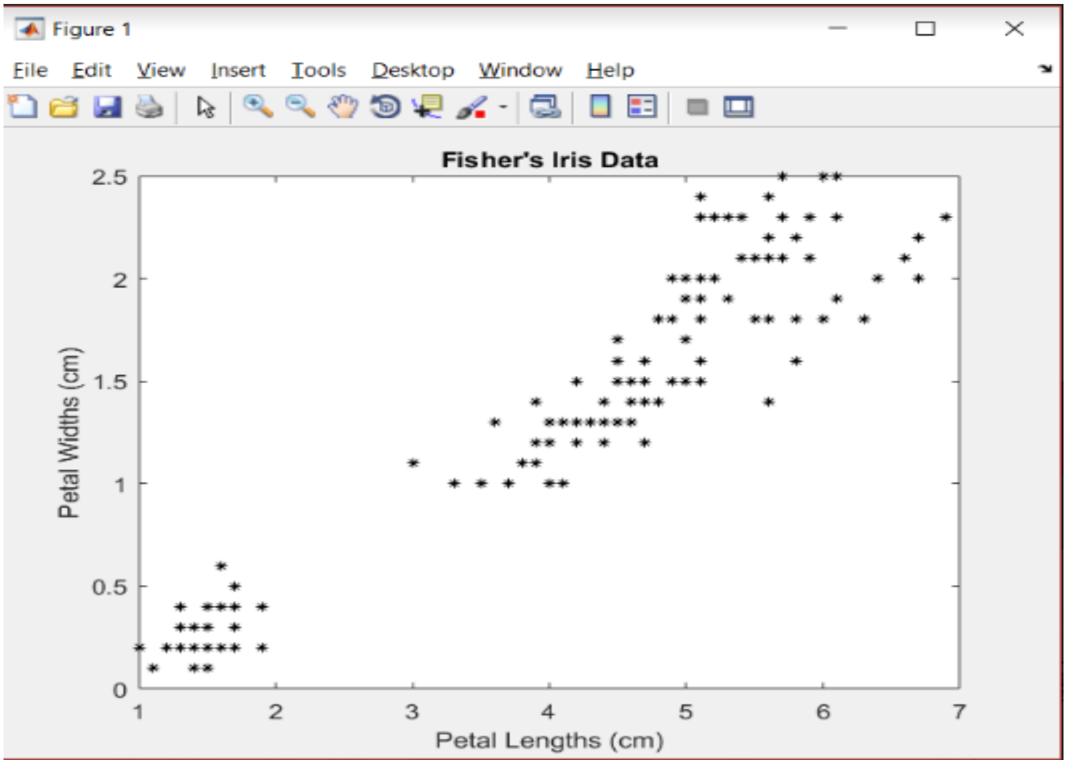
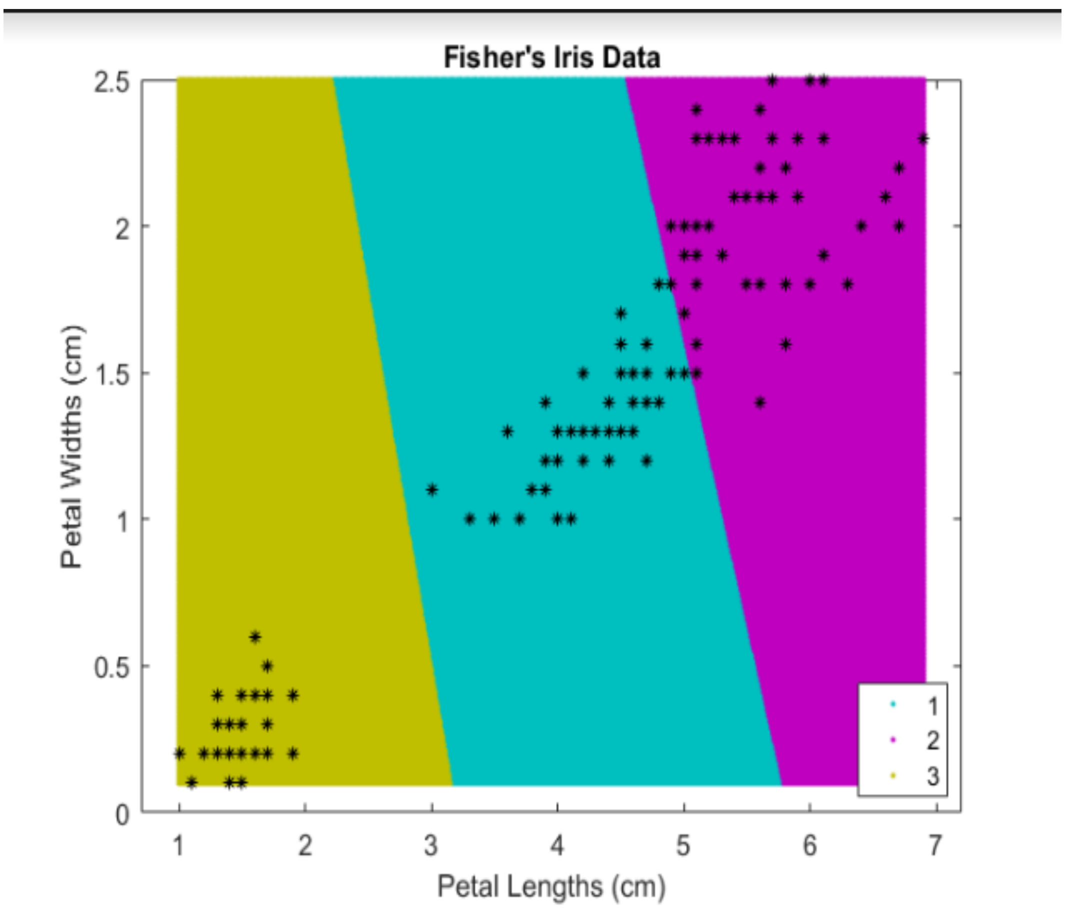


Figure 1
File Edit View Insert Tools Desktop Window Help
Fisher's Iris Data
Petal Widths (cm)
Petal Lengths (cm)


Fisher's Iris Data
Petal Widths (cm)
Petal Lengths (cm)
1
2
3

Code:

1) Kmeans Dataset

```
load kmeansdata
Y = X(:,3:4);
figure;
plot(Y(:,1),Y(:,2),'k*','MarkerSize',5);
title 'K-Means Data';
xlabel 'X';
ylabel 'Y';
rng(1); % For reproducibility
eva = evalclusters(X,'kmeans','CalinskiHarabasz','KList',[1:4]);
[idx,C,sumD,D] = kmeans(Y,4);
x1 = min(Y(:,1)):0.01:max(Y(:,1));
x2 = min(Y(:,2)):0.01:max(Y(:,2));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:),x2G(:)]; % Defines a fine grid on the plot
idx2Region = kmeans(XGrid,4,'MaxIter',1,'Start',C);
figure;
gscatter(XGrid(:,1),XGrid(:,2),idx2Region,...
    [0,0.75,0.75;0.75,0,0.75;0.75,0.75,0;0.75,0.75,0.75],'..');
hold on;
plot(Y(:,1),Y(:,2),'k*','MarkerSize',5);
title 'K-Means Data';
xlabel 'X';
ylabel 'Y';
legend('Region 1','Region 2','Region 3','Region
4','Data','Location','SouthEast');
hold off;
```

Output:

```
Command Window
>> D

D =

   32.4842   44.7349   78.1733    1.1332
   31.2985   28.3614   61.2225    0.1236
   39.1968   32.4856   73.1326    0.2594
   40.4226   25.1963   66.3610    0.7556
   37.7314   38.9524   78.0933    0.4815
   38.2181   23.7844   62.7371    0.7493
   46.8248   47.3781   93.8471    2.3433
   38.4742   29.9993   69.8929    0.2330
   29.9111   41.1405   72.3644    0.7139
   34.6566   49.1528   84.0819    1.8960
   36.9680   26.6371   64.8519    0.3316
   34.7824   37.7767   74.1790    0.2744
   42.5601   16.1333   56.8120    3.0824
   32.2696   40.6589   74.3550    0.5652
   22.7930   28.4175   51.8351    1.1362
   21.0189   24.6142   45.0931    2.2028
   29.3519   45.3611   75.4585    1.3871
   31.0319   56.6570   86.2696    3.7926
   38.9413   30.3953   70.7360    0.2581
   37.5992   36.4389   75.5835    0.2843
   27.6725   40.6839   69.5709    0.8220
   18.5477   27.9591   45.7980    2.4808
   30.7191   35.9912   68.4207    0.1811
   44.5659   24.8492   69.5258    1.3255
   16.1985   17.8267   26.0553    9.3919
   28.2569   13.1240   37.4437    5.1760
   24.1329   33.8676   59.1700    0.7267
   36.6691   44.9082   82.4541    1.1707
   34.6476   21.9410   57.1503    0.9087
   47.3815   44.8478   92.0849    2.0641
   22.8036   14.1878   31.4730    6.8163
   58.7501   62.6124  116.5293    7.0217
   30.6368   13.7260   41.4125    4.1955
   21.3446   30.1981   52.0043    1.3576
   37.3975   36.9200   75.8576    0.3041
fx 59.6276   31.6377   88.6908    3.9892
```

```
Command Window
   21.4592   23.4506   44.1953    2.3400
   57.9583   49.9502  105.2711    4.5412
   16.2973   37.7318   53.0167    3.0553
   43.1487   56.9035   98.4597    3.8446
   37.0126   22.1212   59.6297    0.9537
   22.1708   16.6074   35.0543    5.1187
   42.6669   28.7091   72.2329    0.7102
   33.1804   20.4025   53.7638    1.2590
   37.7446   20.9868   58.9115    1.2279
   26.9777   29.2369   57.5834    0.3815
   25.5716   37.7943   64.6042    0.7455
   32.7680   23.3925   57.0496    0.6433
   20.4495   19.7374   37.5688    4.1285
   75.8924   78.0502  142.0791   14.1649
   53.4354   38.3760   90.9710    2.5198
   24.3427   20.0924   43.4298    2.5099
   31.6321   23.1208   55.5683    0.7205
   28.2430   30.6996   60.5166    0.2042
   30.0904   26.1960   57.5682    0.3567
   33.1226   15.3224   46.6809    3.0433
   14.4213   38.7866   51.3282    3.9557
   38.8501   37.7166   77.9664    0.4478
   47.3546   47.5959   94.4941    2.4424
   60.6838   60.5378  116.4204    6.9167
   42.9668   48.6825   91.5631    2.1513
   26.3730   30.3083   58.0572    0.4085
   50.7703   34.8382   85.3421    1.8908
   43.8412   40.2938   84.8657    1.1215
   34.3331   24.8483   60.3100    0.4290
   43.3349   36.8064   81.1046    0.8076
   22.9844   28.6173   52.2904    1.0762
   32.2176   25.7599   59.2595    0.3232
   48.8535   42.1710   90.8834    1.9766
   57.2587   45.7928  100.9798    3.8842
   27.4579   26.3764   54.9411    0.5607
   36.9716   21.4369   58.7508    1.0878
   31.8372   27.8913   61.2571    0.1395
   29.8452   27.2753   58.5239    0.2673
   30.2781   17.2978   46.5067    2.4394
fx 22.2145   29.5502   52.3790    1.1816
```

```
>> eva

eva =

  CalinskiHarabaszEvaluation with properties:

    NumObservations: 560
         InspectedK: [1 2 3 4]
    CriterionValues: [NaN 477.6418 729.6816 1.4548e+03]
           OptimalK: 4
```

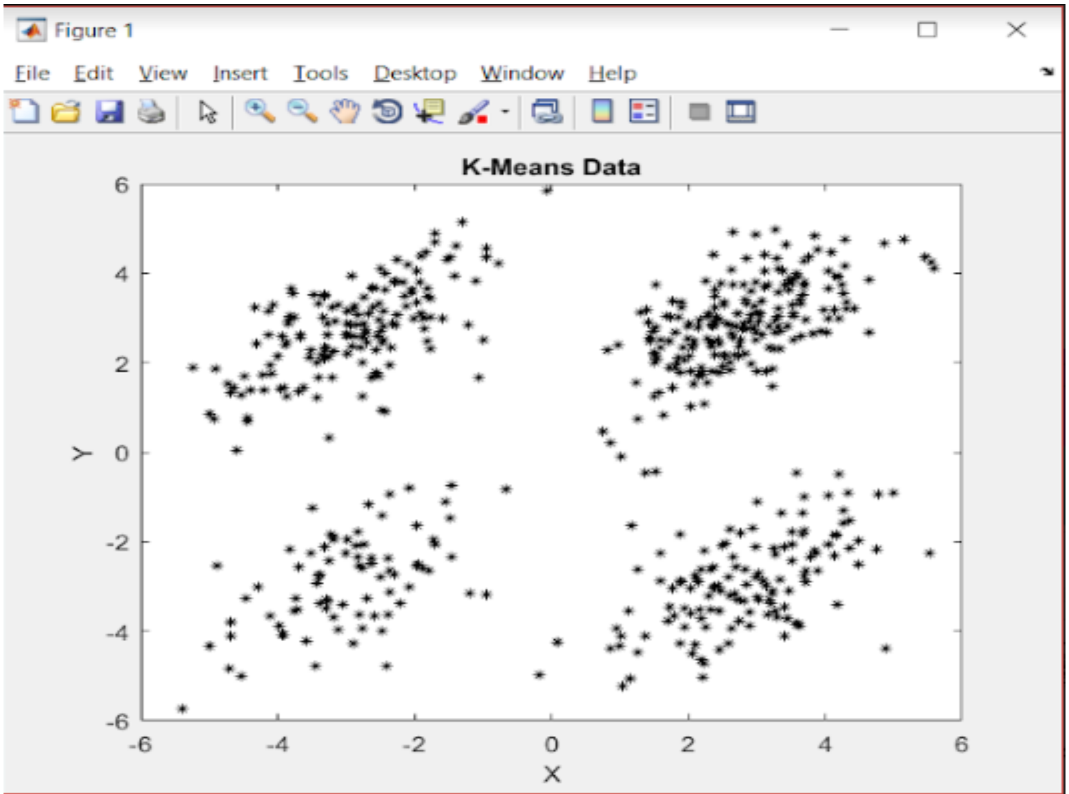


Figure 1
File Edit View Insert Tools Desktop Window Help
K-Means Data
Y
X

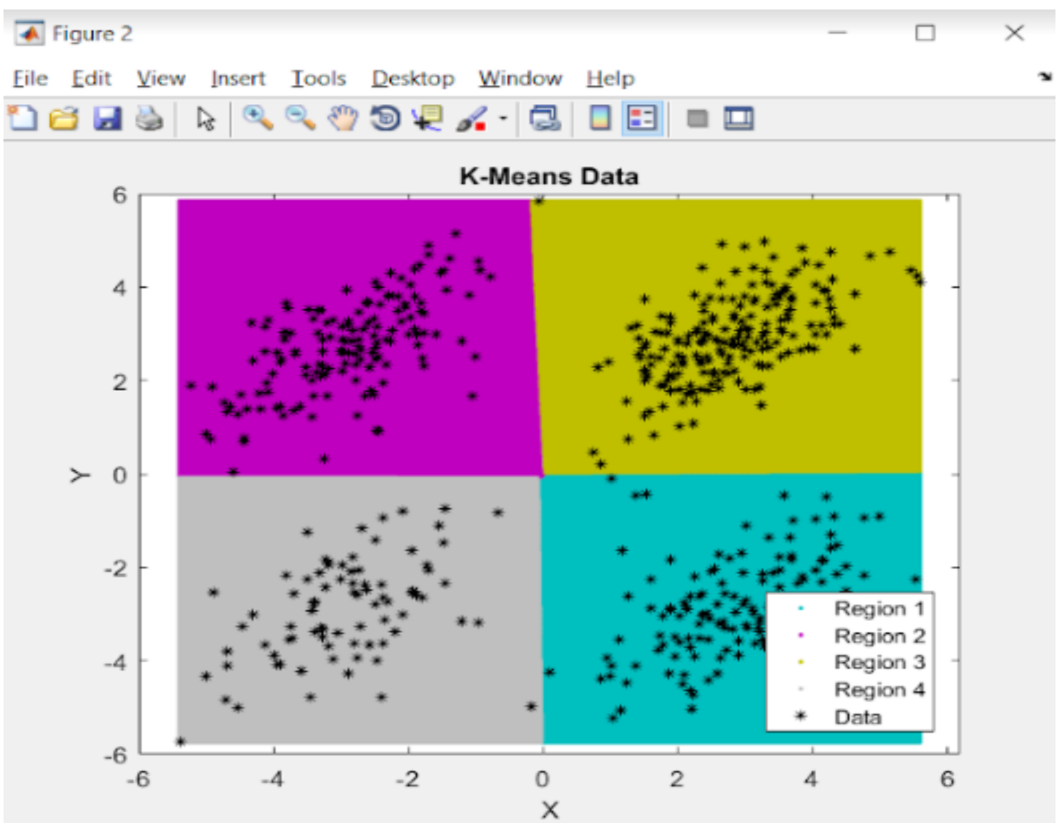


Figure 2
File Edit View Insert Tools Desktop Window Help
K-Means Data
Y
X
Region 1
Region 2
Region 3
Region 4
Data

Experiment-4

Aim: Implement linear regression on Iris Dataset.

Code:

```
load fisheriris
x = hwydata(:,1);
y = hwydata(:,2);
format long
b1 = x\y
yCalc1 = b1*x;
scatter(x,y)
hold on
plot(x,yCalc1)
xlabel('Petal Width')
ylabel('Petal Length')
title('Linear Regression Relation Between Petal Length & Petal
Width')
grid on
X = [ones(length(x),1) x];
b = X\y
yCalc2 = X*b;
plot(x,yCalc2,'--')
legend('Data','Slope','Slope&Intercept','Location','best');
```

Output:

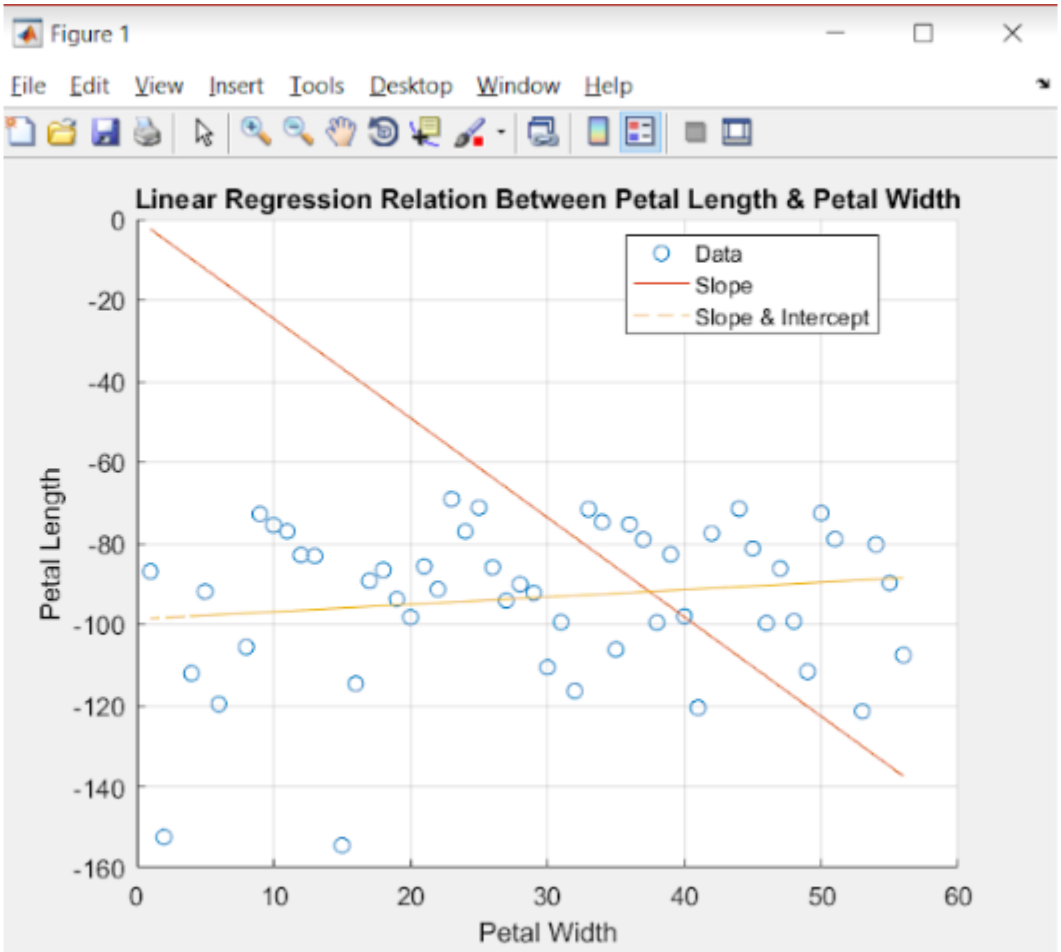
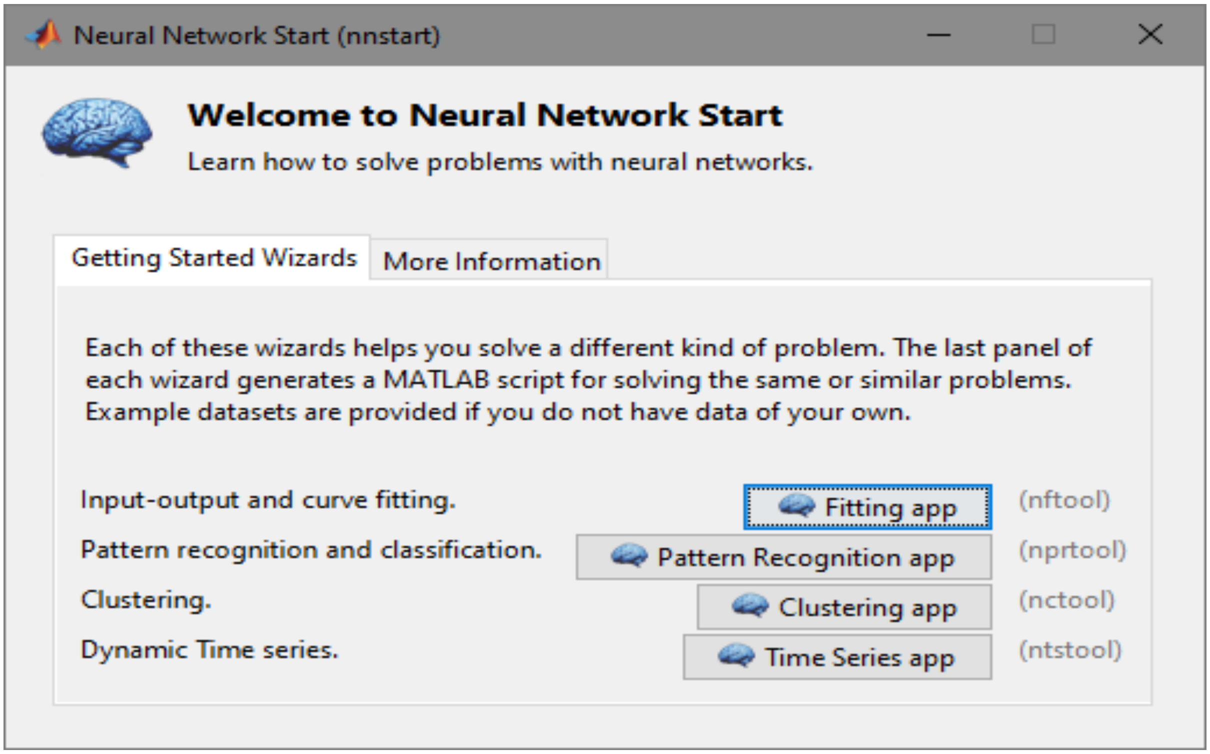
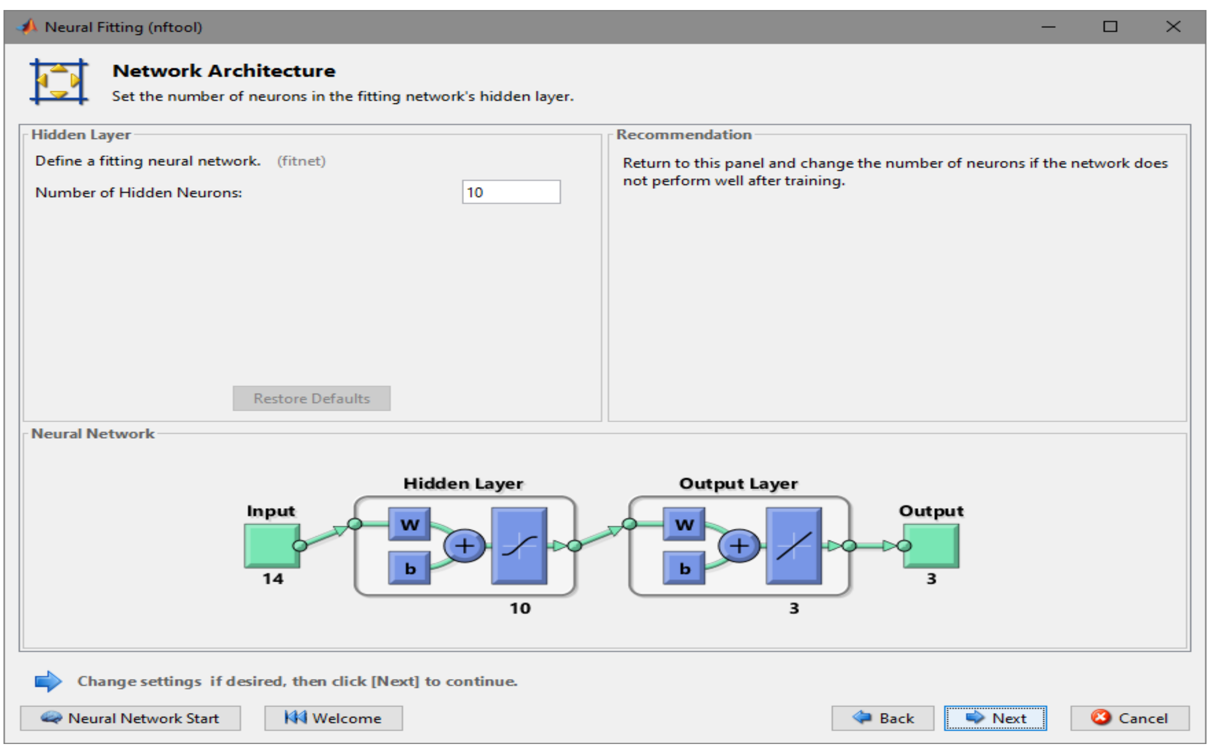


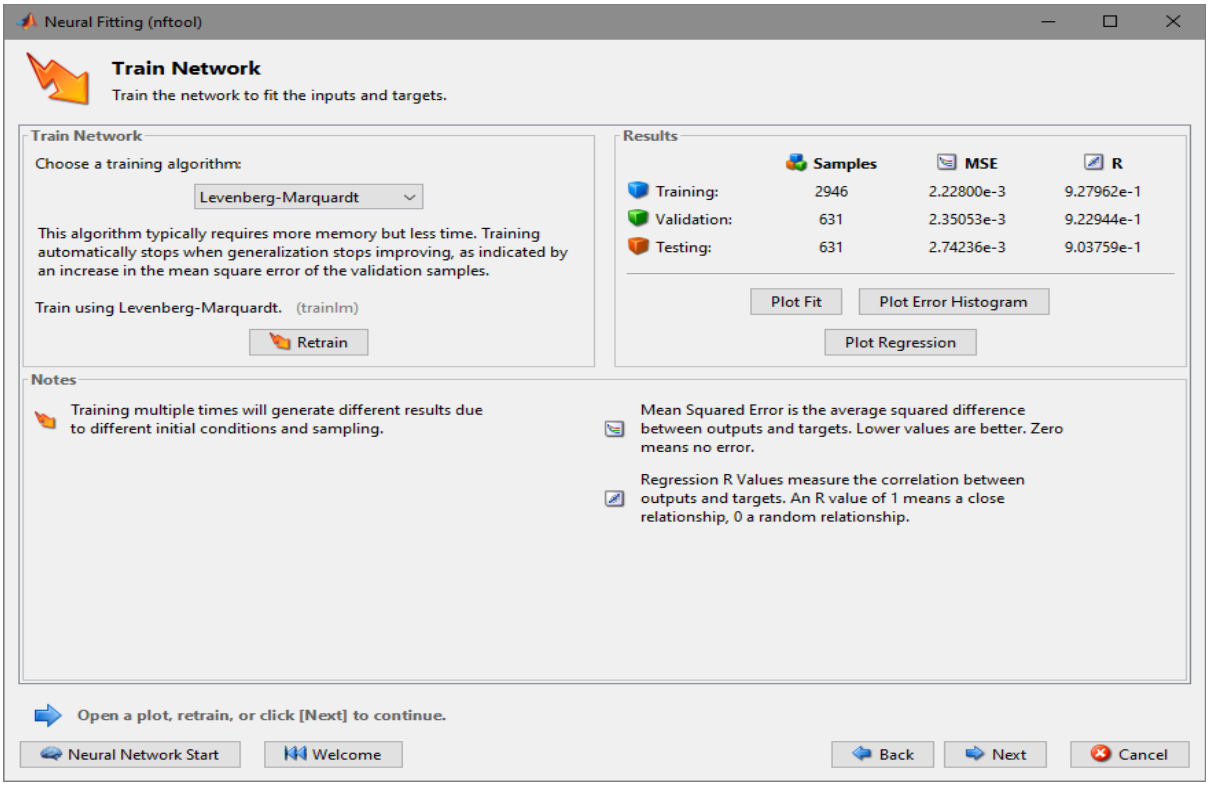
Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Linear Regression Relation Between Petal Length & Petal Width
Data
Slope
Slope & Intercept
Petal Length
Petal Width
0
-20
-40
-60
-80
-100
-120
-140
-160
0
10
20
30
40
50
60

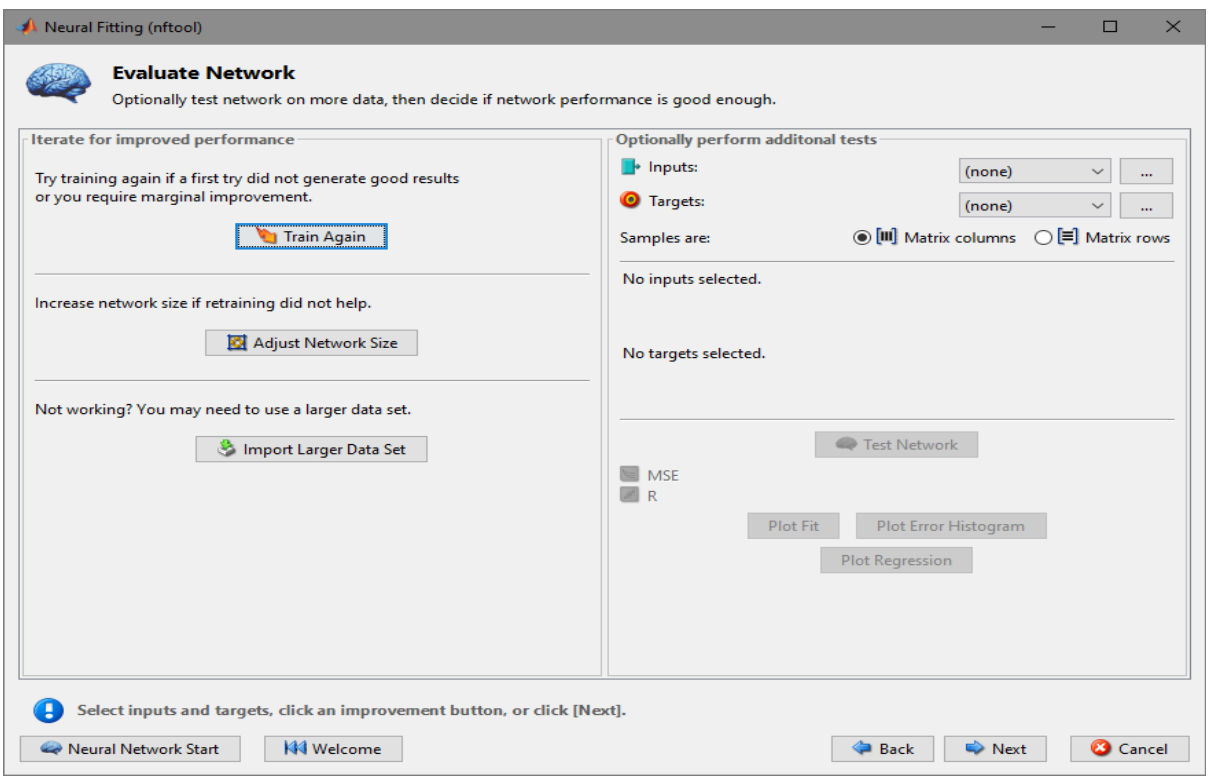
Experiment-5

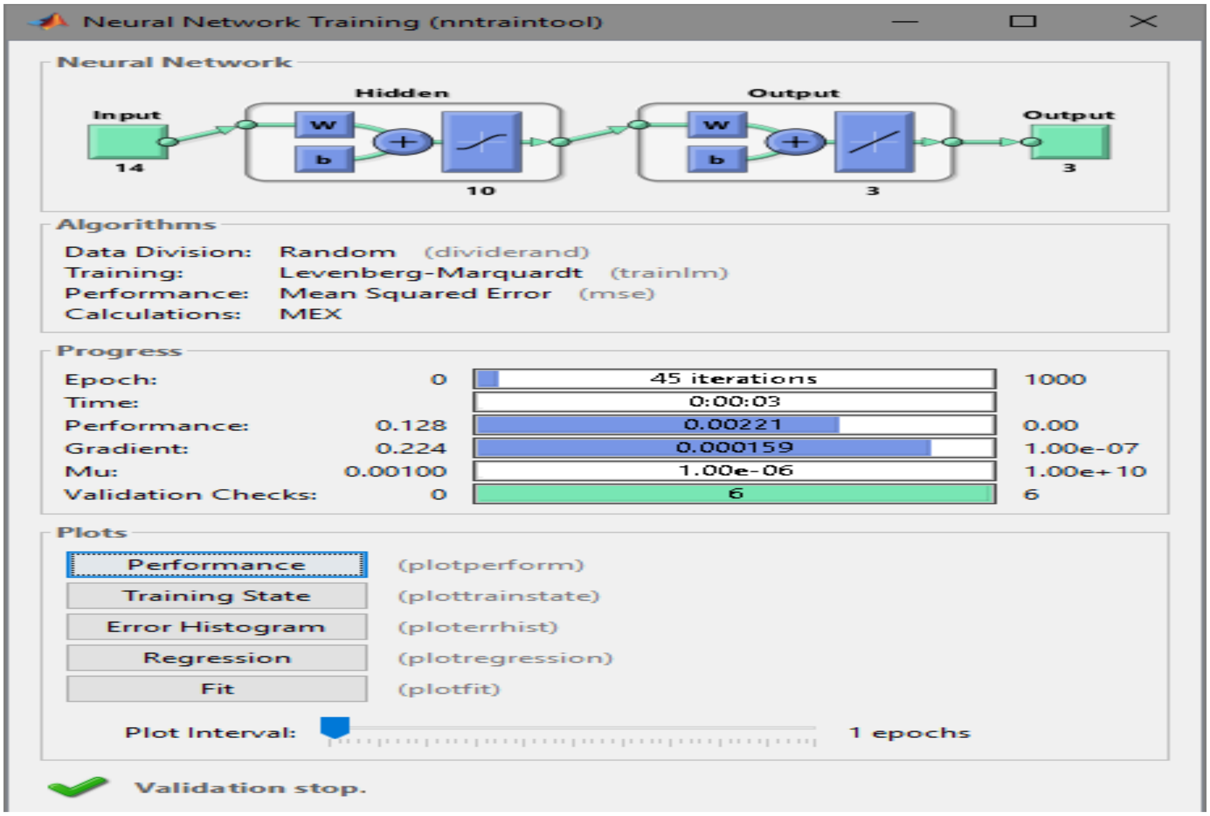
Aim: Classify Dataset using Neural Networks.



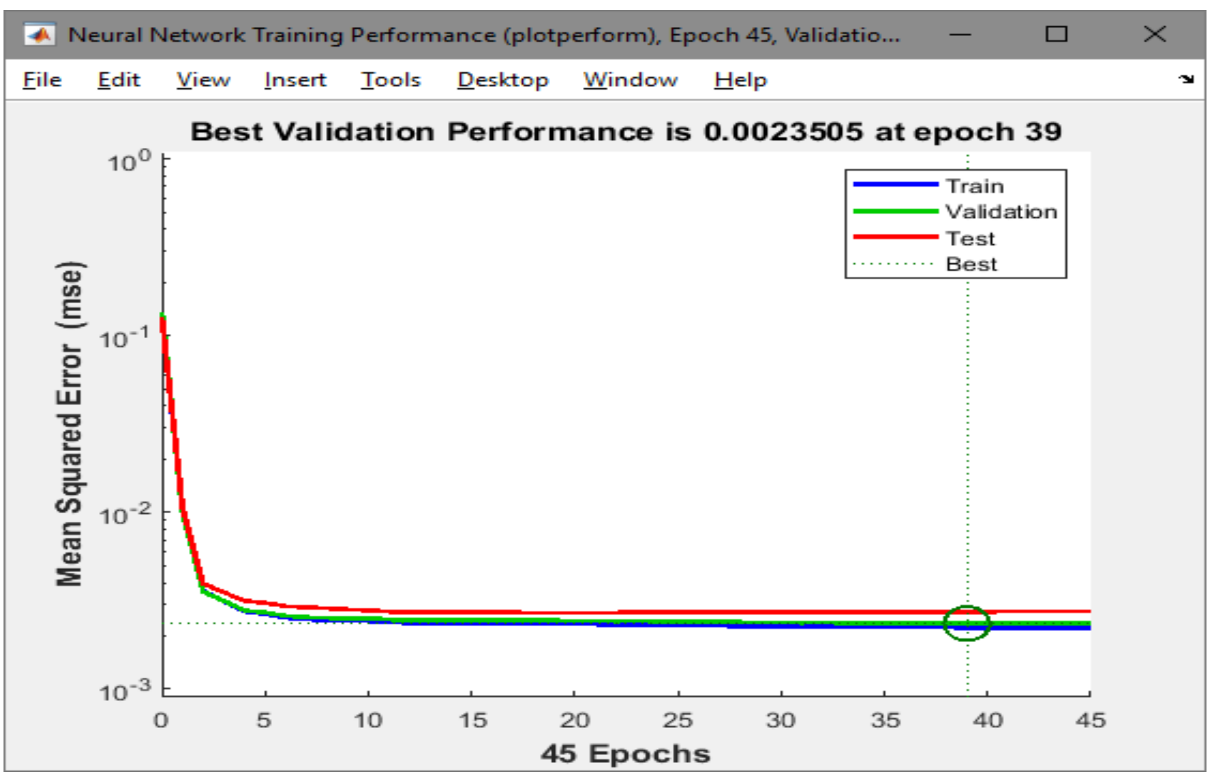



Neural Fitting (nftool)
Train Network
Train the network to fit the inputs and targets.
Train Network
Choose a training algorithm:
Levenberg-Marquardt
This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.
Train using Levenberg-Marquardt. (trainlm)
Retrain
Results
Samples
MSE
R
Training: 2946 2.22800e-3 9.27962e-1
Validation: 631 2.35053e-3 9.22944e-1
Testing: 631 2.74236e-3 9.03759e-1
Plot Fit
Plot Error Histogram
Plot Regression
Notes
Training multiple times will generate different results due to different initial conditions and sampling.
Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.
Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.
Open a plot, retrain, or click [Next] to continue.
Neural Network Start
Welcome
Back
Next
Cancel

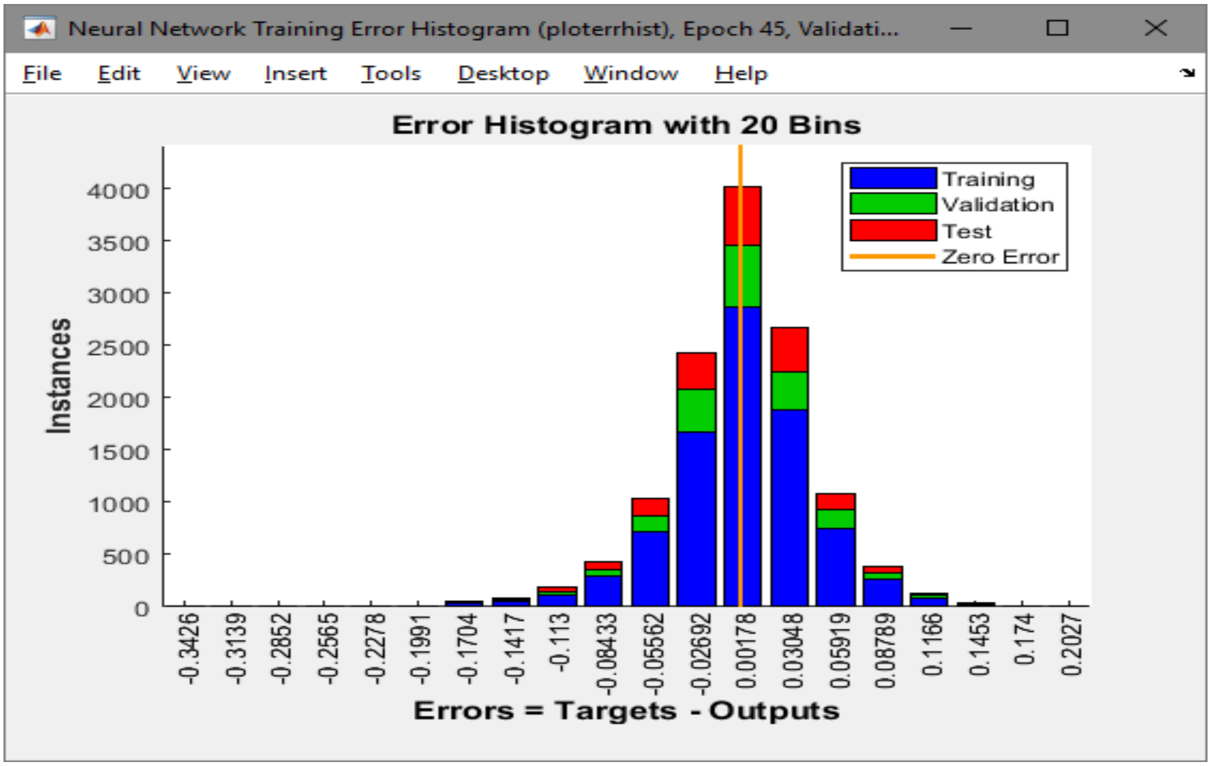

Neural Fitting (nftool)
Evaluate Network
Optionally test network on more data, then decide if network performance is good enough.
Iterate for improved performance
Try training again if a first try did not generate good results or you require marginal improvement.
Train Again
Increase network size if retraining did not help.
Adjust Network Size
Not working? You may need to use a larger data set.
Import Larger Data Set
Optionally perform additonal tests
Inputs:
(none)
Targets:
(none)
Samples are:
Matrix columns
Matrix rows
No inputs selected.
No targets selected.
Test Network
MSE
R
Plot Fit
Plot Error Histogram
Plot Regression
Select inputs and targets, click an improvement button, or click [Next].
Neural Network Start
Welcome
Back
Next
Cancel

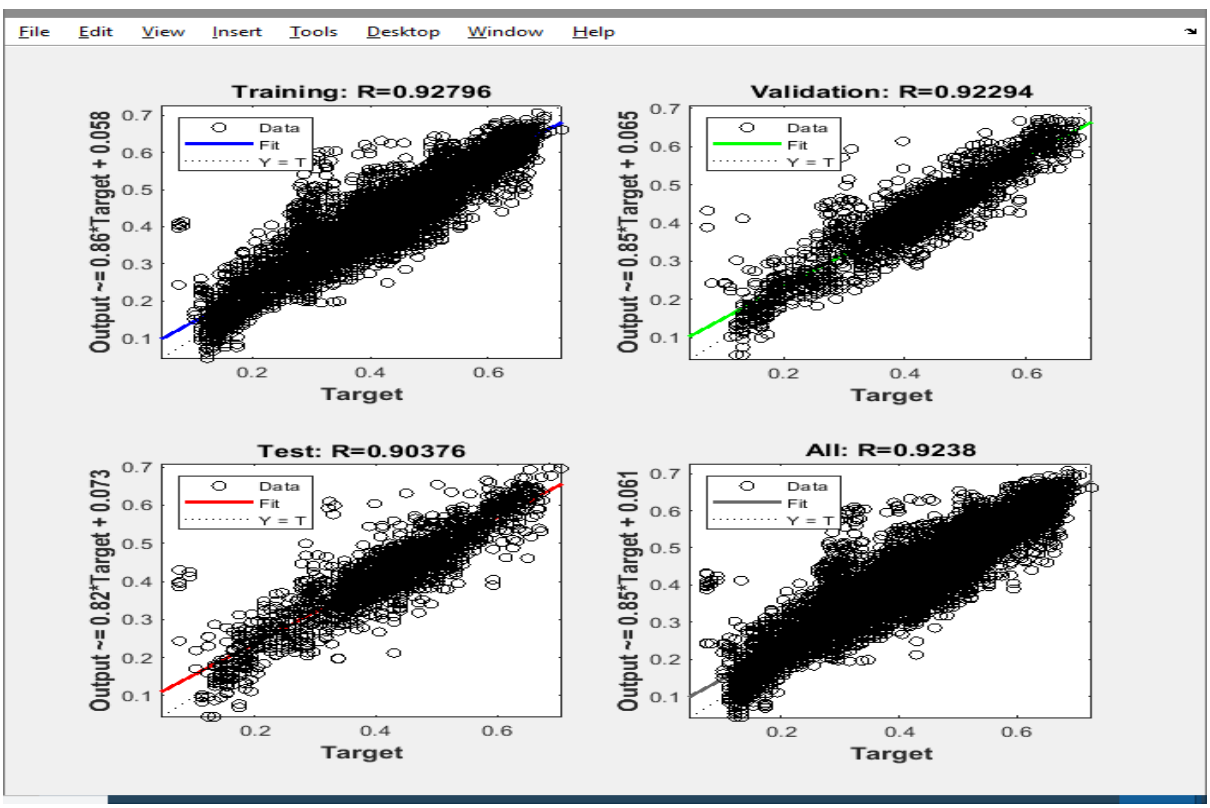


Neural Network Training (nntraintool)
Neural Network
Input
14
Hidden
W
b
10
Output
W
b
3
Output
3
Algorithms
Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX
Progress
Epoch: 0 45 iterations 1000
Time: 0:00:03
Performance: 0.128 0.00221 0.00
Gradient: 0.224 0.000159 1.00e-07
Mu: 0.00100 1.00e-06 1.00e+10
Validation Checks: 0 6 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)
Plot Interval: 1 epochs
Validation stop.



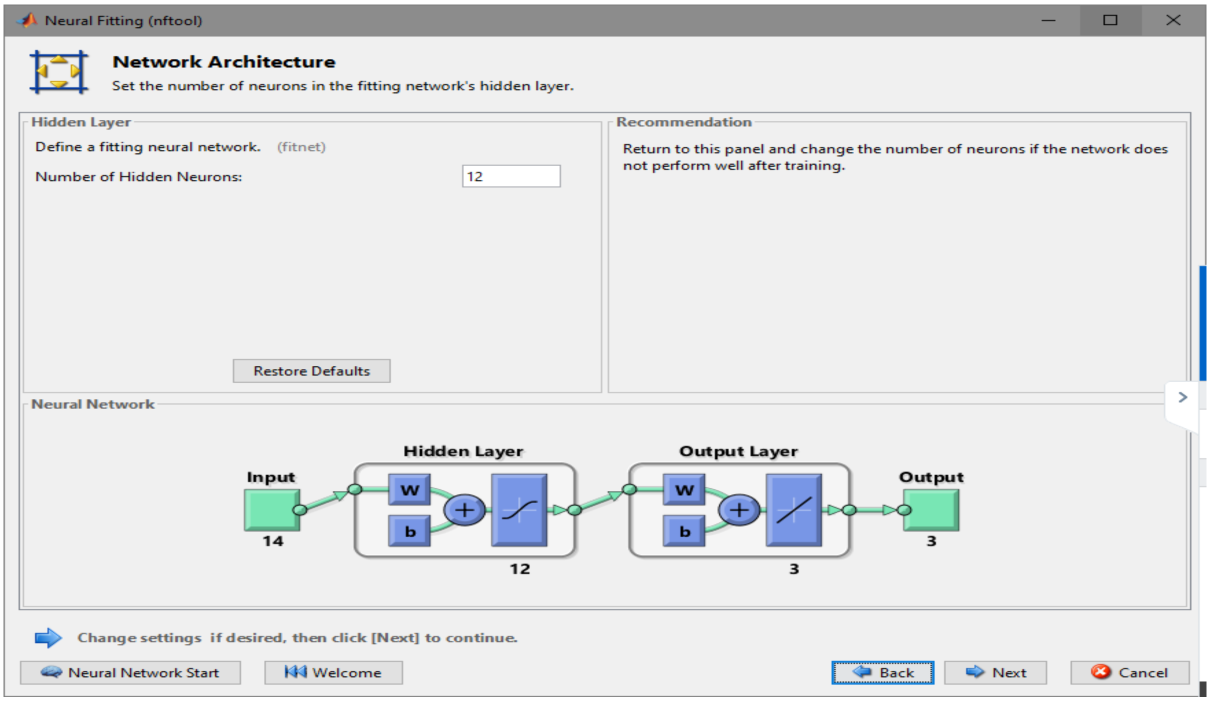
Neural Network Training Performance (plotperform), Epoch 45, Validatio...
File Edit View Insert Tools Desktop Window Help
Best Validation Performance is 0.0023505 at epoch 39
Mean Squared Error (mse)
10^0
10^{-1}
10^{-2}
10^{-3}
0 5 10 15 20 25 30 35 40 45
45 Epochs
Train
Validation
Test
Best

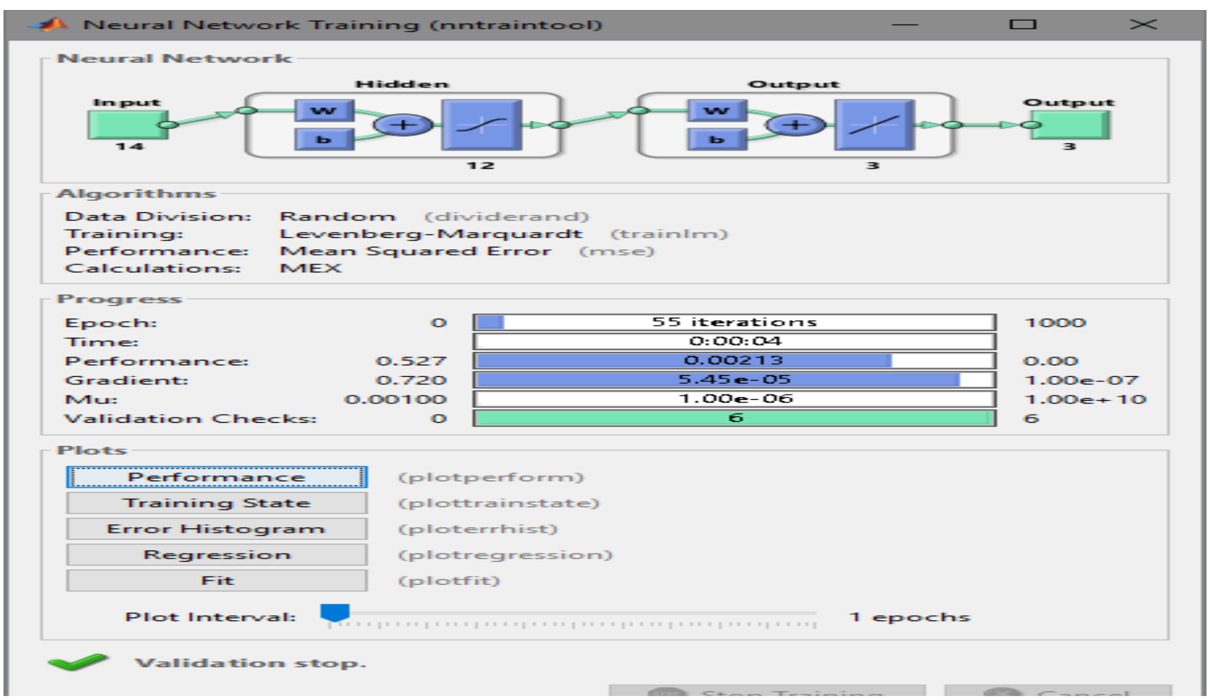

Neural Network Training Error Histogram (ploterrhist), Epoch 45, Validati...
File Edit View Insert Tools Desktop Window Help
Error Histogram with 20 Bins
Instances
Training
Validation
Test
Zero Error
Errors = Targets - Outputs

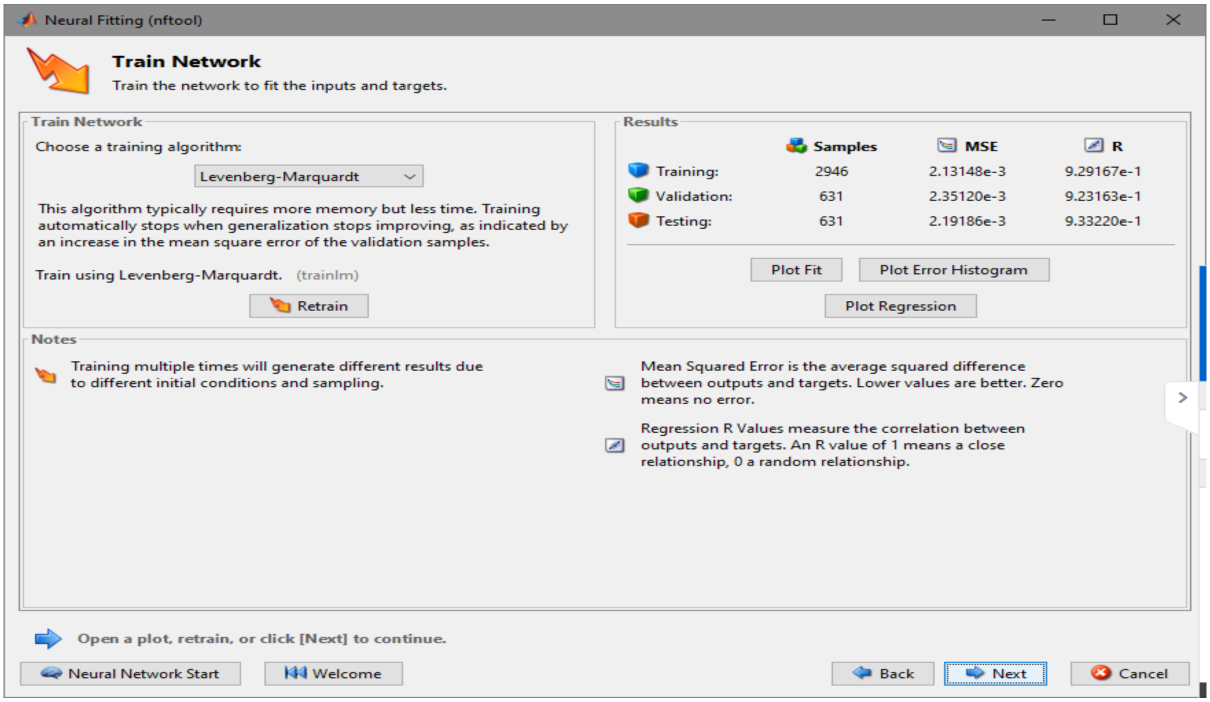

File Edit View Insert Tools Desktop Window Help
Training: R=0.92796
Output ~= 0.86*Target + 0.058
Validation: R=0.92294
Output ~= 0.85*Target + 0.065
Test: R=0.90376
Output ~= 0.82*Target + 0.073
All: R=0.9238
Output ~= 0.85*Target + 0.061
Data
Fit
Y = T
Target

Experiment-6

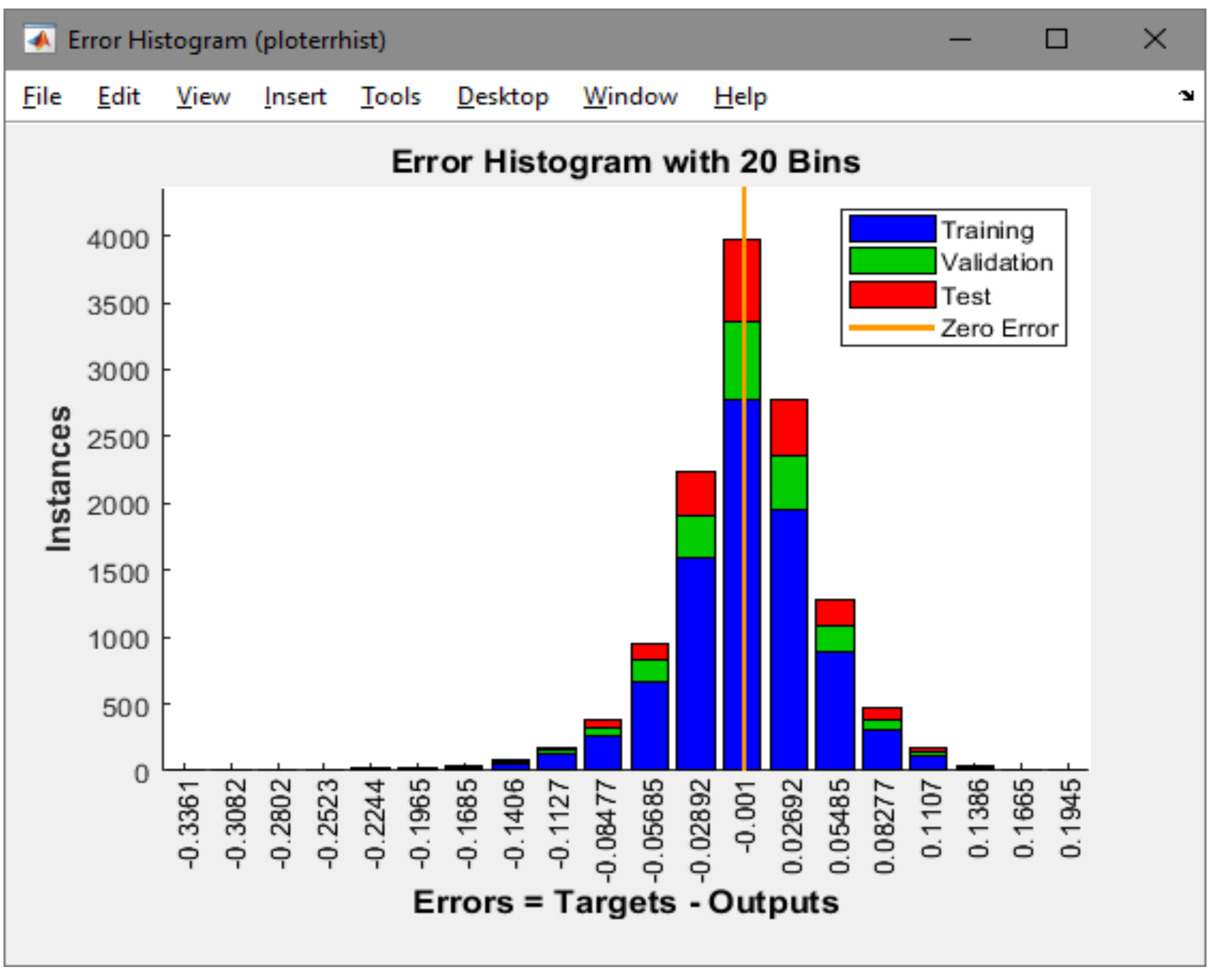
Aim: Classify Dataset by increasing number of layers and number of neurons in each layer using Neural Networks .Report classification accuracy.



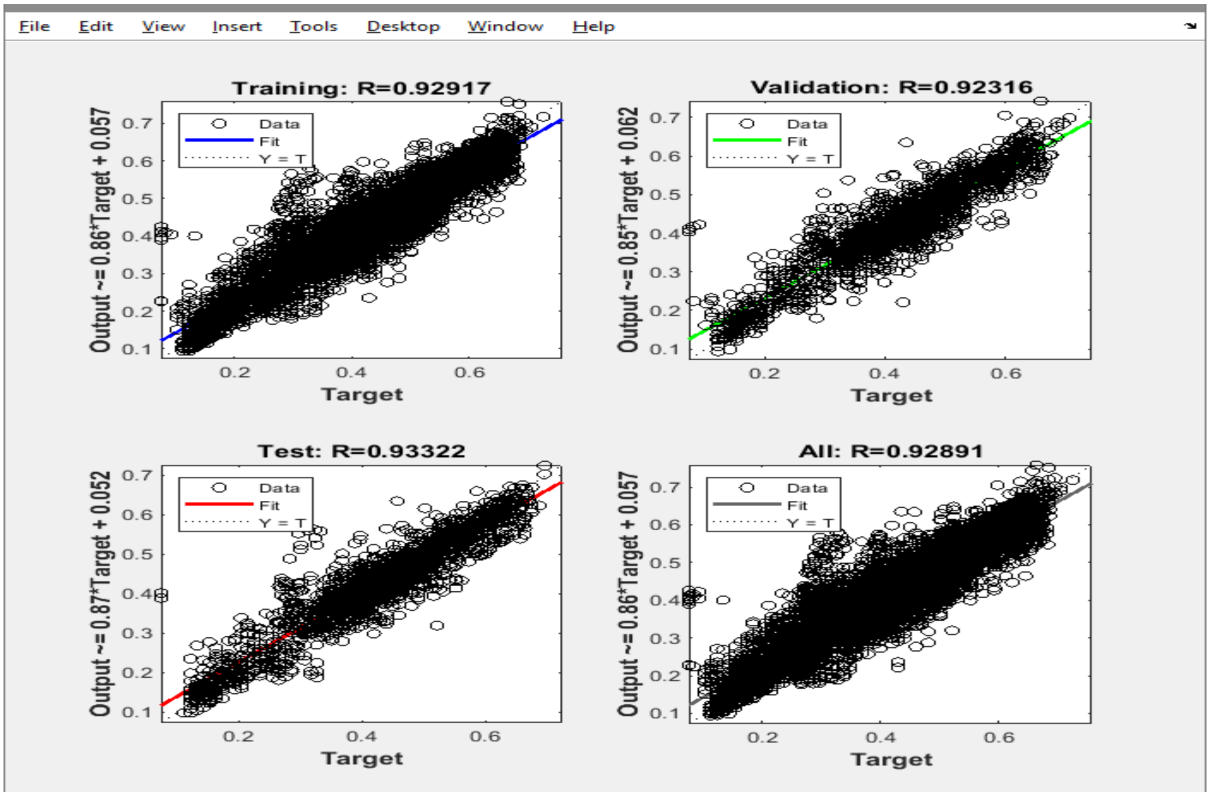


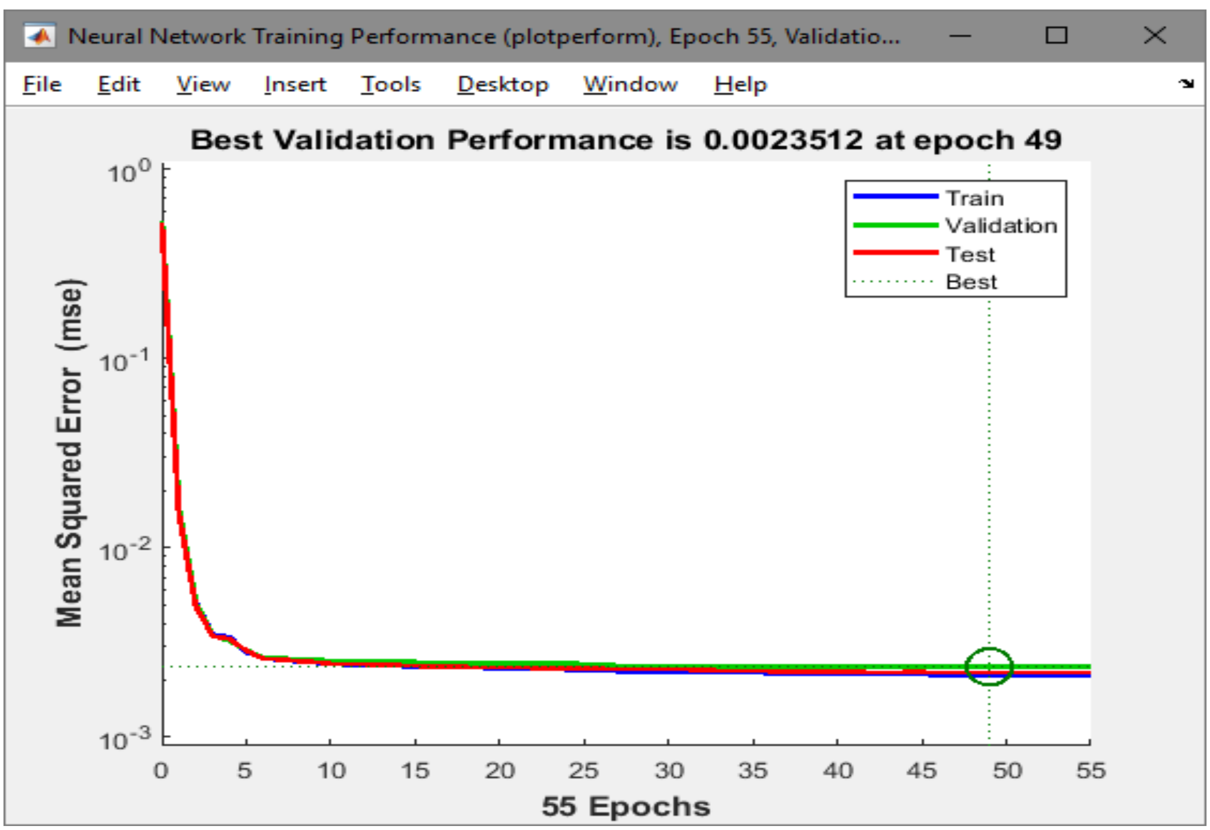


Neural Fitting (nftool)
Train Network
Train the network to fit the inputs and targets.
Train Network
Choose a training algorithm:
Levenberg-Marquardt
This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.
Train using Levenberg-Marquardt. (trainlm)
Retrain
Results
Samples MSE R
Training: 2946 2.13148e-3 9.29167e-1
Validation: 631 2.35120e-3 9.23163e-1
Testing: 631 2.19186e-3 9.33220e-1
Plot Fit
Plot Error Histogram
Plot Regression
Notes
Training multiple times will generate different results due to different initial conditions and sampling.
Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.
Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.
Open a plot, retrain, or click [Next] to continue.
Neural Network Start
Welcome
Back
Next
Cancel



Error Histogram (ploterrhist)
File Edit View Insert Tools Desktop Window Help
Error Histogram with 20 Bins
Instances
0
500
1000
1500
2000
2500
3000
3500
4000
Training
Validation
Test
Zero Error
-0.3361
-0.3082
-0.2802
-0.2523
-0.2244
-0.1965
-0.1685
-0.1406
-0.1127
-0.08477
-0.05685
-0.02892
-0.001
0.02692
0.05485
0.08277
0.1107
0.1386
0.1665
0.1945
Errors = Targets - Outputs


File Edit View Insert Tools Desktop Window Help
Training: R=0.92917
Output ~= 0.86*Target + 0.057
Validation: R=0.92316
Output ~= 0.85*Target + 0.062
Test: R=0.93322
Output ~= 0.87*Target + 0.052
All: R=0.92891
Output ~= 0.86*Target + 0.057
Data
Fit
Y = T
Target


Neural Network Training Performance (plotperform), Epoch 55, Validatio...
File Edit View Insert Tools Desktop Window Help
Best Validation Performance is 0.0023512 at epoch 49
Train
Validation
Test
Best
Mean Squared Error (mse)
55 Epochs

Experiment-7

Aim: Use Covertype Dataset and implement PCA. Reduce the number of attributes to 10.Classify this data, using any two machine learning algorithms.

Code:

```
covertype = xlsread('covertype.xlsx');
[coeff,score,latent] = pca(covertype);
figure, biplot(coeff(:,1:2),'scores',score(:,1:2),'varlabels',
{'v_1','v_2','v_3','v_4','v_5','v_6','v_7','v_8','v_9','v_1_0'})
;
figure, biplot(coeff(:,1:3),'scores',score(:,1:3),'varlabels',
{'v_1','v_2','v_3','v_4','v_5','v_6','v_7','v_8','v_9','v_1_0'})
;
```

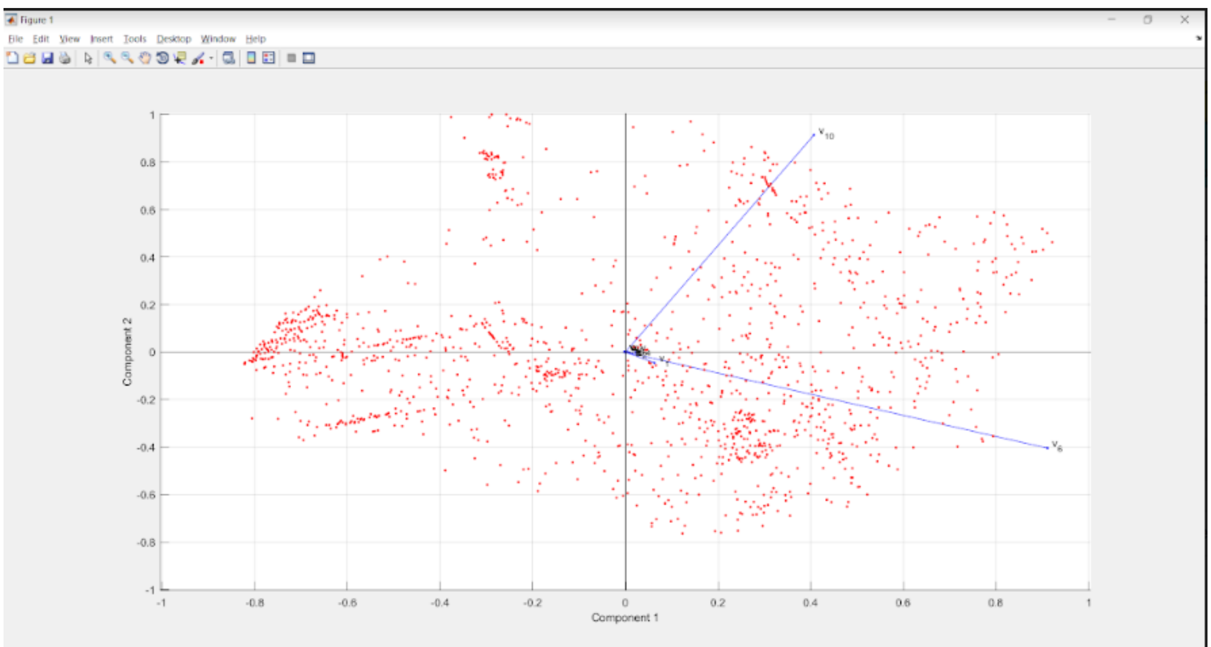
Output:

Command Window

```
coeff =

    0.0624   -0.0454    0.2777    0.6895    0.6638   -0.0148    0.0262    0.0074    0.0057   -0.0002
    0.0144   -0.0074   -0.0542   -0.6613    0.7170    0.0661   -0.2002    0.0270   -0.0160    0.0013
   -0.0009   -0.0003    0.0008   -0.0000   -0.0007    0.1170   -0.0584   -0.1932    0.8905    0.3906
    0.0219    0.0001    0.9501   -0.2361   -0.1538   -0.1103   -0.0670   -0.0264    0.0036   -0.0005
   -0.0001   -0.0009    0.1248   -0.0463   -0.0036    0.8675    0.4251    0.2176   -0.0397    0.0016
    0.9109   -0.4047   -0.0391   -0.0353   -0.0599    0.0061   -0.0050   -0.0019   -0.0008    0.0000
   -0.0024    0.0010    0.0021    0.0797   -0.0724    0.0303   -0.4085    0.6600   -0.1515    0.6017
    0.0040   -0.0004   -0.0053   -0.0446    0.0229   -0.2548    0.2037    0.6905    0.4132   -0.4935
    0.0076   -0.0014   -0.0077   -0.1402    0.1104   -0.3887    0.7498    0.0381   -0.1071    0.4918
    0.4069    0.9133   -0.0040    0.0129    0.0125    0.0027   -0.0005   -0.0004    0.0002    0.0000
```

Command Window

```
>> score

score =

   1.0e+03 *

  -1.0877    4.2291    0.0316   -0.0147   -0.0535   -0.0490    0.0158    0.0009   -0.0017   -0.0008
  -1.2203    4.2285   -0.0099   -0.0081   -0.0399   -0.0516    0.0192    0.0028   -0.0015   -0.0018
   1.2946    2.9942   -0.0016   -0.0285   -0.0183    0.0296    0.0031    0.0248    0.0006    0.0002
   1.2475    3.1135   -0.0226   -0.0420   -0.0109    0.0857    0.0121    0.0379    0.0069   -0.0002
  -1.2421    4.1796   -0.0632    0.0158   -0.0362   -0.0411    0.0266    0.0045   -0.0020   -0.0018
  -1.5912    4.1820    0.0789   -0.0752    0.0089   -0.0619   -0.0165    0.0080    0.0022   -0.0006
  -0.9843    4.1579    0.0421   -0.0098   -0.0621   -0.0396    0.0084   -0.0040   -0.0002   -0.0001
  -1.0511    4.1566    0.0100   -0.0040   -0.0503   -0.0380    0.0169    0.0019   -0.0017   -0.0008
  -0.9592    4.1330    0.0218    0.0022   -0.0532    0.0112    0.0272    0.0052   -0.0018    0.0009
  -0.9923    4.1326    0.0220   -0.0075   -0.0472   -0.0236   -0.0045   -0.0029    0.0000    0.0008
  -0.9043    4.0841   -0.0486   -0.0981    0.0641    0.0070    0.0252    0.0237   -0.0020   -0.0009
   2.3483    0.2610    0.0406   -0.1026   -0.1209   -0.0094   -0.0290    0.0126    0.0032    0.0003
   1.3073    2.9555   -0.1310   -0.0341   -0.0535    0.0691   -0.0297    0.0236    0.0080   -0.0005
  -0.8766    4.0595   -0.0807   -0.1047    0.0753    0.0027    0.0316    0.0229    0.0019   -0.0003
  -1.2230    3.5460   -0.1848   -0.1191   -0.0279   -0.0198    0.0088    0.0164   -0.0017   -0.0009
  -1.1644    3.4937   -0.2080   -0.0490   -0.1123   -0.0157    0.0165    0.0021   -0.0021    0.0007
  -1.0728    4.0205   -0.1103   -0.1183    0.1200   -0.0282   -0.0037    0.0123   -0.0051   -0.0013
  -1.2923    3.5844   -0.1556   -0.0513   -0.0801   -0.0156    0.0082    0.0067   -0.0015    0.0002
  -1.2204    3.5146   -0.1498   -0.0234   -0.1444   -0.0326    0.0449   -0.0003   -0.0014   -0.0007
  -1.1817    3.4787   -0.1628   -0.0467   -0.1214   -0.0184    0.0275    0.0025   -0.0027   -0.0005
  -1.1616    3.4607   -0.1899   -0.0614   -0.0997   -0.0060    0.0027    0.0052   -0.0014    0.0007
   2.2133    0.6172   -0.1021   -0.1045   -0.0330   -0.0060    0.0185    0.0104    0.0121    0.0007
   1.3738    2.7957   -0.0853   -0.0155   -0.0689    0.0867   -0.0433    0.0161    0.0043   -0.0006
  -1.2684    3.5327   -0.1215   -0.0541   -0.1063   -0.0236    0.0042   -0.0020   -0.0014   -0.0003
  -1.1977    3.4631   -0.1266   -0.0417   -0.1352   -0.0175    0.0299   -0.0069   -0.0009    0.0009
  -1.0992    3.3746   -0.2575   -0.1023   -0.0546   -0.0117    0.0053    0.0168   -0.0009   -0.0000
  -1.1116    3.3717   -0.2315   -0.1301   -0.0401   -0.0189   -0.0012    0.0200    0.0045    0.0007
   2.6889   -0.6098   -0.0073   -0.0657   -0.1154    0.0023   -0.0430    0.0124    0.0065   -0.0000
   1.5458    2.3590   -0.0745   -0.0335   -0.0400   -0.0019    0.0111    0.0109   -0.0057   -0.0021
   1.3426    2.8762   -0.1568   -0.0179   -0.0718    0.0690   -0.0527    0.0126    0.0062   -0.0001
   1.2936    2.9964   -0.2240   -0.0232   -0.0786    0.0640   -0.0781    0.0011    0.0103   -0.0021
  -1.3062    3.5492   -0.1232   -0.0776   -0.0596   -0.0215   -0.0062    0.0116   -0.0005   -0.0003
  -1.2912    3.5317   -0.1206   -0.0809   -0.0647   -0.0228    0.0022    0.0118   -0.0010    0.0000
fx  -1.2609    3.4990   -0.0941   -0.0408   -0.1281   -0.0334    0.0280   -0.0041   -0.0011   -0.0007
```



KNN:

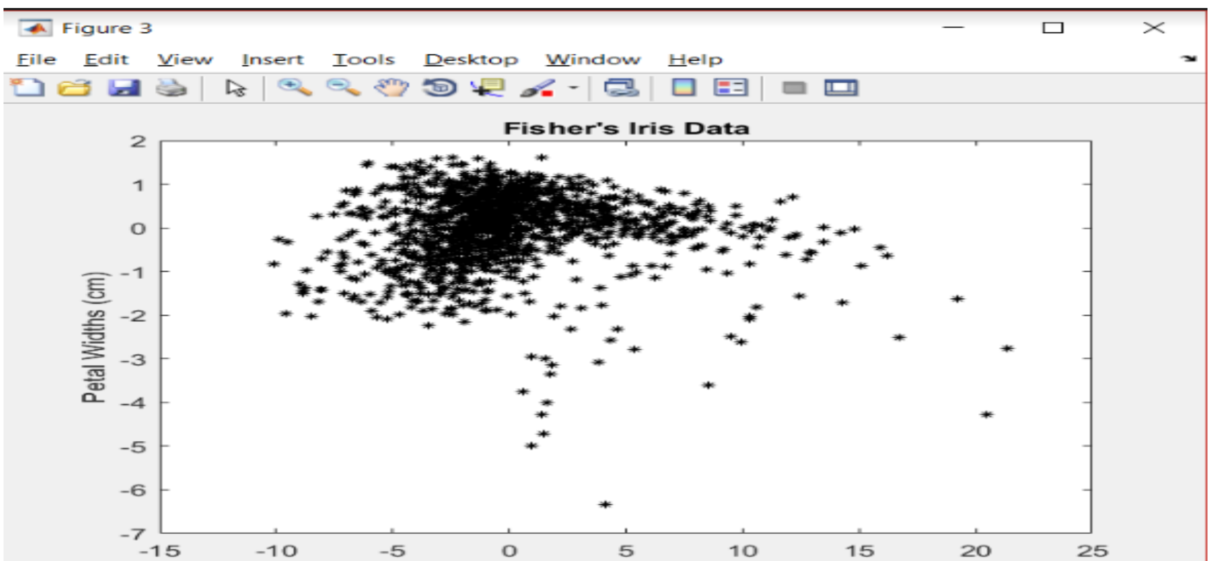
```
score1 = xlsread('score1.xlsx');
Y = score1(:,11);
%Mdl = fitcknn(score1,Y);
Mdl = fitcknn(score1,Y,'NumNeighbors',5,'Standardize',1);
```

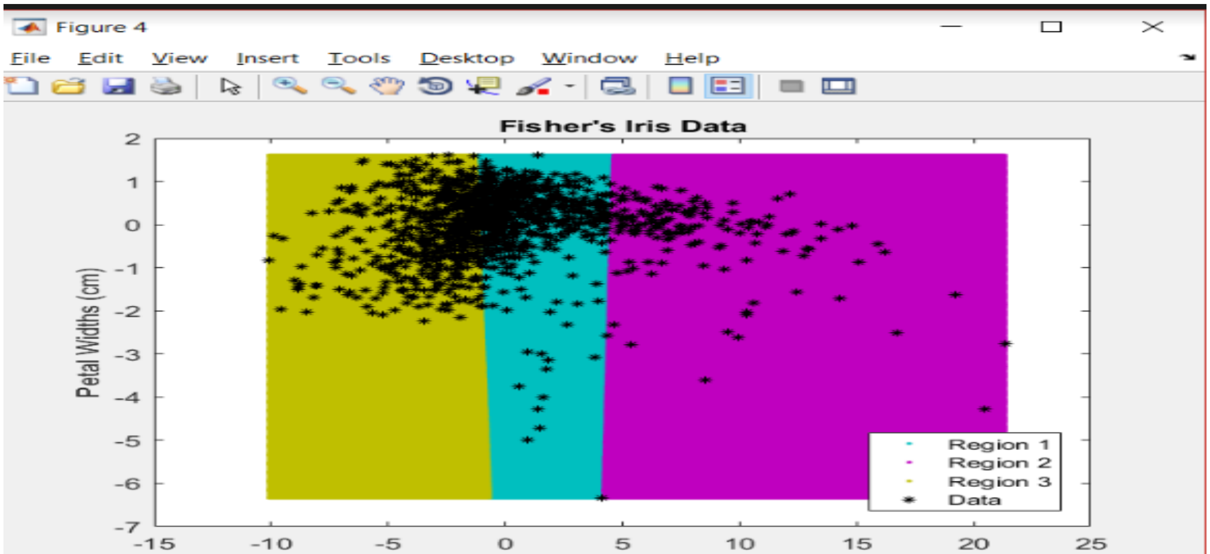
```
>> Mdl

Mdl =

  ClassificationKNN
             ResponseName: 'Y'
    CategoricalPredictors: []
               ClassNames: [1 2 5]
           ScoreTransform: 'none'
          NumObservations: 1370
                 Distance: 'euclidean'
             NumNeighbors: 5


  Properties, Methods

>> Mdl.ModelParameters

ans =


       NumNeighbors: 5
           NSMethod: 'exhaustive'
           Distance: 'euclidean'
         BucketSize: []
        IncludeTies: 0
     DistanceWeight: 'equal'
          BreakTies: 'smallest'
           Exponent: []
                Cov: []
              Scale: []
    StandardizeData: 1
            Version: 1
             Method: 'KNN'
               Type: 'classification'
```

Kmeans

```
X = score(:,9:10);
figure;
plot(X(:,1),X(:,2),'k*','MarkerSize',5);
title 'CoverType Data';
rng(1); % For reproducibility
[idx,C,sumD,D] = kmeans(X,3);
x1 = min(X(:,1)):0.01:max(X(:,1));
x2 = min(X(:,2)):0.01:max(X(:,2));
```

```
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:),x2G(:)]; % Defines a fine grid on the plot
idx2Region = kmeans(XGrid,3,'MaxIter',1,'Start',C);
figure;
gscatter(XGrid(:,1),XGrid(:,2),idx2Region,...
    [0,0.75,0.75;0.75,0,0.75;0.75,0.75,0],'..');
hold on;
plot(X(:,1),X(:,2),'k*','MarkerSize',5);
title 'CoverType Data';
legend('Region 1','Region 2','Region
3','Data','Location','SouthEast');
hold off;
```





Experiment-8

Aim: Classify dataset using SVM's.

Code:

```
% SVM Linear classification
% A 2-feature example

clear all; close all;

% Load training features and labels
[y, x] = libsvmread('twofeature.txt');

% Set the cost
C = 100;

% Train the model and get the primal variables w, b from the
model
% Libsvm options
% -s 0 : classification
% -t 0 : linear kernel
% -c somenumber : set the cost
model = svmtrain(y, x, sprintf('-s 0 -t 0 -c %g', C));
w = model.SVs' * model.sv_coef;
b = -model.rho;
if (model.Label(1) == -1)
    w = -w; b = -b;
```

```
end

% Plot the data points
figure
pos = find(y == 1);
neg = find(y == -1);
plot(x(pos,1), x(pos,2), 'ko', 'MarkerFaceColor', 'b'); hold on;
plot(x(neg,1), x(neg,2), 'ko', 'MarkerFaceColor', 'g')

% Plot the decision boundary
plot_x = linspace(min(x(:,1)), max(x(:,1)), 30);
plot_y = (-1/w(2))*(w(1)*plot_x + b);
plot(plot_x, plot_y, 'k-', 'LineWidth', 2)

title(sprintf('SVM Linear Classifier with C = %g', C),
'FontSize', 14)
```

Output: