Assignment 2

MAT8190 Mathematics Statistics Complementary Studies

Student: John Worrall
Student ID: Q12222096
Due Date 22nd October 2018
Weighting: 60%

Contents

Submission Requirements – 10 MARKS

Provide results.

See provide Excel file provided, labeled ***Mat8180Assignment2-Results.xlsx***

Report – 40 MARKS

The following models are split training, validating and testing set. The MLR and RF use the validation set for training, please see matlab code for further information.

Figures of (ANN, MLR, RF).

ANN – Artificial Neural Network

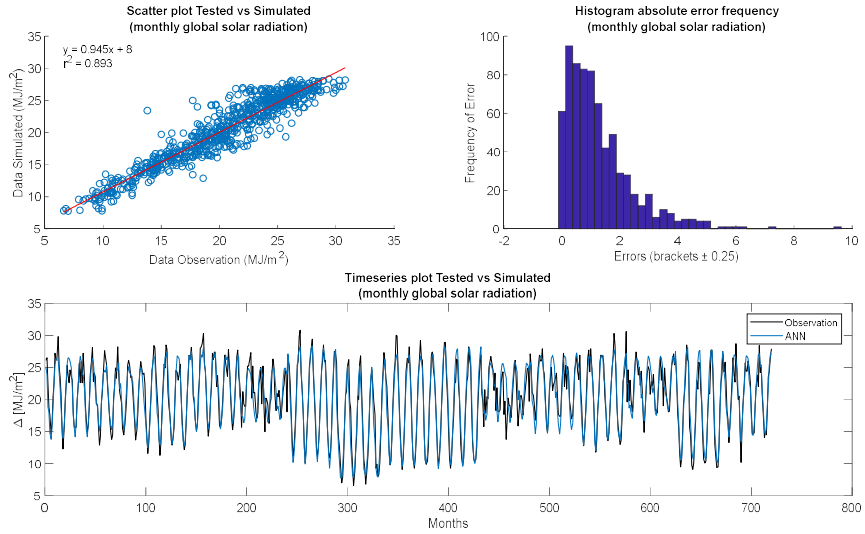

Fig. 1 *ANN Performance figures (top left – scatter obs vs sim; top right - histogram absolute error freq; bottom – time series obs vs sim)*

Above figure 1 represents performance of ANN model against observation given a number of metrics. The top left, scatter plot shows high positive correlation ($r^2 = 0.89$) of observation to simulation, indicate by the cluster of points around the red line. Further the visualisation of absolute difference between observed and simulation values is depicted on the top right histogram. The majority of error peaking below the frequency of one and a right skewed distribution with one outlier at approximately 10. Time series shows the change in observed variable given the change in months for both simulated (blue) and test (black).The ANN follows closely the test although it is noted on extreme (min and max) values the model does not match as well.

Testing Metrics	Model – Run 10
ANN – (`'logsig', 'purelin','trainlm' 10 hidden neurons`)	
r	0.945
ENS	0.892
d	0.936
Pdv	0.707
RMSE	1.728
MAE	1.278

Table 1 *ANN Performance*

Table 1, are error and correlation metrics of simulated values against observed. The pearson correlation, $r = 0.945$ shows a close relationship of how observed and simulated values perform. Further error investigation of goodness of fit ($RMSE$) and the deviations from observed regardless of the sign(MAE) both indicate a well-fitting model (below 2). ENS, d and Pdv agree with low error metrics showing inversely high values.

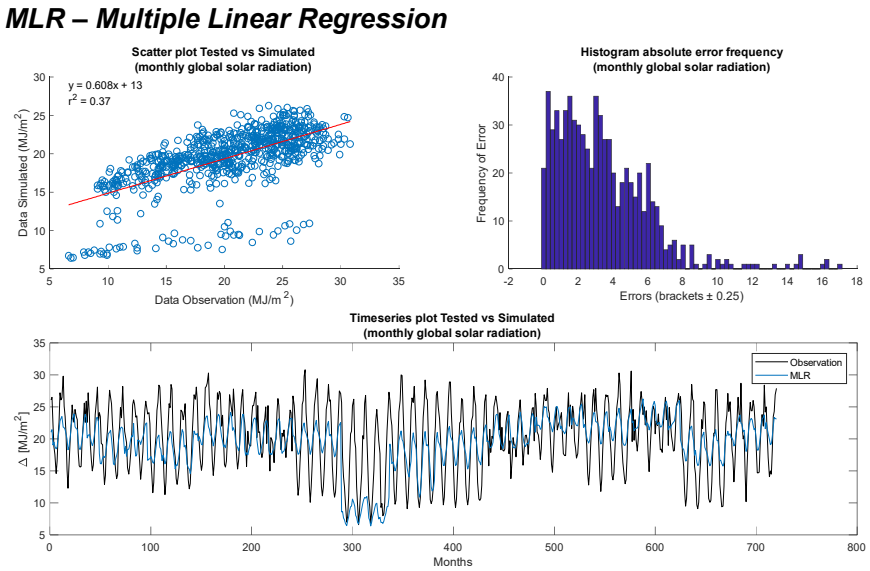

Fig. 2 *MLR Performance figures (top left – scatter obs vs sim; top right - histogram absolute error freq; bottom – time series obs vs sim)*

Above figure 2 represents performance of MLR model against observation given a number of metrics. The top left, scatter plot shows low positive correlation ($r^2 = 0.37$) of observation to simulation, indicative by the sparse cluster of points around the red line.

Further the visualisation of absolute difference between observed and simulation values is depicted on the top right histogram. The majority of error is spread with two peaks with approximately frequency around one and two. Although right skewed distribution there is a left tail indicating large errors above 10. Time series shows the change in observed variable given the change in months for both simulated (blue) and test (black).The MLR does not follows the test as well as the ANN, more representative of the average.

Testing Metrics	
	MLR
r	0.608
ENS	0.328
d	0.379
Pdv	0.229
RMSE	4.306
MAE	3.367

Table 2 *MLR Performance*

Table 2, are error and correlation metrics of simulated values against observed. The pearson correlation, $r = 0.608$ is an average relationship of how observed and simulated values perform. Error investigation of goodness of fit ($RMSE$) and the deviations from observed regardless of the sign (MAE) agree of a higher error model, indicative of a poorer fitting model in comparison (above 3). ENS, d and Pdv agree with higher error metrics showing inversely low values.

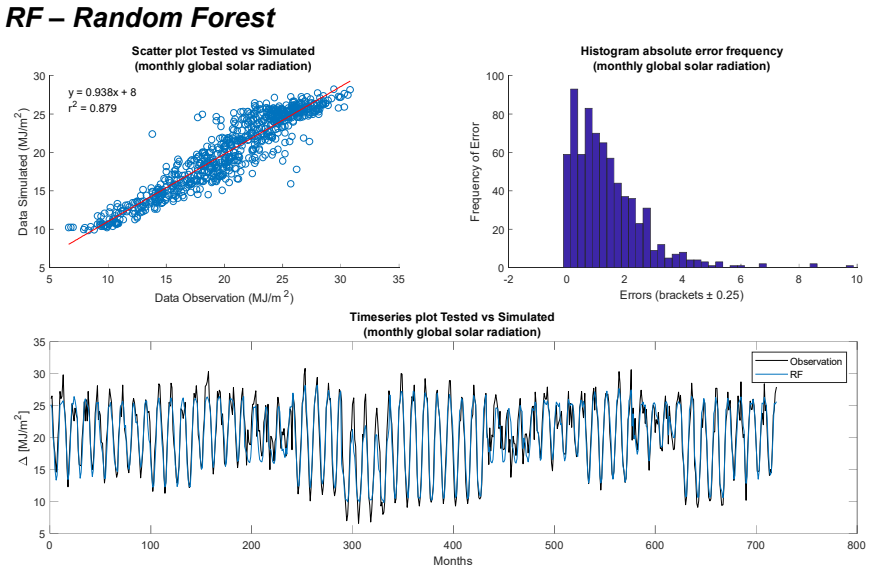

Fig.3 *RF Performance figures (top left – scatter obs vs sim; top right - histogram absolute error freq; bottom – time series obs vs sim)*

Above figure 3 represents performance of RF model against observation given a number of metrics. The top left, scatter plot

shows high positive correlation ($r^2 = 0.938$) of observation to simulation, indicative by the close cluster of points around the red line, with some notable outliers.
Further the visualisation of absolute difference between observed and simulation values is depicted on the top right histogram. The majority of error is spread with peaks with approximately frequency around one, with a right skewed distribution and possible outliers around 9 and 10. Time series shows the change in observed variable given the change in months for both simulated (blue) and test (black).The RF does well to follow the test, however again but to a lower extent then the MLR it is more representative of the average.

Testing Metrics	
	RF
r	0.939
ENS	0.879
d	0.922
Pdv	0.686
RMSE	1.826
MAE	1.370

Table 3 *RF Performance*

Table 3, are error and correlation metrics of simulated values against observed. The pearson correlation, $r = 0.939$ is an high value indicating a good relationship of how observed and simulated values perform. Error investigation of goodness of fit ($RMSE$) and the deviations from observed regardless of the sign (MAE) agree of a low error model, indicative of a good fitting model in comparison (below 2). ENS, d and Pdv agree with low error metrics showing inversely high values.

Challenge Problem *– 40 MARKS*

Provide code for ANN neuron selection model.

```
% optimize your model, for Neurons.
for hn = 1:1:250
    net = newff(minmax(valP),[hn 1],{'logsig' 'purelin'},'trainlm');
    net=init(net);
    net.trainParam.show=100;
    net.trainParam.epochs=50;
    net.trainParam.goal=0.0001;
    net.performFcn='mse';
    w1 = net.IW{1,1};
    w2 = net.LW{2,1};
    b1 = net.b{1};
    b2 = net.b{2};


    %Build on validation set
```

```
    [net, tr] = train(net, valP, valT); % fix
    a = sim(net, pn);
    av=a';
    z = [a' qn'];
    pval=chkIn;  % test
    qval=chkOut;
    y = sim(net, chkIn');
    dataSimTest = y';
    zv = [y' chkOut];
     %Asses
    [r,ENS,d,Pdv,RMSE,MAE,PI] =asseMetric(chkOut,dataSimTest);
    TestErrorsMetrics(:,hn) = PI';
    SimulatedResults(:,hn)=dataSimTest;

end
```

Observe the testing performance as the ANN model where hn is varied from 1 to 250.

As algorithm increase in neurons, the run takes longer. To speed computation matlab code utilising GPU and use of core of CPU were implemented. However, convergence became exponentially longer as the hidden neurons increased.

Model	Neuron Rank	r	RMSE	MAE
ANN	**Run 10**	**0.945**	**1.728**	**1.278**
	Run 12	0.942	1.772	1.291
	Run 2	0.941	1.858	1.308
	Run 18	0.941	1.830	1.331
	Run 47	0.936	1.880	1.361
	Run 8	0.943	1.772	1.364
	Run 7	0.930	1.960	1.478
	Run 9	0.930	2.044	1.484
	Run 20	0.925	2.123	1.581
	Run 59	0.924	2.093	1.602

Table 4 *Top ten Rank Neuron (hn) performance*

Plot graph of RMSE as a function of hn, testing period.

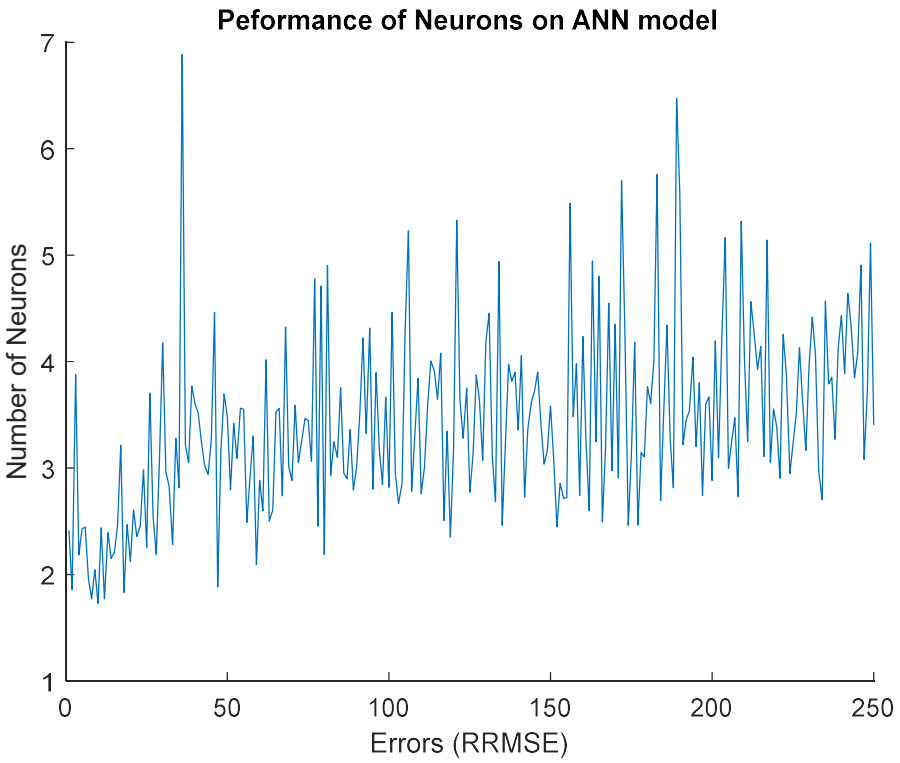

Figure 4 *Function of RRMSE with Neurons on ANN model*

Make a conclusion on the optimal number of hidden neurons for your best ANN model.

Run 10 that is 10 hidden neurons produce the more accurate ANN modelled based on performance metrics. From testing stage optimum epoch was set.

Conclusions

Artificial Neural Network (ANN) and Random Forest (RF) model significantly outperforms the Multiple Linear Regression (MLR). With ANN simulated variables performing slightly better than the RF model in correlation and error metrics.

Due to the nature of the algorithm, ANN and RF can describe relationship of non-linear while MLR forecast values with the use of the linear relations between regressors.

ANN are organized in layers made up of a number of interconnected nodes containing an activation function. Patterns are presented to the network via the input layer, which communicates to one or more hidden layers where the actual processing is done as a system of weighted connections. The hidden layers then link to an 'output layer' where the answer is outputted. As there are different learning rules, the best is chosen by trial in our study, and the backpropagational learning is very slow in training

and requiring a lot of epochs (chosen in training) before convergence and is a disadvantage compared to the other methods.

RF classifies predictor variables by splitting value by usefulness in final model. If variable is less informative the variable will be less useful at the prediction time. Where a random subset of variables at a node is asses, this is done by at each split with best entropy will be kept, while others will be discarded and possibly (sampled with replacement) tested in subsequent node. As there can be a large number of deep trees, the process can be seen at a black box. However, the final train model can be decomposed to show the strongest contributors to predicting the response variable. Therefore the more random a variable or feature the less contribution and uncertainties obtained in final model parameters.

MLR does not account for the non-linear relationship of predictor variables and response value. The MLR with larger samples may perform better with linear relationships and interpretable effects. And due to method of finding parameters sum of least squares, a less computationally once off simple matrix operation the training and testing times is significantly lower than that of RF or ANN.

In conclusion the application of predicting monthly global solar radiation based on given predictors has shown with the ANN that the internal patterns have been train well. As the MLR has not forecasted as well, we can suggest that the variables relationship is dynamic and non-linear – suited to ANN numerical solution. Although ANN can capture many types of relationship that conventional methods (MLR) can't we showed disadvantage of ANN heavy computational requirements when training.

Reference

Dawson, C.W., Abrahart, R.J., See, L.M., 2007. HydroTest: a web-based toolbox of evaluation metrics for the standardised assessment of hydrological forecasts. nviron. Model. Softw. 22 (7), 1034–1052.

Deo, R.C., Byun, H.-R., Adamowski, J.F., Begum, K., 2016. Application of effective drought index for quantification of meteorological drought events: a case study in Australia. Theor. Appl. Climatol.1–21.

Legates, D.R., McCabe, G.J., 1999. Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation. Water Resour. Res. 35, 233–241.

Krause, P., Boyle, D.P., Base, F., 2005. Comparison of different efficiency criteria for hydrological model assessment. Adv. Geosci. 5, 89–97.

Willmott, C.J., 1981. On the validation of models. Phys. Geogr. 2 (2), 184–194.

Willmott, C.J., 1982. Some comments on the evaluation of model performance. Bull. Am. Meteorol. Soc. 63 (11), 1309–1313.

Willmott C.J.: On the evaluation of model performance in physical geography, Spatial statistics and models, 443460, 1984.

Appendix

```
% Optimization of Artificial Neural Network Models and Evaluation
against
%Random Forest and Multiple Linear Regression
% MAT8190 Assignment 2 - MATHEMATICS STATISTICS COMPLEMENTARY STUDIES
% Author: John Worrall
% Description: RF modeld  and MLR for prediction
% Requirments: Excel file (RandomForestDataA1-SD-VERSION.xlsx)
%----------------------------

clear
clc
close all

% Number of models (Default 5 times)
nModels = 5;

% Choose Options (CO)
% 1 - Artifical Neural Network
% 2 - Multiple Linear Regression Model
% 3 - Random Forest
CO = 3;

%randomise parameters, ensure the results are reproducible.
rand('twister', 123);
s = RandStream('mlfg6331_64');

%IO
trnData=xlsread('RandomForestDataA2-SD-VERSION.xlsx','Training');
chkData=xlsread('RandomForestDataA2-SD-
VERSION.xlsx','CheckData_Testing');

%Prepare Train/Test
trnIn = trnData(:,1:6);
trnOut = trnData(:,7);
chkIn = chkData(:,1:6);
chkOut = chkData(:,7);

% Use GPU if available
ngpus=gpuDeviceCount;
disp([num2str(ngpus) ' GPUs found'])
if ngpus>0
    lgpu=1;
    disp('GPU found')
    useGPU='yes';
else
    lgpu=0;
    disp('No GPU found')
    useGPU='no';
end
% Find number of cores
ncores=feature('numCores');
disp([num2str(ncores) ' cores found'])
% Find number of cpus
import java.lang.*;
r=Runtime.getRuntime;
ncpus=r.availableProcessors;
```

```
disp([num2str(ncpus) ' cpus found'])
if ncpus>1
    useParallel='yes';
else
    useParallel='no';
end
[archstr,maxsize,endian]=computer;
disp(['This is a ' archstr ' computer that can have up to '
num2str(maxsize) ' elements in a matlab array and uses ' endian '
byte ordering.'])

% Set up the size of the parallel pool if necessary
npool=ncores;

% Opening parallel pool
if ncpus>1
    tic
    disp('Opening parallel pool')
    % first check if there is a current pool
    poolobj=gcp('nocreate');
% If there is no pool create one
    if isempty(poolobj)
        command=['parpool(' num2str(npool) ');'];
        disp(command);
        eval(command);
    else
        poolsize=poolobj.NumWorkers;
        disp(['A pool of ' poolsize ' workers already exists.']);
    end
        % Set parallel options
        paroptions = statset('UseParallel',true);
        %Set parrellel streams to have same seed value for repeated
results
        %paroptions = statset('UseParallel',true,'Streams', s,
'UseSubStreams',true);
        toc
end

if CO == 1
% %run pacf - on objective variable (for additional data)
% obVar = [trnOut; chkOut];
% figure
% subplot(2,1,1)
% autocorr(obVar)
% subplot(2,1,2)
% parcorr(obVar)
% %determine t-4 lag

%add lag variable (t-4)
%remove first 4 rows
%AORDOut(4,:)=[];

%Normalise
% x = [min(trnData,[],1);max(trnData,[],1)]
% b = bsxfun(@minus,trnData,x(1,:));
% b = bsxfun(@rdivide,b,diff(x,1,1))

%=============================================================
%devided into training/validation/testing
% 10% cut from training set - 1123(training) + 125(validation)
```

```
% Split training set into 40:60 (Training:Validation) (499:749)
pn=trnIn(1:499,:)';
qn=trnOut(1:499,:)';

valP=trnIn(500:1248,:)';
valT=trnOut(500:1248,:)';

% %Create ANN
% net = newff (minmax(pn),[3 1],{'tansig' 'purelin'},'trainlm');
% %net = newff (minmax(pn),[250 1],{'tansig' 'purelin'},'trainlm');
%
% %optimize your model, for HiddenTransfer functions
% ParA =
{'purelin','logsig','purelin','logsig','purelin','logsig','purelin','
logsig','purelin','logsig','purelin','logsig'};
% ParB =
{'logsig','purelin','logsig','purelin','logsig','purelin','logsig','p
urelin','logsig','purelin','logsig','purelin'};
% ParC =
{'trainlm','trainlm','trainbfg','trainbfg','traingdx','traingdx','tra
inscg','trainscg','traincgf','traincgf','traincgp','traincgp'};
% %2*6 = 12 combinations
% for hn  = 1:1:12
%     net = newff (minmax(pn),[3 1],{ParA{hn} ParB{hn}},ParC{hn});
%     net=init(net);
%     net.trainParam.show=100;
%     net.trainParam.epochs=500;
%     net.trainParam.goal=0.0001;
%     net.performFcn='mse';
%
%     w1 = net.IW{1,1};
%     w2 = net.LW{2,1};
%     b1 = net.b{1};
%     b2 = net.b{2}; %check
%
%     [net, tr] = train(net, pn, qn); % fix
%     a = sim(net, pn);
%     av=a';
%     z = [a' qn'];
%     pval=chkIn;
%     qval=chkOut;
%     y = sim(net, chkIn')
%     dataSimTest = y';
%     zv = [y' chkOut];
%
%     %Asses
%     [r,ENS,d,Pdv,RMSE,MAE,PI] =asseMetric(chkOut,dataSimTest);
%     HiddenTransferMetrics(:,hn) = PI';
%     HiddenTransferResults(:,hn)=dataSimTest;
% end
% %From above, run 2 is clearly the best with 0.89 r, 2.345 RMSE,
1.786 MAE
% %That is - the following.. parameter A 'logsig', parameter B
'purelin'
% %prameter C 'trainlm'


% optimize your model, for Neurons.
%
% %Train
```

```
% net = newff(minmax(pn),[30 1],{'logsig' 'purelin'},'trainlm');
%     net=init(net);
%     net.trainParam.show=100;
%     net.trainParam.epochs=500;
%     net.trainParam.goal=0.0001;
%     net.performFcn='mse';
%     w1 = net.IW{1,1};
%     w2 = net.LW{2,1};
%     b1 = net.b{1};
%     b2 = net.b{2};
%     [net, tr] = train(net, pn, qn); % fix
%     a = sim(net, pn);
%
%
% for hn = 1:1:250
%     net = newff(minmax(valP),[hn 1],{'logsig'
'purelin'},'trainlm');
%     net=init(net);
%     net.trainParam.show=100;
%     net.trainParam.epochs=50;
%     net.trainParam.goal=0.0001;
%     net.performFcn='mse';
%     w1 = net.IW{1,1};
%     w2 = net.LW{2,1};
%     b1 = net.b{1};
%     b2 = net.b{2};
%
%
%     %Build on validation set
%     [net, tr] = train(net, valP, valT); % fix
%     a = sim(net, pn);
%     av=a';
%     z = [a' qn'];
%
%
%     pval=chkIn;  % test
%     qval=chkOut;
%     y = sim(net, chkIn');
%     dataSimTest = y';
%     zv = [y' chkOut];
%
%     %Asses
%     [r,ENS,d,Pdv,RMSE,MAE,PI] =asseMetric(chkOut,dataSimTest);
%     TestErrorsMetrics(:,hn) = PI';
%     SimulatedResults(:,hn)=dataSimTest;
%
% end

%RMSE as a function of hn
rmseModel=xlsread('Mat8180Assignment2-Results.xlsx','ANN_HN_RMSE');
rmseModel = rmseModel(:,1);
    figure
    hold on;
    plot(rmseModel)
    title('Peformance of Neurons on ANN model');
    xlabel('Errors (RRMSE)');
    ylabel('Number of Neurons');

    hold off;
```

```
% ============================================================
%    ANN results
bestModel=xlsread('Mat8180Assignment2-
Results.xlsx','ANN_NuronsRuns');
bestModel = bestModel(:,10);
    subplot(2,2,1)
    %figure
    hold on;
    scatter(chkOut,bestModel)
    %title('Scatter plot Tested vs Simulated (monthly global solar
radiation)');
    title({'Scatter plot Tested vs Simulated','(monthly global solar
radiation)'})
    xlabel('Data Observation (MJ/m^2)');
    ylabel('Data Simulated (MJ/m^2)');
    %add line
    nnR = corrcoef(chkOut,bestModel);
    nnR = nnR(1,2);
    coeffs = polyfit(chkOut, bestModel, 1);
    % Get fitted values
    fittedX = linspace(min(chkOut), max(chkOut), 198);
    fittedY = polyval(coeffs, fittedX);
    C = round(polyval(coeffs, fittedX(1,1),0));
    plot(fittedX, fittedY, 'r-', 'LineWidth', 1);
    txt1 = ['y = ' num2str(round(nnR,3)) 'x + ' num2str(C)];
    ylim =get(gca,'ylim')-7
    text(min(chkOut),max(ylim), txt1);
    text(min(chkOut),max(ylim)-2, ['r^2 = '
num2str(round(nnR*nnR,3))]);
    hold off;

  %histogram
    subplot(2,2,2)
    dataErr = abs(chkOut - bestModel);
    hmin = min(dataErr);
    hmax = max(dataErr);
    hold on;
    hist(dataErr,[hmin:0.25:hmax]);
    %title('Histogram absolute error frequency(monthly global solar
radiation)');
    title({'Histogram absolute error frequency','(monthly global
solar radiation)'})
    ylabel('Frequency of Error');
    xlabel('Errors (brackets ± 0.25)');
    hold off;


% Timeseries
    %figure
    subplot(2,1,2)
    plot(chkOut,'k');
    hold on;
    plot(bestModel);
    %title('Timeseries plot Tested vs Simulated (monthly global solar
radiation)');
    title({'Timeseries plot Tested vs Simulated','(monthly global
solar radiation)'})
    xlabel('Months')
    ylabel('\Delta [MJ/m^2]')
```

```
    var = {{'Observation','ANN'},'Location','northeast'}
    legend(var{:})

%legend({'Test','arima1','arima2','t1mamdami','t1Sugerno','allSugerno
','nt1Sugerno','en50ANFIS','en100ANFIS','SOMANFIS'},'Location','north
west','NumColumns',2)
    hold off

%-------------------------------------------------------------
% Multiple Linear Regression Model
elseif CO == 2
    pn=trnIn(1:499,:)';
    qn=trnOut(1:499,:)';
    valP=trnIn(500:1248,:)';
    valT=trnOut(500:1248,:)';

    %MLR with validation set
    In = valP';
    Out = valT';

%     In = trnIn;
%     Out = trnOut;
    c = regress(Out,In);
    cIn = c';

    y2= cIn.*chkIn; %predict(c,chkIn);
    dataSimMLR = sum(y2,2);
    dataObs = chkOut;

    %Assement error metrics --------------------------------

[nnR,nnENS,nnD,nnPDEV,nnRMSE,nnMAE,nnPI]=asseMetric(dataObs,dataSimML
R);
    ErrorsMLR = nnPI;
    %asseMetricVis(dataObs,dataSimMLR,nnR,1,'Multiple Linear
Regression');

%VISUALS
%    MLR results
    subplot(2,2,1)
    %figure
    hold on;
    scatter(chkOut,dataSimMLR)
    %title('Scatter plot Tested vs Simulated (monthly global solar
radiation)');
    title({'Scatter plot Tested vs Simulated','(monthly global solar
radiation)'})
    xlabel('Data Observation (MJ/m^2)');
    ylabel('Data Simulated (MJ/m^2)');
    %add line
    nnR = corrcoef(chkOut,dataSimMLR);
    nnR = nnR(1,2);
    coeffs = polyfit(chkOut, dataSimMLR, 1);
    % Get fitted values
    fittedX = linspace(min(chkOut), max(chkOut), 198);
    fittedY = polyval(coeffs, fittedX);
    C = round(polyval(coeffs, fittedX(1,1),0));
    plot(fittedX, fittedY, 'r-', 'LineWidth', 1);
    txt1 = ['y = ' num2str(round(nnR,3)) 'x + ' num2str(C)];
    ylim =get(gca,'ylim')-1
```

```
    text(min(chkOut),max(ylim), txt1);
    text(min(chkOut),max(ylim)-2, ['r^2 = '
num2str(round(nnR*nnR,3))]);
    hold off;
  %histogram
    subplot(2,2,2)
    dataErr = abs(chkOut - dataSimMLR);
    hmin = min(dataErr);
    hmax = max(dataErr);
    hold on;
    hist(dataErr,[hmin:0.25:hmax]);
    %title('Histogram absolute error frequency(monthly global solar
radiation)');
    title({'Histogram absolute error frequency','(monthly global
solar radiation)'})
    ylabel('Frequency of Error');
    xlabel('Errors (brackets ± 0.25)');
    hold off;
% Timeseries
    %figure
    subplot(2,1,2)
    plot(chkOut,'k');
    hold on;
    plot(dataSimMLR);
    %title('Timeseries plot Tested vs Simulated (monthly global solar
radiation)');
    title({'Timeseries plot Tested vs Simulated','(monthly global
solar radiation)'})
    xlabel('Months')
    ylabel('\Delta [MJ/m^2]')
    var = {{'Observation','MLR'},'Location','northeast'}
    legend(var{:})

%legend({'Test','arima1','arima2','t1mamdami','t1Sugerno','allSugerno
','nt1Sugerno','en50ANFIS','en100ANFIS','SOMANFIS'},'Location','north
west','NumColumns',2)
    hold off

%-------------------------------------------------------------
elseif CO == 3

    %Random Forest Model variables
    tic % starts the timer.
    leaf=5; % this number could be varied.
    ntrees=800; % this number could be varied.
    fboot=1; % this number could be varied.
    surrogate='on'; % this could be set 'on' or 'off'

    pn=trnIn(1:499,:)';
    qn=trnOut(1:499,:)';

    valP=trnIn(500:1248,:);
    valT=trnOut(500:1248,:);

%      for x = 1:nModels
%          b = TreeBagger(ntrees,valP,valT,'Method','regression',...
%         'oobvarimp','on','surrogate',surrogate,...
%         'minleaf',leaf,'FBoot',fboot,'Options',paroptions);
%
%         %Predict with traing ------------------
```

```
%         y = predict(b, valP);
%         simulatedTrain = y;
%         dataObsTrain = trnOut;
%         mseTrain = oobError(b,'mode','ensemble'); %single MSE for
RF
%
%         %Predict with testing ------------------
%         dataSim = predict(b,chkIn);
%         dataObs = chkOut;
%         runs(:,x) = dataSim;
%
%         %Assement check error metrics ----------------------------
---
%
[nnR,nnENS,nnD,nnPDEV,nnRMSE,nnMAE,nnPI]=asseMetric(dataObs,dataSim);
%         ErrorsTest = nnPI;
%         %asseMetricVis(dataObs,dataSim,nnR,1,'Random Trees - Test
Errors');
%         runsErrorTestRF(:,x) = nnPI;
%
%      end

 % Random Forest visuals
RFModel=xlsread('Mat8180Assignment2-Results.xlsx','RF_Result');
RFModel = RFModel(:,11);

   subplot(2,2,1)
    %figure
    hold on;
    scatter(chkOut,RFModel)
    %title('Scatter plot Tested vs Simulated (monthly global solar
radiation)');
    title({'Scatter plot Tested vs Simulated','(monthly global solar
radiation)'})
    xlabel('Data Observation (MJ/m^2)');
    ylabel('Data Simulated (MJ/m^2)');
    %add line
    nnR = corrcoef(chkOut,RFModel);
    nnR = nnR(1,2);
    coeffs = polyfit(chkOut, RFModel, 1);
    % Get fitted values
    fittedX = linspace(min(chkOut), max(chkOut), 198);
    fittedY = polyval(coeffs, fittedX);
    C = round(polyval(coeffs, fittedX(1,1),0));
    plot(fittedX, fittedY, 'r-', 'LineWidth', 1);
    txt1 = ['y = ' num2str(round(nnR,3)) 'x + ' num2str(C)];
    ylim =get(gca,'ylim')-2
    text(min(chkOut),max(ylim), txt1);
    text(min(chkOut),max(ylim)-2, ['r^2 = '
num2str(round(nnR*nnR,3))]);
    hold off;
  %histogram
    subplot(2,2,2)
    dataErr = abs(chkOut - RFModel);
    hmin = min(dataErr);
    hmax = max(dataErr);
    hold on;
    hist(dataErr,[hmin:0.25:hmax]);
    %title('Histogram absolute error frequency(monthly global solar
radiation)');
```

```
    title({'Histogram absolute error frequency','(monthly global
solar radiation)'})
    ylabel('Frequency of Error');
    xlabel('Errors (brackets ± 0.25)');
    hold off;
% Timeseries
    %figure
    subplot(2,1,2)
    plot(chkOut,'k');
    hold on;
    plot(RFModel);
    %title('Timeseries plot Tested vs Simulated (monthly global solar
radiation)');
    title({'Timeseries plot Tested vs Simulated','(monthly global
solar radiation)'})
    xlabel('Months')
    ylabel('\Delta [MJ/m^2]')
    var = {{'Observation','RF'},'Location','northeast'}
    legend(var{:})

%legend({'Test','arima1','arima2','t1mamdami','t1Sugerno','allSugerno
','nt1Sugerno','en50ANFIS','en100ANFIS','SOMANFIS'},'Location','north
west','NumColumns',2)
    hold off

end
```