Using Neural Networks Committee Machines to Improve Outcome Prediction Assessment in Nonlinear Regression

Élia Yathie Matsumoto and Emílio Del-Moral-Hernandez

Abstract—This study proposes a methodology to improve nonlinear regression model prediction assessment by the construction of a model for error pattern recognition to estimate whether the model outcome prediction value will fall outside the model confidence interval. The methodology was evaluated on six experiments using five widely known public databases from UCI Machine Learning Repository. The essays provided evidences that the pattern recognition models were able to identify observations, in the testing datasets, that are more likely to produce higher error values, and improve the overall outcome of the nonlinear regression models predictions.

I. INTRODUCTION

COUNTLESS numbers of error analysis and confidence interval construction methods to improve regression model prediction assessment have been proposed in technical literature. They can be simple, like the conventional linear regression error analysis method [1] that, given a very restrictive set of assumptions, assumes that the errors produced by the linear model for the training dataset can be adjusted by a normal distribution, and this information can be used to predict the model behavior for the testing dataset. In econometrics, Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models assume that the variance of the errors of the model is a function of the variance of the original variable and the errors analysis is handled as a linear autoregressive problem [2],[3].

In the case of nonlinear regression models, error analysis can reach higher level of complexity and sophistication [4]. Particularly with reference to neural network models confidence interval construction, there are studies that explore maximum likelihood properties [5], bootstrap techniques [6], Bayesian approximation approaches [7] and novel frameworks like Lower Upper Bound Estimation (LUBE) [8],[9] and Conformal Prediction [10],[11]. Each one of them has its specific assumptions, advantages and limitations.

The method presented in this study does not intend to replace or over perform anyone of them, on the contrary, the goal is to provide complementary information to improve nonlinear regression model prediction assessment, no matter which approach is used to construct the model and its confidence interval.

The basic idea is to evaluate how much to believe in a single instance by providing an estimation of whether the model individual outcome values, in the testing dataset, will fall out of the model confidence interval.

It proposes to classify the observations considering the errors produced by the nonlinear regression model in two categories, in or out of a specific confidence interval, and then constructing a model to recognize this pattern.

This pattern recognition model will attempt to capture the numerical limitations of the regression model and when it is applied to the testing dataset, it is expected to be able to indicate for which observations the regression model are more likely to produce errors with higher values.

Although the underlying concept of the methodology is simple, the challenge is the design of this specific pattern recognition model because the dataset to be handled is supposed to be highly imbalanced.

For the purposes of this study, all models were constructed using Feedforward Multilayered Perceptron Neural Network Committee Machine architecture (NNCM), and the method was evaluated on six experiments using five widely known public databases from UCI Machine Learning Repository: Abalone Age [12], Concrete Compressive Strength [13], Energy Efficiency [14], Boston Housing Data [15], and Parsonsons Telemonitoring [16].

II. METHODOLOGY

The objective of this research is to delineate a method to improve nonlinear regression model prediction assessment by extracting knowledge from the errors produced by the model with the training dataset.

The proposition is to extract this knowledge by classifying the observations in two categories, taking into account the errors of the regression model, whether they fall out of the confidence interval, and then constructing another model to recognize this pattern, using, as input, the regression model outcome for the training dataset and the training dataset itself.

The concept of the method is very straightforward, and can be applied to any nonlinear regression model (1), that is able to provide a confidence interval \( [\delta^\text{inf}_q, \delta^\text{sup}_q] \), where \( \alpha \) is the level of confidence, such as in formula (2).

\[
\hat{Y} = G(X) + \varepsilon \quad (1)
\]

\[
P(\delta^\text{inf}_q < (\varepsilon = Y - \hat{Y}) < \delta^\text{sup}_q) > \alpha \quad (2)
\]

Given the calibrated regression model and its confidence interval, the first step is to classify the training dataset...
observations using the model errors, and create a new variable, observation error class \( C_e \), defined as in (3).

\[
C_e = \begin{cases} 
0 & \text{if } \alpha \in [\delta_{\text{Lim}}^{\text{Inf}}, \delta_{\text{Lim}}^{\text{Sup}}] \\
1 & \text{otherwise}
\end{cases}
\]  

(3)

Then, this new variable \( C_e \) is used to design a model for pattern recognition, as in (4), using \( Z_{\text{train}} = [Y_{\text{train}} X_{\text{train}}] \), as input data, where \( Y_{\text{train}} \) is the outcome of the regression model for \( X_{\text{train}} \); and as output data, \( C_{\text{train}} \), the training dataset observation error class.

\[
\text{PatterRec_Model}(Z_{\text{train}}) = C_{\text{train}}
\]  

(4)

Thus, this new model is conditioned to the numerical limitations of the original nonlinear regression model and the training dataset. Assuming that the training and testing datasets have the same probability distribution, when applied to the testing dataset, the model for pattern recognition is prone to identify observations whose errors are expected to fall out of the confidence interval and classify them as positive, with value 1 (in this paper, the observation will be said to be a “classified observation”).

Therefore, the classified observations can be analyzed in detail, or just be excluded from the testing dataset in order to improve the overall outcome in testing dataset. The process does not improve the quality of the regression model itself, but it intends to be able to provide useful information for data analysis and decision making.

As much higher the level of confidence \( \alpha \) is, fewer error values are predicted to fall out of the confidence interval, so the variable \( C_e \) (observation error class) is supposed to be imbalanced, with low number of observation with value 1.

Under this circumstance, as already mentioned, the most relevant mission to be accomplished is the design of the model for pattern recognition, because standard pattern recognition algorithms that use MSE (Mean Squared Error) value optimization strategy are prone to work well with balanced data, but to be biased towards the majority class in the case of imbalanced data.

The values of the classical two-by-two confusion matrix are used for metric evaluation. Fig. 1 shows the confusion matrix labels for a two-class classification.

<table>
<thead>
<tr>
<th>True Class</th>
<th>Prediction Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative (0)</td>
<td>True-Negative (TN)</td>
</tr>
<tr>
<td>Positive (1)</td>
<td>False-Negative (FN)</td>
</tr>
</tbody>
</table>

Fig. 1. Confusion matrix (also called contingency table).

More specifically, four evaluation metrics are observed: Accuracy, Precision, Specificity, and Sensitivity [17].

The first metric, Accuracy, measures the percentage of correct predictions made by the model (5).

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]  

(5)

The second metric, Precision, measures the percentage of positive predictions made by the model that are correct (6).

\[
\text{Precision} = \frac{TP}{TP + FP}
\]  

(6)

The third metric, Specificity, measures the percentage of true negative patterns that are correctly detected by the model, or the accuracy on the negative cases (7).

\[
\text{Specificity} = 1 - \frac{FP}{TN + FP}
\]  

(7)

The fourth metric, Sensitivity, also called Recall, measures the percentage of true positive patterns that are correctly detected by the model, or the accuracy on the positive cases (8).

\[
\text{Sensitivity} = \frac{TP}{TP + FN}
\]  

(8)

Ideally, the models would be expected to produce high percentage values for all metrics, however, according to the literature [17],[18], in the case of extremely imbalanced datasets, the Sensitivity is often very low. In practice, it means that the rare cases are hard to be identified.

III. EXPERIMENTS

All models were constructed based on Feedforward Multilayered Perceptron Neural Network Committee Machine architecture (NNCM).

The choice was based on NNCM improved prediction accuracy, strong robustness and generalization attested by several studies [19],[20], moreover NNCM has well established methods for confidence interval construction by combining the different outcomes from all the individual models [7],[21].

A. NNCM Regression Models

The NNCM regression models were constructed as committee machines composed of individual feedforward multilayered perceptron neural networks.

The size of the NNCM (number of individual neural networks) was defined, after testing the values of 5, 10, 20, and 50. For each case, 20 models were constructed, and the averages of the MSE values were statistically compared, using hypothesis test with 90% level of confidence. The architecture that produced the smallest MSE with training dataset was chosen.

Each neural network model (9) was designed with similar architecture, using three layers: Input, Hidden and Output.

\[
\hat{y} = MLP(X)
\]  

(9)

The number of neurons in the Input layer was defined by the number of input parameters. The Output layer was created with one neuron. For the Hidden layer, the number of neurons was defined using the same procedure described above, statistically comparing the MSE values produced by NNCM regression models using three different numbers of neurons in the Hidden layer: the same number of input parameters, twice and three times the number of input parameters.

The training method applied to calibrate the neural networks was the Levenberg-Marquardt backpropagation
algorithm with MSE performance function.

The NNCM regression models were trained using subsets of the training datasets defined according to the Bagging (bootstrap aggregating) [6] ensemble learning method. For each neural network, the original training dataset was divided into two bootstrap sample subsets: a training subset with 85% of the original data used to calculate the weights and bias of the neural network, and a testing subset with the remaining 15% used for cross-validation to avoid overfitting.

The NNCM regression model outcome was composed according to the formula (10).

\[ \hat{Y} = \sum_{i=1}^{N} w_i \cdot MLP_i(X) \quad (10) \]

The weights, \( w_{i=1...N} \), were defined by optimizing the MSE error function.

Regarding features selection, as suggested by several studies [22]-[25], it was used the leave-one-out cross validation method.

**B. NNCM Regression Models Confidence Intervals**

The confidence intervals were constructed by adjusting Student’s t location-scale (or just t) distributions to the errors produced by the models with the training dataset.

A random variable \( x \) with density function as in (11), is said to have a t location-scale distribution with parameters \( \mu, \sigma^2, \nu \) (\( \nu \) degrees of freedom).

\[ \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sigma\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left[1 + \frac{(x-\mu)^2}{\nu\sigma^2}\right]^{-\frac{\nu+1}{2}} \quad (11) \]

The t location-scale distribution is frequently used to modeling data distribution with slightly skewed and heavier tails than normal distribution [1], [23].

**C. NNCM Models for Error Pattern Recognition**

Similarly to what was done for the NNCM regression models, the NNCM models for error pattern recognition were constructed as committee machines composed of individual feedforward three layered perceptron neural networks (12).

\[ \hat{c}_i = PRMLP_i(Z) \quad (12) \]

Also, the same processes applied to NNCM regression models were used to define the size of the NNCM, the number of neurons in the Hidden layer, and for selection feature.

Specifically, aiming at improving accuracy with imbalanced datasets, the performance function definition, the training subsets bootstrap process and the NNCM composition were modified.

In the case of the performance function, instead of the standard MSE, it was used a weighted MSE function considering the cost matrix (13) to balance the false-positive and false-negative misclassification. \( FP \) penalty is the value that penalizes the false-positive outcomes, and \( FN \) penalty, the false-negatives.

\[ Cost \ matrix = \begin{bmatrix} 0 & FP \ penalty \\ FN \ penalty & 0 \end{bmatrix} \quad (13) \]

Additionally, the bootstrap sampling process was adapted to create the subsets keeping their imbalanced proportion as close as possible to the original training dataset.

The NNCM model for pattern recognition outcome was composed according to the formula in (14).

\[ \hat{c}_i = \sum_{i=1}^{N} \omega_i \cdot PRMLP_i(Z) \quad (14) \]

The weights, \( \omega_{i=1...N} \), were defined by optimizing the same previously described weighted MSE function, with the same cost matrix.

**IV. NUMERICAL RESULTS**

The methodology was evaluated on six experiments using five widely known public databases from UCI Machine Learning Repository:

- **Abalone:** Abalone age [12]
  - Origin/Donor: Prof. Sam Waugh
  - Dept of Computer Science, University of Tasmania
- **Concrete:** Concrete Compressive Strength [13]
  - Origin: Prof. I-Cheng Yeh
  - Dept of Information Mgmt, Chung-Hua University
- **Energy:** Energy Efficiency [14]
  - Origin: Tsanas, A. and Xifara, A.
  - Indus.& Applied Mathematics, University of Oxford
- **Housing:** Boston Housing Data [15]
  - Origin: Harrison, D. and Rubinfeld, D.L
  - StatLib Library (Carnegie Mellon University)
- **Parkinsons:** Parkinsons Telemonitoring [16]
  - Origin: Athanasios Tsanas & Max Little
  - University of Oxford, in collaboration with 10 medical centers in the US and Intel Corporation who developed the telemonitoring device to record the speech signals.

**A. Data Description**

Each one of the five databases was split into two sub-datasets: a training dataset with approximately 92% of the total observations and a testing dataset with approximately 8% of the total observations.

All models were design and calibrated using the training datasets and evaluated with the testing datasets.

**TABLE I** shows a summary of the size of the five databases information.

<table>
<thead>
<tr>
<th>Database</th>
<th>Number of input variables</th>
<th>Number of output variables</th>
<th>Training Dataset Size</th>
<th>Testing Dataset Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abalone</td>
<td>8</td>
<td>1</td>
<td>3943</td>
<td>334</td>
</tr>
<tr>
<td>Concrete</td>
<td>8</td>
<td>1</td>
<td>948</td>
<td>82</td>
</tr>
<tr>
<td>Energy</td>
<td>8</td>
<td>2</td>
<td>707</td>
<td>61</td>
</tr>
<tr>
<td>Housing</td>
<td>15</td>
<td>1</td>
<td>466</td>
<td>40</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>9</td>
<td>1</td>
<td>5405</td>
<td>470</td>
</tr>
</tbody>
</table>

Six experiments, one for each output variable, were developed using these five databases.

The dataset Energy has two responses variables, Cooling Load and Heating Load. For this reason, from this point, there will be two references to the dataset Energy experiments: EnergyC, associated to response variable
Cooling Load, and EnergyH, to Heating Load.

**B. NNCM Regression Models Outcomes**

All NNCM regression models were implemented in MATLAB with Neural Network, Optimization and Statistics Toolbox, and constructed according to the description and following the procedures detailed in Session III.A.

For all experiments, the NNCM regression models were composed of **20 individual neural networks** with hidden layer with the number of neurons equals to **twice the number of input parameters**.

Leave-one-out cross validation were applied and all features were selected to compose the input data.

TABLE II shows the NNCM regression models performance metrics, RMSE (Root Means Squared Error), R-Squared and Adjusted R-Squared for all training datasets.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>RMSE</th>
<th>R-Squared</th>
<th>Adj R-Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abalone</td>
<td>1.5408</td>
<td>0.6161</td>
<td>0.6160</td>
</tr>
<tr>
<td>Concrete</td>
<td>3.1788</td>
<td>0.9618</td>
<td>0.9618</td>
</tr>
<tr>
<td>EnergyC</td>
<td>0.6602</td>
<td>0.9951</td>
<td>0.9951</td>
</tr>
<tr>
<td>EnergyH</td>
<td>0.3148</td>
<td>0.9990</td>
<td>0.9990</td>
</tr>
<tr>
<td>Housing</td>
<td>1.2601</td>
<td>0.9796</td>
<td>0.9796</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>3.1173</td>
<td>0.9011</td>
<td>0.9011</td>
</tr>
</tbody>
</table>

The same performance metrics for the testing datasets are displayed on TABLE III.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>RMSE</th>
<th>R-Squared</th>
<th>Adj R-Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abalone</td>
<td>1.6604</td>
<td>0.6115</td>
<td>0.6104</td>
</tr>
<tr>
<td>Concrete</td>
<td>4.8049</td>
<td>0.9143</td>
<td>0.9132</td>
</tr>
<tr>
<td>EnergyC</td>
<td>0.6743</td>
<td>0.9960</td>
<td>0.9959</td>
</tr>
<tr>
<td>EnergyH</td>
<td>0.4135</td>
<td>0.9987</td>
<td>0.9986</td>
</tr>
<tr>
<td>Housing</td>
<td>3.3789</td>
<td>0.8679</td>
<td>0.8644</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>3.1843</td>
<td>0.9056</td>
<td>0.9054</td>
</tr>
</tbody>
</table>

The figures at APPENDIX I (Fig 3. to Fig. 8) show the models outputs scatter plot (observations against estimations) and it is possible to visually confirm that NNCM regression models achieved reasonable effectiveness, even in the case of Abalone experiment.

Any other nonlinear regression model could be used in the experiments; however, since the model for pattern recognition depends on the errors produced by the regression models, the methodology may not work properly in cases of regression models with extremely low performance.

**C. NNCM Regression Models Confidence Intervals**

The confidence intervals were constructed by adjusting t location-scale distributions to the errors produced by the NNCM regression models with the training datasets, and calculating the values of the inverse cumulative distribution functions for the t distributions, evaluated at the probability $\alpha = 90\%$.

TABLE IV shows inferior and superior limits values of the confidence intervals. It is possible to verify that the error distributions were slightly skewed.

As consequence of calculating the values of the inverse cumulative distribution functions for the adjusted t distributions, evaluated at the probability $\alpha = 90\%$, the percentage of observations, in the training dataset, located inside of the confidence intervals resulted close to 90%. As expected, lower percentage values were achieved in the testing datasets.

The confidence interval limits were used to classify the observations in the training dataset according to the definition in Section II, formula (3), and this classification was used to construct the models for the error pattern recognition.

**D. NNCM Models for Error Pattern Recognition Outcomes**

All models were implemented in MATLAB with Neural Network Toolbox, Optimization Toolbox, and Statistics Toolbox, and constructed according to the description and following the procedures detailed in Session III.C.
Again, the NNCM models were composed of **20 individual neural networks** hidden layer with the number of neurons equal to **twice the number of input parameters**. Also, all features were selected to be used as input data.

The penalty values of the cost matrices were defined after testing different values for the FN penalty, keeping the FP penalty valued fixed in 1.

In order to balance the false-positive and false-negative misclassification, in the training datasets, the following criterion was applied: increase the FN penalty, starting from 1.0 with step 0.1, trying to increment the number of classified observation without raising to much the number of false-positive.

The FN penalty values defined for each experiment are displayed on **TABLE VI**.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>FN Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abalone</td>
<td>1.1</td>
</tr>
<tr>
<td>Concrete</td>
<td>1.5</td>
</tr>
<tr>
<td>EnergyC</td>
<td>1.5</td>
</tr>
<tr>
<td>EnergyH</td>
<td>1.7</td>
</tr>
<tr>
<td>Housing</td>
<td>3.0</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>1.5</td>
</tr>
</tbody>
</table>

As shown in **TABLE VII**, the experiments confirmed the information mentioned in the literature that, for extremely imbalanced dataset, **Sensitivity** is often low.

This indicates that the models were able to classify only few observations; however, as indicated by reasonable high **Precision** values, these classifications were correct, in most part of the cases, in training and most importantly, in testing datasets.

In other words, the models for pattern recognition datasets were able to identify observations in testing dataset that are more likely to produce higher error values.

As expected, all experiments were able to produced high **Accuracy** and **Specificity** percentage values denoting good performance in correct classifying the majority class.

**TABLE VII** shows the confusion matrices and the four evaluation metrics (Accuracy, Precision, Specificity, and Sensitivity) of the models in all experiments, for training and testing datasets.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Training Dataset</th>
<th>Testing Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abalone</td>
<td>3414 30</td>
<td>294 3</td>
</tr>
<tr>
<td>Accuracy</td>
<td>90.03%</td>
<td>89.52%</td>
</tr>
<tr>
<td>Precision</td>
<td>60.53%</td>
<td>62.50%</td>
</tr>
<tr>
<td>Specificity</td>
<td>99.13%</td>
<td>98.99%</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>11.53%</td>
<td>13.51%</td>
</tr>
<tr>
<td>Concrete</td>
<td>839 22</td>
<td>64 1</td>
</tr>
<tr>
<td>Accuracy</td>
<td>91.35%</td>
<td>92.93%</td>
</tr>
<tr>
<td>Precision</td>
<td>55.10%</td>
<td>80.00%</td>
</tr>
<tr>
<td>Specificity</td>
<td>97.44%</td>
<td>98.46%</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>31.03%</td>
<td>23.53%</td>
</tr>
<tr>
<td>EnergyC</td>
<td>613 17</td>
<td>55 2</td>
</tr>
<tr>
<td>Accuracy</td>
<td>91.51%</td>
<td>90.16%</td>
</tr>
<tr>
<td>Precision</td>
<td>66.67%</td>
<td>71.43%</td>
</tr>
<tr>
<td>Specificity</td>
<td>97.30%</td>
<td>96.15%</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>44.16%</td>
<td>55.56%</td>
</tr>
</tbody>
</table>

The figures at APPENDIX II (Fig 9. to Fig. 14) depict similar information in graphic format, showing all error values, the classified observations and confidence interval, in the testing datasets.

As noted before, the information about the classified observations can be used for data analyses or decision making, for instance, to assist input data measurement error identification, or just to define observations to be excluded from the testing dataset in order to improve the overall prediction outcome.

**TABLE VIII** summarizes the total effect of the exclusion of the classified observations from the testing dataset. It shows the percentage of improvement in RMSE, R-Squared and Adjusted R-Squared values compared to the values on **TABLE III**, for all experiments.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>RMSE</th>
<th>R-Squared</th>
<th>Adj R-Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abalone</td>
<td>1.5374</td>
<td>0.6240</td>
<td>0.6229</td>
</tr>
<tr>
<td>Improv.%</td>
<td>7.40%</td>
<td>2.04%</td>
<td>2.05%</td>
</tr>
<tr>
<td>Concrete</td>
<td>4.2649</td>
<td>0.9215</td>
<td>0.9205</td>
</tr>
<tr>
<td>Improv.%</td>
<td>11.24%</td>
<td>0.79%</td>
<td>0.79%</td>
</tr>
<tr>
<td>EnergyC</td>
<td>0.5197</td>
<td>0.9970</td>
<td>0.9969</td>
</tr>
<tr>
<td>Improv.%</td>
<td>22.94%</td>
<td>0.10%</td>
<td>0.10%</td>
</tr>
<tr>
<td>EnergyH</td>
<td>0.4086</td>
<td>0.9987</td>
<td>0.9987</td>
</tr>
<tr>
<td>Improv.%</td>
<td>1.20%</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>Housing</td>
<td>2.6446</td>
<td>0.9073</td>
<td>0.9044</td>
</tr>
<tr>
<td>Improv.%</td>
<td>21.73%</td>
<td>4.54%</td>
<td>4.63%</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>2.9744</td>
<td>0.9173</td>
<td>0.9171</td>
</tr>
<tr>
<td>Improv.%</td>
<td>6.59%</td>
<td>1.30%</td>
<td>1.30%</td>
</tr>
</tbody>
</table>

V. Concluding

Although the percentage of improvement in RMSE, R-Squared, and Adjusted-Squared values shown in **TABLE VIII** seem arguably small, the models for pattern recognition were consistently capable to correctly predict classified observations in testing datasets, accomplishing to yield high Precision and Specificity percentage values.

Ultimately, the experiments provided evidence that the methodology presented in this work is able to properly identify observations for which the regression models can
potentially fail, and these individualized indications may be used as an additional assessment to provide better support for tasks like measurement error detection, mistyped data recognition, outlier identification, and general data analysis and investigation.

The method only assumes that training and testing datasets follow the same probability distribution, and it can be applied to any kind of regression model with any kind of confidence interval construction method.

Regarding the model for error pattern recognition, the key point is the capability of the model of properly handling imbalanced data, because standard classification algorithms tend to misclassify the minority class observations and generate high number of false-positives. Due to this requirement, the methodology would not be recommended for databases with a relatively small number of observations.

Further research could evaluate the methodology on different databases, using other kinds of nonlinear regression models, other confidence interval construction methods, and other models for pattern recognition.

**APPENDIX I: REGRESSION MODELS OUTPUTS**

![Fig. 3. NNCM regression model outcome for Abalone.](image)

![Fig. 4. NNCM regression model outcome for Concrete.](image)

![Fig. 5. NNCM regression model outcome for EnergyC.](image)

![Fig. 6. NNCM regression model outcome for EnergyH.](image)

![Fig. 7. NNCM regression model outcome for Housing.](image)

![Fig. 8. NNCM regression model outcome for Parkinsons.](image)

**APPENDIX II: TESTING DATASETS ERRORS PLOT**

![Fig. 9. Abalone: errors plot for testing dataset with confidence interval and classified observations.](image)
REFERENCES


Fig. 10. Concrete: errors plot for testing dataset with confidence interval and classified observations.

Fig. 11. EnergyC: errors plot for testing dataset with confidence interval and classified observations.

Fig. 12. EnergyH: errors plot for testing dataset with confidence interval and classified observations.

Fig. 13. Housing: errors plot for testing dataset with confidence interval and classified observations.

Fig. 14. Parkinsons errors plot for testing dataset with confidence interval and classified observations.