Engineering and Technology Journal e-ISSN: 2456-3358
Volume 04 Issue 07 July- 2019, Page No.-618-624
DOI:10.31142/etj/v4i7.01, I.F. - 4.449
© 2019, ETJ

Comparison of the Prediction Accuracy thru Artificial Neural Networks with Respect to Multiple Linear Regression using R

Carlos Eduardo Belman López[1], José Alfredo Jiménez García[2], José Antonio Vázquez López[3]

[1,2,3]National Technological of Mexico in Celaya, Department of Industrial Engineering; Av. García Cubas 1200, Celaya, Guanajuato, México.

ABSTRACT: In this paper a comparison of the prediction accuracy of a response variable given a set of predictors was made using statistical and artificial intelligence methods using R language. The compared methods were the multiple lineal regression by the least squares method and the back propagation network (BP).The goal was to decrease the reducible error when predicting the output variable and being able to select a model, an indispensable step when developing a prediction model. The methodology consisted in two validation strategies. The first strategy measured just the training error rate using 100% of data. The second strategy used a validation set approach, dividing the observations in two parts, 50% is for a training set used to fit the models, and the remaining 50% is for a validation set used to test the fitted models. This methodology made it possible a comparison between the training error rate and testing error rate. The measures utilized to evaluate the efficiency were the sum of squared error (SSE) and the coefficient of determination (R^2). The results showed that BP network can significantly decrease the reducible error improving the prediction accuracy. It is important to highlight the prediction accuracy with new or unseen observations not used during the training instead of how well the models work with the training data.

KEYWORDS: Prediction, Artificial Neural Networks, Multiple Linear Regression, Multilayer Perceptron, R Language

I. INTRODUCTION

Day to day, it is often required to solve problems involving sets of variables that have some inherent relationship with each other [18]. Then it may be interesting to develop a prediction model, that is, a procedure to estimate the value of a response variable given a set of independent variables. There are many examples of this requirement, in areas such as chemistry, manufacturing, sales [14], marketing, financial[3], etcetera. This is also a very studied case in areas of recent creation such as statistical learning[4], machine learning [9] and data mining[19].

In this setting, the independent variables might go by different names, such as predictors, regressors, features, inputs, or sometimes just variables. The response variable is often called the dependent variable or output variable and is typically denoted using the symbol ‘y’.

Suppose, it is observed a quantitative response y and a set of p different predictors$\mathbf{X} = (\mathbf{X1}, \mathbf{X2}, \dots, \mathbf{Xp})$. It is assumed that there is a relationship between ‘y’ and X, which can be written in the form, $\mathbf{y} = \mathbf{F}(\mathbf{X}) + \boldsymbol{\varepsilon}$, where F is an unknown function that represents the real relationship between y and X, and ϵ represents the random error term. It is possible to make a prediction of ‘y’, using$\hat{\mathbf{y}} = \boldsymbol{f}(\mathbf{X})$, where f represents the estimated function for ‘F’. f is often treated as a black box, in the sense that one does not usually worry about the exact form of f, if it yields accurate predictions for ‘y.

Generally, f will not be a perfect estimation for the real F, and the prediction accuracy can be affected by two kind of errors called reducible error and irreducible error.

The reducible error may be decreased by selecting the appropriate learning method (statistical model or ANN model) to estimate the real F. The irreducible error is always present and does not allow us to get a perfect estimation for F, it is because ‘y’ is also in function of ε, and ε cannot be estimated using the set of predictors. ε may also contain significant variables that were no considered for the learning method and therefore they cannot be used to make a prediction [7].

Model selection is an indispensable step in the process of developing a functional prediction model or a model for understanding the data. Cross-validation is one of the most commonly used methods of evaluating predictive performances of a model. Basically, based on data splitting, part of the data is used for fitting each competing model and the rest of the data is used to measure the predictive performances of the models by the validation errors, and the model with the best overall performance is selected [20].

[12] and [16] presented different works showing the assessment of the predictive ability for different linear regression models using cross validation. The caret package [8] from R[13]also provides many functions to estimate the accuracy and attempt to streamline the process for creating predictive models.

ANN are structures that have a mathematical and statistical behavior with the property of learning, that is, the acquisition of knowledge, that in most cases is based on examples. This learning is produced by a computer style called in parallel, that tries to simulate some capabilities of the human brain, for this reason, they are defined as artificial neural networks to distinguish them from biological models. The three key elements of the biological systems that pretend to emulate the artificial ones are parallel processing, distributed memory and adaptability. ANN have also shown application in areas such as associative memory, optimization, pattern recognition, prediction, classification [2] and decision making[17].

Unlike the parametric methods of inferential statistics, ANN models, such as, the multilayer perceptron (MLP) and other AAN models, are free model estimators, since they do not impose any functional form of departure [10].

It is very important to highlight the existence of clear relationships between the MLP and statistical methods, such as linear regression, logistic regression or discriminant analysis.

The contribution of this paper focuses on finding the method that provides the best prediction accuracy of a response variable, when there are several predictors and then decreasing the reducible error comparing a statistical method among an ANN model. In specific, the predictive methods to be compared are least squares statistical method,and the ANN model is a Multilayer Perceptron (MLP) with Backpropagation algorithm (BP) as a learning strategy. R software was used to generate the models and neuralnet package [5] was used to train the ANN model.

The measures used to evaluate the models were the sum of squared error (SSE) and R^2 statistic. Functions to calculate both measures were provided. The methodology consisted in two validation strategies, the first strategy measures just the training error rate using the whole data set to train the models. The second strategy used a validation set approach dividing the available set of observations into two parts, 50% used to train the models and the remaining 50% in a validation set to test the fitted models with observations unseen during the training. The two validation strategies made it possible a comparison between the training error rate and testing error rate.

A. *Artificial Neural Networks (ANN)*

ANN are structures with a mathematical and statistical behavior with the capacity of learning, that is, the acquisition of knowledge that in most cases is based on examples. This learning is produced by a computer style called in parallel, that tries to simulate some of the capabilities in our brain, for this reason they are defined as artificial neural networks to distinguish them from biological models. The key elements in the biological systems that pretend to emulate the artificial are the parallel processing, the distributed memory and the adaptability[17].

ANN structure is generally conformed by nodes, input set, synaptic weights, propagation rule or base function, activation function and output function.

Figure 1 shows the basic ANN structure with its elements. The node is usually defined as the basic element in the network, which receives a set of inputs (x_j) from the outside or from the output of other nodes. Each entry has a specific associated weight (w_{ij}), which will be increased or decreased in the learning process. Each node applies a base function (u_i) such as the sum of the entries multiplied for the weights. The output value for the base function is transformed by a non-linear activation function f (u_i). The most common activation functions are the sigmoidal, gaussian, step and hyperbolic tangent functions. The output and input variables (y_i, x_j) can be both binary and continuous, depending on the model[10].

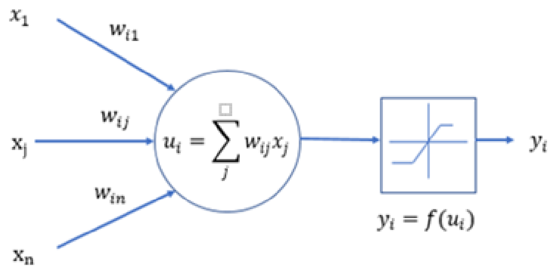

Figure 1. The neuron and its elements.

The MLP is an ANN conformed by multiple layers and it has the capability to act as a universal function approximator through the BP learning algorithm, forcing the neural network to contain at least one hidden layer with enough non-linear units. The MLP can also approximate any function using the input variables to predict the output variable. The BP learning algorithm has the capability to provide the network with generalization so that the model obtains a correct output for an input data set that had not been used before[15].

B. *Multiple Linear Regression*

This is the classic statistical method for predicting an output variable when there are one or more regressors. It is also a parametric method because assumes about the functional form of the relationship between the response (y) with the regressor (x). This means, linear regression assumes the relationship between the response (y) with the regressor (x) is linear with the form shown in equation 1,

$$\mathbf{y} = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 \mathbf{x} + \boldsymbol{\varepsilon}, \qquad (1)$$

where, β_0 is the intersection, β_1 is the slope and ε is the random error with zero mean and variance σ^2[6].

In many cases when there are k regressors that help to explain Y. In this case, the structure of multiple linear regression could be written as equation 2,

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k + \varepsilon \qquad (2)$$

An important aspect within regression analysis is the estimation of the regression coefficients. In the case that exits k independent variables, the estimated response is obtained from equation 3, also known as the multiple linear regression equation, where ŷ is the predicted or adjusted value and each regression coefficient $ß_i$ is estimated through $b_{i.}$

Due to the relationship had been assumed as linear, the problem of estimating the coefficients is greatly simplified. The most common approach to fitting the model is known as least squares method.

$$\hat{y} = b_0 + b_1 x_1 + \dots + b_k x_k \qquad (3)$$

It is evident that the adjusted regression equation is an estimate of the true regression function [18].

C. *Assessing Model Accuracy*

No one method dominates all others over all possible data sets. According to the data set, one method may work better, but some other method may work better on a similar but different data set. Hence it is an important task to decide for any given set of data which method produces the best results. To evaluate the performance of a regression method on a given data set, we need some way to measure how well its predictions match the observed data. It is needed to quantify how well the predicted response value for a given observation is close to the true response value for that observation [7]. SSE and R^2 are measures widely used in regression and prediction cases.

The difference between observation y_i and the corresponding predicted value $\hat{y}_i$, denoted as $e_i = y_i - \hat{y}_i$, it is called residue. The sum of squares of the residuals, or the sum of the squares of the error (SSE), is given by equation 4.

$$SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad (4)$$

The corrected total sum of squares (CTSS) represents the variation in the response values that would ideally be explained in the model and is calculated as shown in equation 5[6].

$$CTSS = \sum_{i=1}^{n} (y_i - \bar{y}_i)^2 \qquad (5)$$

The coefficient of determination (R^2) is defined as a measure of the proportion of variability explained by the adjusted model [18]. The way to calculate R^2 statistic is expressed in equation 6.

$$R_2 = 1 - SSE/CTSS \qquad (6)$$

II. BODY TEXT

The goal of this paper focused on comparing and finding the method that provides a better prediction accuracy for the response variable, when there are several predictors while decreasing the reducible error.

The predictive methods that were compared are least squares statistical method and Back propagation network (BP). The measures used to evaluate the models were SSE and R^2.

Methodology consisted in two validation strategies, the first strategy measured only the training error rate using 100% of the data set and the second strategy used a validation set approach dividing the available set of observations into two parts, 50% is for a training set used to fit the models and the remaining 50% is for a validation set used to test the fitted models with unseen observations. R language was used as software tool for constructing the regression models by the least squares method, as well as for the training and testing the MLP +BP.

The proposed method, is shown in Figure 2, is a stepwise method, which details are below:

- Step 1. Selecting the dataset to test the investigation.
- Step 2. Training the linear regression model by the least squares method using the two validation strategies.
- Step 3. Predicting the response variable using the fitted lineal models. First predicting 100% of observations and second the 50% of new observations.
- Step 4. Normalize data, parameter selection and MLP training using the two validation strategies.
- Step 5. Predicting the response variable using the trained MLP and denormalize the data. First predicting 100% of observations and second the 50% of new observations.
- Step 6. Developing programming functions to calculate the SSE, R^2 and print the results.

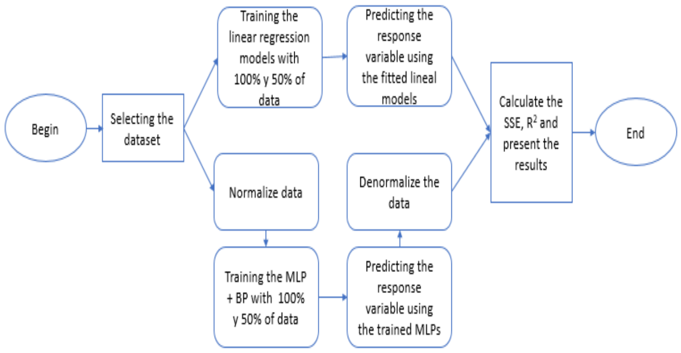
Begin
Selecting the dataset
Training the linear regression models with 100% y 50% of data
Predicting the response variable using the fitted lineal models
Normalize data
Training the MLP + BP with 100% y 50% of data
Predicting the response variable using the trained MLPs
Denormalize the data
Calculate the SSE, R² and present the results
End

Figure 2. Method.

A. *Step 1. Selecting the dataset to test the investigation.*

The data set consists of a group of light trucks with engines that use diesel as fuel, that were tested to know if humidity, air temperature and barometric pressure have some influence on the amount of nitrous oxide they emit (in ppm). The emissions were measured at different times and in different experimental conditions. The example was extracted from [18].Below, the code to create the data matrix:

```
# import the packages to be used.
library(neuralnet)
# Seed is useful to get same results in random generation
set.seed(99)
# data set to be used
>y<-c(0.9,0.91,0.96,0.89,1,1.1,1.15,1.03,0.77,1.07,1.07,
0.94,1.1 ,1.1,1.1,0.91,0.87,0.78, 0.82,0.95)
> x1<-c(72.4,41.6,34.3,35.1,10.7,12.9,8.3,20.1,72.2,24,23.2,
47.4,31.5,10.6,11.2,73.3,75.4,96.6,107.4,54.9)
> x2<-c(76.3,70.3,77.1,68,79,67.4,66.8,76.9,77.7,67.7,76.8,
86.6,76.9,86.3,86,76.3,77.9,78.7,86.8,70.9)
> x3<-c(29.18,29.35,29.24,29.27,29.78,29.39,29.69,29.48,
29.09,29.6,29.38,29.35,29.63,29.56,29.48,29.4, 29.28,29.29,
29.03,29.37)
> data<-data.frame(y,x1,x2,x3)
```

The y variable represents the amount of nitrous oxide emitted, x1 represents the humidity, x2 represents the air temperature and x3 represents barometric pressure.

For the validation set approach the data set is split in two parts, first part contains 50% of data set and it is used for training and the remaining 50% contains new observation not used during the training only for validation. Sample is generated using the 'sample' function in R.

```
# Data set is split in two parts, when validation set is used.
# A sample with 10 observations is used for training,
# and the remaining for testing.
> training<-sample(1:20,10)
```

B. *Step 2. Training the linear regression model by the least squares method using the two validation strategies.*

Regression analysis was performed using the least squares method thru the 'lm' function in R. This function is in the core and it is not necessary to import any package. As told before two validation strategies were thought. The first uses 100% of data set, the second just uses 50% of data set.

```
# First linear regression model is fitted
# with 100% of the data set.
> lmodel1<-lm(y~x1+x2+x3)
# Second linear regression model is fitted
#with only 50% of the data set.
> lmodel2<- lm(y~x1+x2+x3, data[training,])
```

C. *Step 3. Predicting the response variable using the fitted lineal models.*

Function 'predict' was used to predict the values of the response variable 'y', using the fitted linear from step 2. First predicting 100% of observations and second the 50% of new observations.

```
#First fitted model is used to predict 100% of observations
# already used during the training phase.
>lrm.predict<- predict(lmodel1, data)
# Second fitted model is used to predict the remaining 50%
# of new observations not used during the training.
>lrm.predict2 <- predict(lmodel2, data[-training,])
```

D. *Step 4. Normalize data, parameter selection and MLP training.*

Neural networks are not easy to train and before starting the training, it is necessary to perform some data preparation. It is recommended to normalize the data before training an ANN. Avoiding normalization can lead to useless results or a very difficult training process with problems, such as, that most of the time the algorithm will not converge before the maximum number of allowed iterations. You can choose different methods to scale the data, for example, z-normalization or min-max scale. The min-max method was selected scaling the data in the interval [0,1]. In general, the scale in the intervals [0,1] or [-1,1] tends to give better results[1].

Therefore, the first step is to normalize the data by the min-max method, as shown below:

```
# Before to train the ANN model, the first step is to
normalize
# the data by the min-max method
>maxs<- apply(data, 2, max)
> mins <- apply(data, 2, min)
> scaled <- as.data.frame(scale(data, center = mins, scale =
maxs - mins))
```

1) *Parameter selection for the MLP:* To use an MLP, various parameters must be chosen, such as: the number of neurons in the input and output layer, the number of hidden layers and number of neurons per hidden layer, the learning algorithm, the activation function, the function for error calculation and function in the output layer.
    - Neurons in the input layer. This parameter was determined by the number of independent variables that form the training data.
    - Neurons in the output layer. The number of neurons in the output layer was equal to the number of response variables, that is, equal to the number of variables to be predicted.
    - Number of hidden layers. Other works have shown that a hidden layer is enough for most of the problems[10].
    - Number of neurons in the hidden layer. There is no universal rule regarding the number of neurons to be used in the hidden layer, although there are several empirical rules to determine this number. One of these rules with good acceptance says that the approximate number of neurons in the hidden layer is equal to the average number of neurons in the input and output layer[11]. Finally, 3 neurons in the hidden layer was used for this investigation.

2) *MLP Architecture:* Finally, for this MLP, 1 hidden layer was used, with a configuration 3: 3: 1, which indicates that the input layer has 3 neurons, there is 1 hidden layer with 3 neurons and 1 neuron in the output layer. The rest of the parameters used were back propagation algorithm as a learning strategy, hyperbolic tangent function as an activation function, the SSE as a function for error calculation and linear output, it means, the activation function was not applied to the output layer. Figure 3 shows the neural network that was trained using 100% of data, the training was carried out using the 'neuralnet' function of the "neuralnet" package [5]in R, as shown below:

```
# First neural network is fitted with 100% of the data set.
> mlp<-neuralnet(y~x1+x2+x3,scaled,hidden=3,algorithm= "backprop", learningrate=.01, linear.output = TRUE, threshold = 0.01,act.fct='tanh')
# Second neural network is fitted with only 50% of the data set.
> mlp2<-neuralnet(y~x1+x2+x3,scaled[training,],hidden=3, algorithm= "backprop", learningrate=.01, linear.output = TRUE, threshold = 0.01,act.fct='tanh')
> plot(mlp)
```

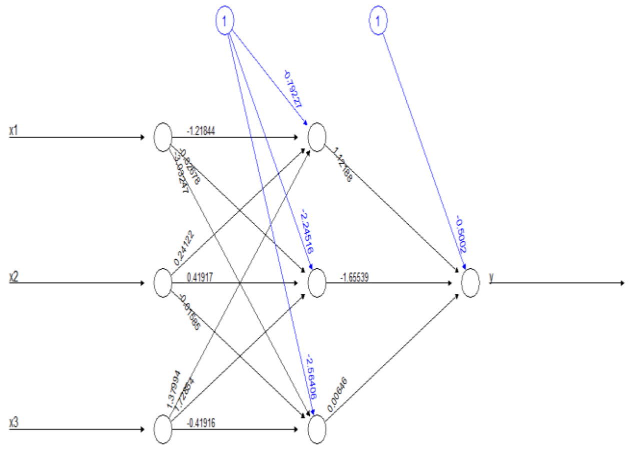

Figure 3. MLP + BP trained with 100% of data.

E. Step 5. Predicting the response variable using the trained MLP and denormalize the data.

The MLP trained in stage 4 was used to predict the response variable. The function 'compute' from "neuralnet" package [5]was used to predict the values, as shown below:

```
# First MLP is used to predict 100% of observations
# already used during the training phase.
>pred.mlp<- compute(mlp, scaled[ , c("x1","x2","x3")])
# Second MLP model is used to predict the remaining 50%
# of new observations not used during the training.
>pred.mlp2 <- compute(mlp2, scaled[ -training, c("x1","x2","x3")])
```

The network produced an output in normalized way, so the values must be returned to the original scale, in order to make a meaningful comparison.

```
# Predicted values have to be returned to the original scale,
# to make a meaningful comparison
# First prediction of 100% of data set already used in training
> prediction1 <- pr.mlp$net.result*(max(data$y)-min(data$y)) +min(data$y)
# Second prediction is for new observations not used during training
> prediction2 <- pr.mlp2$net.result*(max(data$y)-min(data$y)) +min(data$y)
```

F. Step 6. Developing programming functions to calculate the SSE, R^2 and print out the results.

Functions to calculate SSE, CTSS and R^2 were provided at this step. These functions were used to calculate the measures and print the results.

```
#Function to calculate SSE for ANN and linear models
sse<-function(x, y){
sum<-0
        for(i in 1:length(x)){
        sum<-sum+(x[[i]]-y[[i]])^2
        }
return(sum)
}
# training SSE for lineal model with 100% of data.
sse_lm<-sse(y,lrm.predict)
# training SSE for ANN model with 100% of data.
sse_rna<-sse(y,prediction1)
# test SSE for lineal model with 50% of unseen new data.
sse_lm2<-sse(data[-training,]$y,lrm.predict2)
# test SSE for ANN with 50% of unseen new data.
sse_rna2<-sse(data[-training,]$y,prediction2)
#Function to calculate CTSS
ctss<-function(x){
sum<-0
        for(i in 1:length(x)){
        sum<-sum+(x[[i]]-mean(x))^2
        }
return(sum)
}
# training-R^2 for lineal model with 100% of seen data.
R2_lm<-1-sse_lm/ctss(y)
# training-R^2 for ANN model with 100% of seen data.
R2_rna<-1-sse_rna/ctss(y)
# test-R^2 for lineal model with 50% of unseen new data.
R2_lm2<-1-sse_lm2/ctss(data[-training,]$y)
# test-R^2 for ANN model with 50% of unseen new data.
R2_rna2<-1-sse_rna2/ctss(data[-training,]$y)
#Print outputs
sse_lm
sse_lm2
sse_rna
sse_rna2
R2_lm
```

R2_lm2
R2_rna
R2_rna2

III. RESULTS

The results are explained below and showed in Table 1:

- Training error rate. It measured just the training error rate using 100% of the data set.
  - The least squares statistical method obtained an SSE = 0.050478 and $R^2 = 0.80$.
  - ANN model obtained SSE = 0. 02011639 and $R^2 = 0.92048$.
- Validation set approach. It divided the available set of observations into two parts, 50% was a training set intended to fit the models and the remaining 50% in a validation set to test the fitted models
  - The least squares statistical method obtained an SSE = 0. 05648 and $R^2 = 0.6243$.
  - ANN model obtained SSE = 0. 62432 and $R^2 = 0.8496$.

Table 1. SSE and R^2 for the Models.

	Linear Model	MLP + BP	Sample size for training
SSE	0.05048	0.02012	100%
R^2	0.80046	0.92048	100%
SSE	0.05649	0.02261	50%
R^2	0.62432	0.8496	50%

IV. CONCLUSIONS

Model selection is an indispensable step in the process of developing a prediction model or a model for understanding the data.ANN are structures with mathematical and statistical behavior and the capacity of learning. ANN are flexible tools that have already shown application as function estimators and now have increased the interest in using them as prediction tools in areas such as Data Mining and Machine Learning. For these purposes, R language is a flexible, open source and increasingly used tool in data science area when you can train statistical and artificial intelligence models. It is also easy to use, provides very good performance and does not consume a lot of computational resources.

Based on the results, it was concluded that MLP + BP can significantly decrease the reducible error and improve the prediction accuracy of the response variable according to the measures SSE and R^2. The BP network obtained better performance according to the measures in both validation strategies (first measuring just the training error rate and second considering a validation set approach not used during the training). The two validation strategies made it possible a comparison between the training error rate and testing error rate for the models. It is important to highlight the prediction accuracy for new observations not used during the training, instead of how well the methods work with the training data.

Researchers interested in following this work could concentrate on applying the models on large volumes of data (Big Data) and observe the prediction accuracy, training and execution time. About ANN setting, it may be considered using the MLP with another training strategy, for example, resilient back propagation or even use another ANN model, such as, Radial Base Function (RBF).Regarding the measures used to evaluate the prediction accuracy, the PRESS statistic and R^2_{pred}may be considered for assessing both statistical and ANN models.

REFERENCES

1. Alice, M. (2015, Septiembre 23). *R-Bloggers*. Retrieved from R-Bloggers: https://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/
2. Belman L., C., Vázquez L., J., & Hernández R., M. (2017). Classification of multivariate data using artificial neural networks, logistic regression and discriminant analysis. *International Congress "Academia Journals Celaya 2017"*.
3. Brummelhuis, R., & Luo, Z. (2017). Cds rate construction methods by Machine Learning Techniques. *Data Science Central*, 1-51. Retrieved from https://www.datasciencecentral.com/profiles/blogs/choice-of-k-in-k-fold-cross-validation-for-classification-in
4. Du, K.-L., & Swamy, M. (2014). *Neural Networks and Statistical Learning*. London: Springer.
5. Fritsch, S., & Frauke, G. (2016). neuralnet: Training of Neural Networks. Retrieved from https://CRAN.R-project.org/package=neuralnet
6. Hines, W., & Montgomery, D. (1996). *Probability and statistics for engineering and administration*. Ciudad de México: Continental S.A de C.V.
7. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. New York: Springer.
8. Kuhn, M. (2018). caret: Classification and Regression Training. Retrieved from https://CRAN.R-project.org/package=caret
9. Lantz, B. (2013). *Machine Learning with R*. Birmingham: Packt Publishing.
10. Martin del Brío, B., & Sanz Molina, A. (2002). *Neural networks and diffuse systems*. (2a. ed.). Madrid, España: Alfaomega & RA-MA.
11. Naved, I. (2016, Diciembre 26). *iqbalnaved. wordpress.com*. Retrieved from

https://iqbalnaved.wordpress.com/2016/12/26/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-network/

12. Picard, R., & Cook, R. (2012). Cross-Validation of Regression Models. *Journal of the American Statistical Association*, 575-583.
13. R Core Team. (2017). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from The R Project for Statistical Computing: https://www.R-project.org/
14. Ruelas Santoyo, E., & Laguna González, J. (2014). Predictive comparison based in neural network versus statistical methods to forecast sales. *Ingeniería Industrial. Actualidad y Nuevas Tendencias*, 91-105.
15. San Miguel Salas, J. (2016). Development with MATLAB a neural network to estimate the electrical energydemand (Master Thesis). Valladolid, España: Universidad de Valladolid.
16. Shao, J. (2012). Linear Model Selection by Cross-validation. *Journal of the American Statistical Association*, 486-494.
17. Torras P., S., & Monte, E. (2013). *Neural models applied in economics*. Barcelona, España: Addlink.
18. Walpole, R. E., Myers, R. H., Myers, S. L., & Ye, K. (2012). *Statistics and probability for science and engineering*. Ciudad de México: PEARSON.
19. Williams, G. (2011). *Data Mining with Rattle and R*. New York: Springer.
20. Zhang, Y., & Yang, Y. (2015). Cross-validation for selecting a model selection procedure. *Elsevier*, 95-112.