Fitting and classifying the Correlated Data Using the Neural Networks

Ahmed Mohamed mohamed El-Sayed
Al-Obour High Institute For Management & Informatics
Department of Basic Science.
Kilo 21 Cairo-Belbies Road, P.O. Box 27 - Obour City, Egypt.
atabl18@yahoo.com

ABSTRACT

Neural networks have been one of the fascinating machine learning models, not only because of the fancy back propagation algorithm but also because of their complexity. Neural networks have not been popular, partly because they were, and still are in some cases, computationally expensive and partly because they did not seem to yield better results when compared with simpler methods such as support vector machines. Nevertheless, the neural networks have raised attention and become popular.
In this paper, we are going to fit and classify neural networks using R package “neuralnet”, and fit a linear model "VGLM" and non-linear model “VGAM” models as a comparison applied on the environmental data. These data were collected from the Hunua Ranges, this is a small forest in southern Auckland, New Zealand. At 392 sites in the forest, the presence/absence of 17 plant species was recorded, as well as the altitude. Each site was of area size about 200 m^2. We will study the effect of the presence/absence these species on the sea level for all areas in

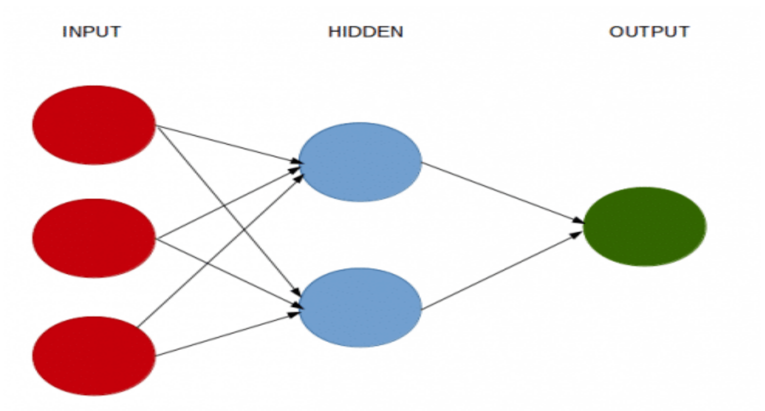
this forest. One of the most important procedures when forming a neural networks is data normalization. This involves adjusting the data to a common scale so as to accurately compare predicted and actual values. We used normalization process to scale data. Our neural networks has been created using the training data, then compare this to the test data to gauge the accuracy of the neural network forecast. We are used the neural networks and compares its results to tradition methods for regression and classification. Traditionally, the average MSE for the neural networks is lower than a linear model. But in our case the opposite is happened. Hidden layers use back propagation to optimize the weights of the input variables in order to improve the predictive power of the model. We study the effect of changing the number of hidden layers on the accuracy of our model. Also, traditionally an increasing of number of the hidden layers increase the accuracy of our model. Finally, A confusion matrix is used to determine the number of true and false generated by our predictions. This due to the accuracy rate of classification by divided the number of true on the total objects of a test data. A high accuracy rate expresses a good classifying.

Keywords: Vector Generalized Linear Model (VGLM); Regression, Classification; Neural Network (NN); R program; Traning data; Test Data; Corss-Validation.

Introduction

Neural networks, a beautiful biologically inspired programming paradigm which enables a computer to learn from observational data. Deep learning a powerful set of techniques for learning in neural networks. Neural networks and deep learning currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing. Neural networks have always been one of the fascinating machine learning models, not only because of the fancy back propagation algorithm but also because of their complexity (deep learning with many hidden layers) and structure inspired by the brain. Neural networks have not always been popular, partly because they were, and still are in some cases, computationally expensive and partly because they did not seem to yield better results when compared with simpler methods such as support vector machines. Nevertheless, Neural Networks have, once again, raised attention and become popular.

A neural network consists of: Input layers: Layers that take inputs based on existing data. Hidden layers: Layers that use back propagation to optimize the weights of the input variables in order to improve the predictive power of the model. Output layers: Output of predictions based on the data from the input and hidden layers. The next graphical representation of the model with the weights on each connection:



Our goal is to develop a neural network (NN) to determine if the explanatory variable(s) effect on the response variable or not. As such, we are using the neural network (NN) to solve a classification problem. By classification, we mean 1's where the data is classified by categories. e.g. a fruit can be classified as an apple, banana, orange, etc. In our dataset, we assign a value of 1 to a plant that effects on the level of altitude. We assign a value of 0 to a plant that does not effect on the level of altitude.

There is no fixed rule as to how many layers and neurons to be used. although there are several more or less accepted rules of thumb. Usually, if at all necessary, one hidden layer is enough for a vast number of applications. As far as the number of neurons is concerned, it is should be between the input layer size and the output layer size, usually 2/3 of the input size. At least in our experience testing again and again is the best solution since there is no guarantee that any of these rules will fit your model best. In a neural network graph, the black lines show the

connection between each layer and the weights, while the blue lines show the bias term added in each step. The bias can be thought as the intercept of a linear model.
Now, we can try to predict the values for the test set and calculate the MSE. Remember that the net will output a normalized prediction, so, we need to scale it back in order to make a meaningful comparison or just a simple prediction.
Apparently, the net is doing a better work than the linear model. We can see that the predictions made by the neural network are in general more concentrated around the regression line. A perfect alignment with the line, would indicates a MSE of 0 and thus an ideal perfect prediction than those made by the linear model. But in this paper the types of data (for example binary data) may be play to change this idea. It is maybe opposite.
Once again, we must be careful because these results depend on the train set. After that, we are going to be perform a cross validation in order to be more confident about the results. We need to check that no data point is missing, otherwise we need to fix the dataset. We proceed by randomly splitting the data into a train and a test set. One of the most important procedures when forming a neural network is data normalization. The normalization process is used to scale the data to be comparable. This involves adjusting the data to a common scale so as to accuracy compare predicted and actual values. Failure to normalize the data will typically result in the prediction value

remaining the same across all observations, regardless of the input values. Scale the data frame automatically using the scale function in R program to transform the data using the normalization technique. Before fitting a neural network, some preparations need to be done. As a first step, we are going to address data preprocessing. It is good practice to normalize your data before training a neural network. We cannot emphasize enough how important this step is depending on our dataset, there are two methods for scaling: Normalization, may lead to useless results or to a very difficult training process. Most of the times the algorithm will not converge before the number of maximum iterations allowed. Min-Max method usually scale the data in the interval [0,1]. Sometimes, scaling in the intervals [0,1] tends to give better results. We therefore scale and split the data before moving on. Then we fit a linear regression model and test it on the test set. Note that we are using the gml() function instead of the lm() this will become useful later when cross validating the linear model.

The sample (*x*, size) function simply outputs a vector of the specified size of randomly selected samples from the vector *x*. By default, the sampling is without replacement. index is essentially a random vector of indices. Riedmiller and Braun [1] presented a direct adaptive method for faster backpropagation learning. Intrators [2] have used the neural nets for interpretation of nonlinear models. Bishop[3] and Ripley [4] have devoted the

neural networks for pattern recognition. Wild [5] has presented a vector generalized additive models (VGAM). Wood[6] presented stable and efficient multiple smoothing parameter estimation for generalized additive models. Anastasiadis A. et. al.[7] have presented a new globally convergent training scheme based on the resilient propagation algorithm. Yee [8] presented the VGAM Package. Also Yee [9] presented a Vector generalized linear (VGLM) and additive (VGAM) models with an implementation in R program. Yee [10] has given some comments on smoothing parameter and model selection for general smooth models.
This paper can be organized as follow: Practical data & the methodology are presented in section 2. The VGAM method be used in section 3, The model with (2,1) hidden layers is devoted in section 4, The model with (5,2) hidden layers is devoted also in section 5, Results and discussion are presented in section 6, Finally the conclusions are presented in section 7.

2. Practical Data & Methodology

In this paper, we used the Hunua Ranges Data. These data were collected from the Hunua Ranges. It is a small forest in southern Auckland, New Zealand. At 392 sites in the forest, the presence/absence of 17 plant species was recorded, as well as the altitude. Each site was of area size 200m^2.

Source: Dr Neil Mitchell, University of Auckland.

Data frame contains the following columns:

17 columns presented the plant species:

Agaaus: Agathis australis, or Kauri
Beitaw: Beilschmiedia tawa, or Tawa
Corlae: Corynocarpus laevigatus
Cyadea: Cyathea dealbata
Cyamed: Cyathea medullaris
Daccup: Dacrydium cupressinum
Dacdac: Dacrycarpus dacrydioides
Eladen: Elaecarpus dentatus
Hedarb: Hedycarya arborea
Hohpop: Species name unknown
Kniexc: Knightia excelsa, or Rewarewa
Kuneri: Kunzea ericoides
Lepsco: Leptospermum scoparium
Metrob: Metrosideros robusta
Neslan: Nestegis lanceolata
Rhosap :Rhopalostylis sapida
Vitluc: Vitex lucens, or Puriri

The 18th column presented:

Altitude: Meters above sea level

In this paper, we are going to fit a simple neural network using the “neuralnet” package and fit a linear, "vglm" and nonlinear "vgam" models. Altitude is a continuous response variable, and there are 17 binary explanatory variables (presence=1,absence =0) for plant species. We are going to use 2 hidden layers with the configuration 17:5:2:1. The input layer has 17 inputs, two hidden layers have 5 and 2 neurons, and one output layer. Since we are dealing with a regression or classification problem. We are going to use MSE as a measure of how much our predictions are far away from the the real data. So, we can use a configuration 17:2:1:1 to know how much the results are changed or not.

The test set is based on the remaining objects to test the accuracy of prediction. Using "neuralnet" function to regress the dependent variable (the altitude) against the other independent variables (17 plant species). Deciding on the number of hidden layers in a neural network (NN) is not an exact result. In fact, there are instances where the accuracy will likely be higher without any hidden layers. Therefore, the train and error play a significant role in this process. One possibility is to compare how the accuracy of the predictions change as we modify the number of hidden layers. As already mentioned, our neural network has been created using the training data. Then we compare this to the test data to gauge the accuracy of the neural network forecast.

Cross validation (CV): is another very important step building predictive models. There are different kind of cross validation methods. We are going to implement a fast cross validation using

a loop for neural networks and cv.glm() function in the “boot” package for a linear model.
As far as we know, there is no built-in function in R program to perform a cross-validation on this kind of neural network. Here is the 10-fold cross-validation MSE for the linear model. So that we are splitting the data to (1:300) objects as a train set, and (301:392) objects a test set in a random way for 1 time. In each time we use the train set for fitting and use the test set for testing the predicted values. Then we ate getting the averages of the accuracy of the prediction process.

3. Fitting Data Using VGAM Technique

In this section, we are used the original data which have 17 correlated binary response variables represented the plant species, and one explanatory variable which represents the altitude. In this case we study the effect of altitude on presence/absence all species together. The process must be clear, once we combine the three correlated response variables together. At last time we used just only two response correlated variables. Since we have 17 response variables. Then we study the effect of the altitude on presence or absence the plant species. Graphically, we displayed the regression process and the probability of absence the plant.
Figures 1,3,5,7,9,11 display the fitting of plant species simultaneously three-three species. Also, Figures 2,4,6,8,10,12 display the probability of presence these species as shown below.

We used the "VGAM" package in R program to do this study, we have the next results:

1)- Fitting (Agaaus, Beitaw, Corlae) Species Simultaneously:

Residual deviance = 1066.364

Log-likelihood: = 533.182

Terms and Approximate Chi-squares for nonparametric effects

	Term	Chi-square	P-value
Predictor 1	0.8	9.2773	0.00167 **
Predictor 2	1.8	2.0974	0.31865
Predictor 3	0.3	0.1437	0.32464

Coefficients of VGAM:

	Logit (agaaus)	Logit (beitaw)	Logit (corlae)
Intercept	-1.301849	-1.037499108	-1.54376268
Predictor	0.000039	0.004109184	-0.01009208

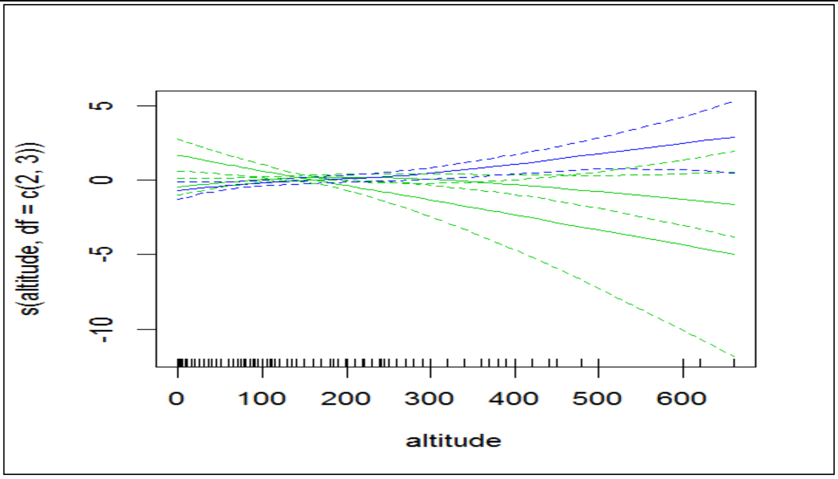


Figure 1: Fitting (agaaus, beitaw, corlae) species simultaneously

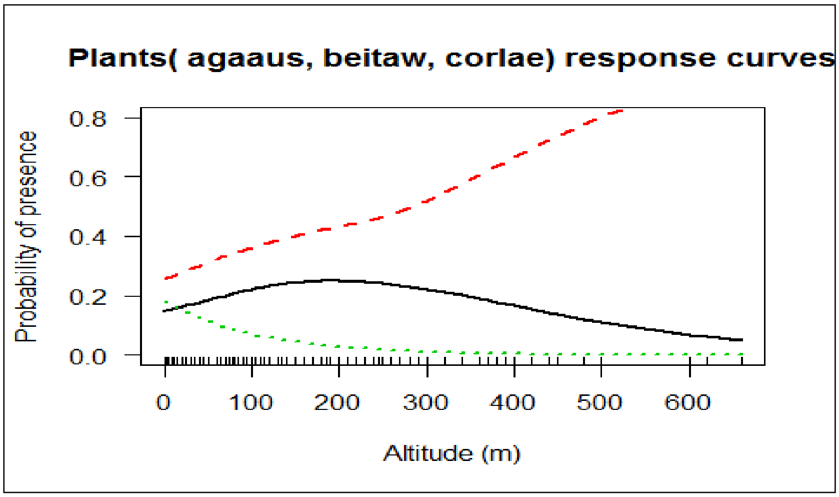


Figure 2: Probability of presence (agaaus, beitaw, corlae) species

2)- Fitting (Cyadea, Cyamed, Daccup) Species Simultaneously:

Residual deviance = 1403.888

Log-likelihood = -701.9439

Terms and Approximate Chi-squares for nonparametric effects

	Terms	Chi-square	P-value
Predictor1	0.9	6.8073	0.0079073**
Predictor2	2	9.2878	0.0092059**
Predictor3	0.9	7.3585	0.0051601**

Coefficients of VGAM:t

	Logit (cyadea)	Logit (cyamed)	Logit (daccup)
Intercept	-0.7472683034	-1.336367516	-0.6768249852
Predictor	0.0002653719	0.001995996	-0.0008349688

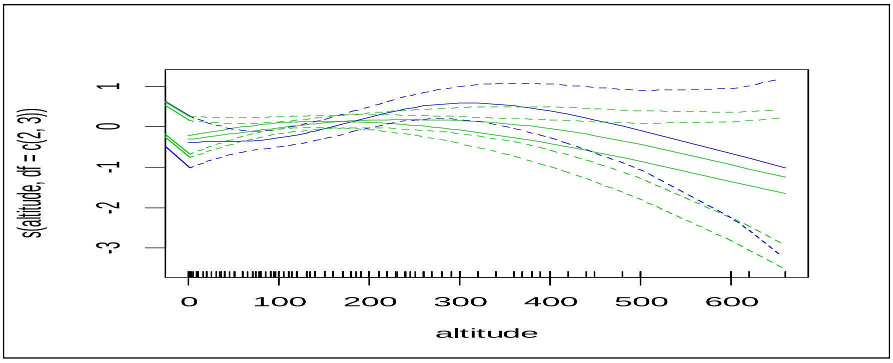


Figure 3: Fitting (cyadea, cyamed, daccup) species simultaneously

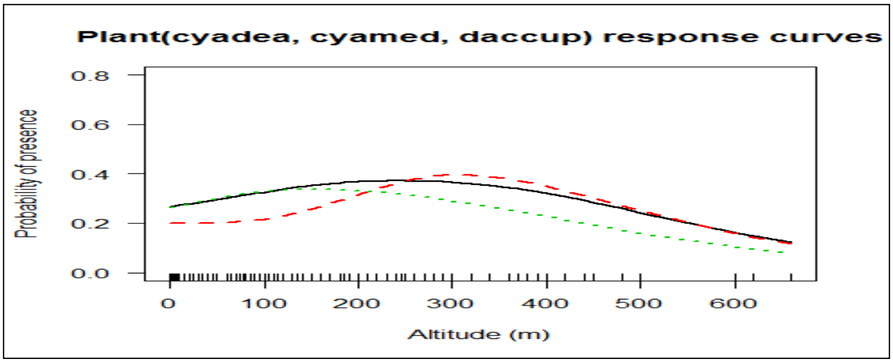


Figure 4: Probability of presence (cyadea, cyamed, daccup) species

<u>3- Fitting (Dacdac, Eladen, Hedarb) Species Simultaneously:</u>

Residual deviance = 749.1986

Log-likelihood = -374.5993

Terms and Approximate Chi-squares for nonparametric effects

	Term	Chi-square	P-value
Predictor1	0.7	1.8131	0.116981
Predictor2	2.1	4.0159	0.144761
Additive predictor3	1.1	4.5380	0.038269

Coefficients of VGAM:

	Logit (dacdac)	Logit (eladen)	Logit (hedarb)
Intercept	-1.071454227	-3.784255480	-3.009439771
Predictor	-0.003457255	0.003308942	0.006344386

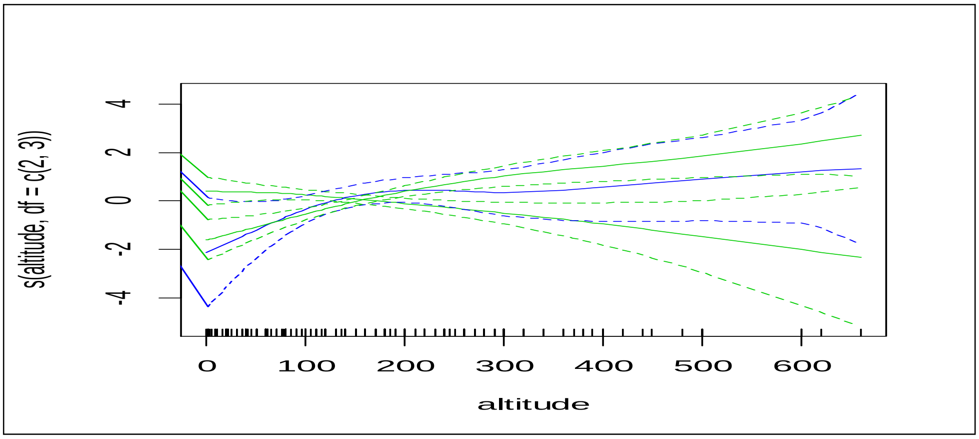


Figure 5: Fitting (dacdac, eladen, hedarb) species simultaneously

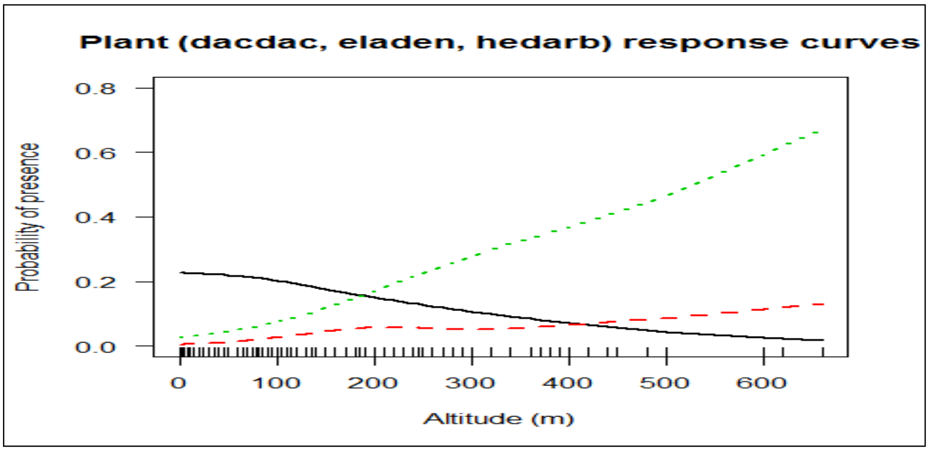


Figure ٦: Probability of presence (dacdac, eladen, hedarb) species

4- Fitting (Hohpop, Kniexc, Kuneri) Species Simultaneously:

Residual deviance = 957.154

Log-likelihood = -478.577

Terms and Approximate Chi-squares for nonparametric effects

	Term	Chi-square	P-value
Predictor 1	1		
Predictor 2	1.9	46.356	0.000 ***
Predictor 3	0.5	4.054	0.01872

Coefficients of VGAM:

	Logit (**hohpop**)	Logit (**kniexc**)	Logit (**kuneri**)
Intercept	-2.55011716	-0.072104661	0.254382868
Predictor	-0.04625166	0.002721116	-0.006016674

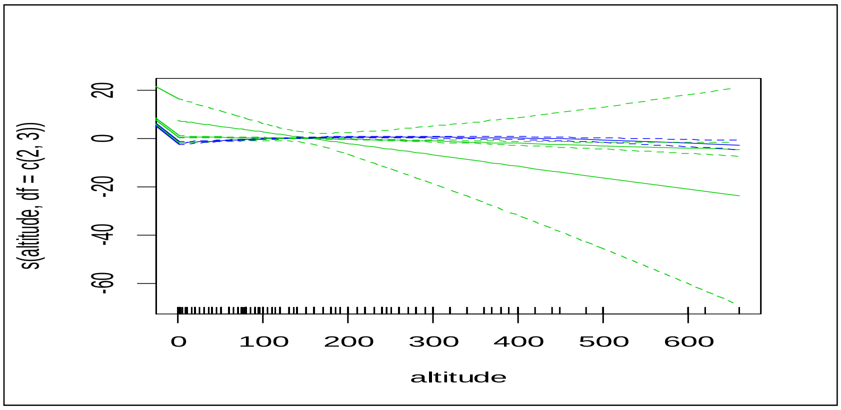


Figure 7: Fitting (hohpop, kniexc, kuneri) species simultaneously

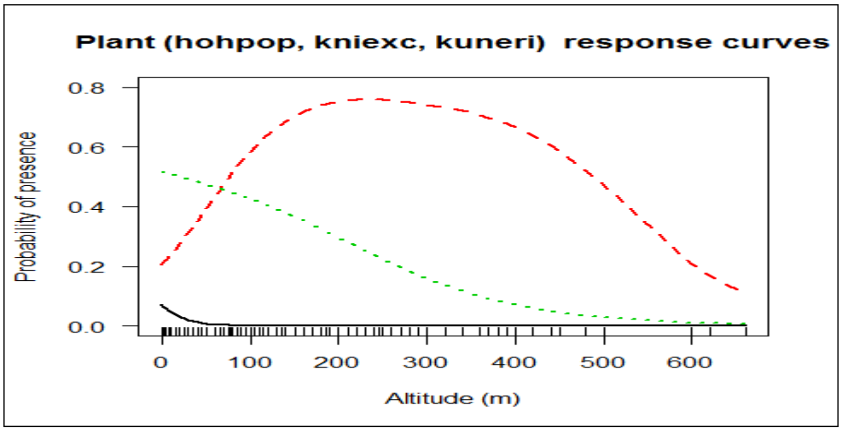


Figure 8: Probability of presence (hohpop, kniexc, kuneri) species

5)- Fitting (Lepsco, Metrob, Neslan) Species Simultaneously:

Residual deviance = 399.3723

Log-likelihood = -199.6861

Terms and Approximate Chi-squares for nonparametric effects

	Term	Chi-square	P-value
Predictor1	0.6	2.82916	0.04369
Predictor2	1.9	2.36519	0.28513
Predictor3	0.6	0.12506	0.53368

Coefficients of VGAM:

	Logit (**lepsco**)	Logit (**metrob**)	Logit (**neslan**)
Intercept	-1.930782135	-3.827899955	-2.517612863
Predictor	-0.006934513	0.002785812	-0.004700706

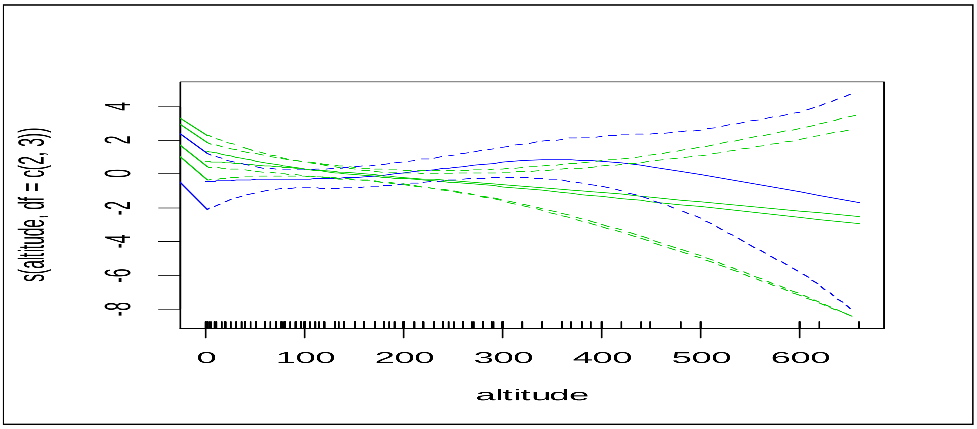


Figure 9: Fitting (lepsco, metrob, neslan) species simultaneously

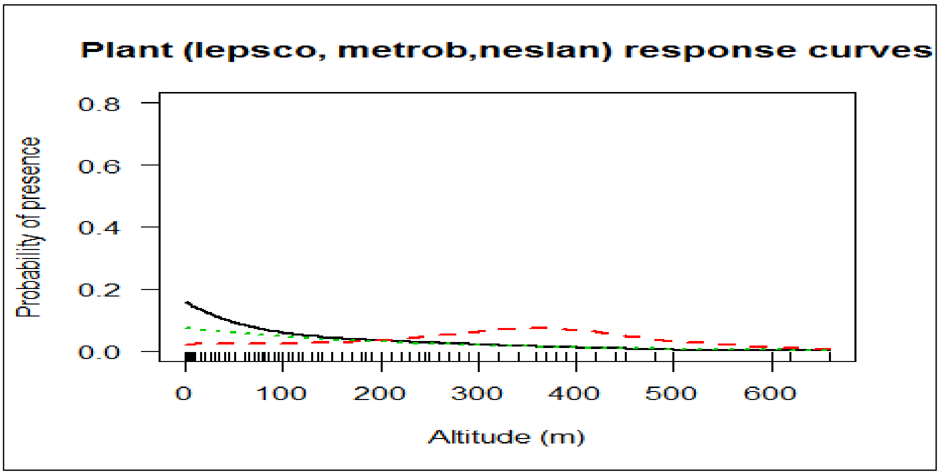


Figure 10: Probability of presence (lepsco, metrob, neslan) species

6)-Fitting (Rhosap, Vitluc) Species Simultaneously:

Residual deviance = 707.5368

Log-likelihood = -353.7684

Terms and Approximate Chi-squares for nonparametric effects

	Term	Chi-square	P-value
Predictor1	0.8	9.8448	0.001093
Predictor2	0.9	2.5233	0.097576

Coefficients of VGAM:

	Logit (**rhosap**)	Logit (**vitluc**)
Intercept	-1.355082	-0.07009608
A. predictor	0.0000167	-0.01140925

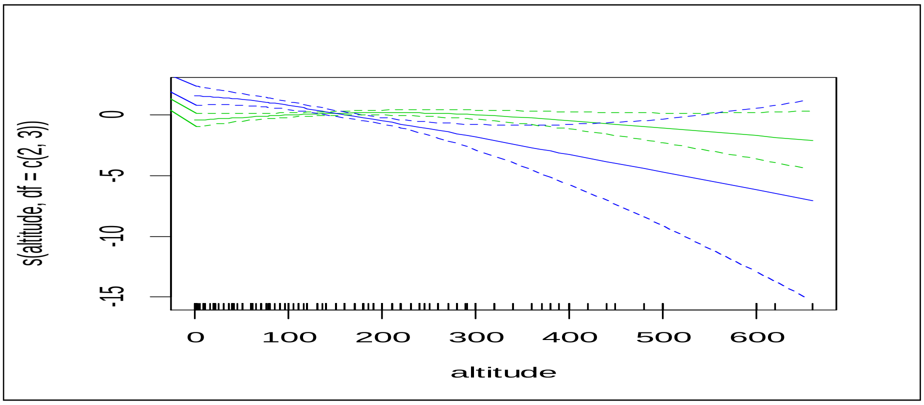


Figure 11: Fitting (rhosap, vitluc) species simultaneously

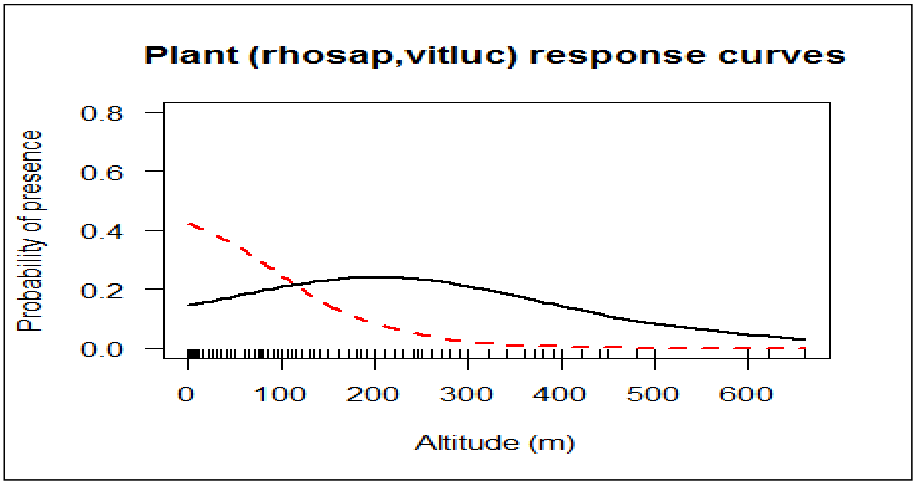


Figure 12: Probability of presence (rhosap, vitluc) species

4. The Model with (2,1) Hidden Layers

In this section, we used the altitude as the response variable, and the plant species as the explanatory variables. Before fitting the data, we will normalize the original data. Using the train set (300 objects) for fitting a linear model using GLM technique. Then we have the next results:

Coefficients	Estimate	S. Error	t value	P-value
Intercept	-0.08843	0.04446	-1.989	0.047680 *
Agaaus	0.05328	0.04838	1.101	0.271698
Beitaw	0.04737	0.05243	0.904	0.366976
Corlae	-0.05832	0.04689	-1.244	0.214585
Cyadea	-0.03705	0.04764	-0.778	0.437367
Cyamed	0.08166	0.04748	1.720	0.086562
Daccup	-0.06927	0.05049	-1.372	0.171145
Dacdac	-0.07552	0.04809	-1.570	0.117441
Eladen	0.08045	0.05694	1.413	0.158811
Hedarb	0.17067	0.04900	3.483	0.000574 ***
Hohpop	-0.05063	0.03875	-1.307	0.192432
Kniexc	0.06966	0.04787	1.455	0.146699
Kuneri	-0.23195	0.05260	-4.409	0.0000148***
Lepsco	-0.05363	0.05557	-0.965	0.335385
Metrob	0.03149	0.05007	0.629	0.529854
Neslan	-0.13392	0.04332	-3.092	0.002190 **
Rhosap	-0.05171	0.04935	-1.048	0.295695
Vitluc	-0.20880	0.04759	-4.388	0.0000162***

Null deviance = 224.27 , Residual deviance = 158.92 ,
AIC = 698.74, MSE = 1.29

Fitting Data Using “neuralnet” Package:

A neural network(NN) are explaining their outcome is more difficult than explaining the outcome of simpler model such as a linear model. Therefore, you might want to take into account this factor too, depending on the kind of application you need. Furthermore, extra care is needed to fit a neural network and small changes can lead to different results. So, we have 17 inputs, and the configuration 17:2:1:1. We now generate the error of the neural network model along with the weights between the inputs, hidden layers and the output. From our study, we have:

Error = 2.117, MSE(NN) = 37.757, Threshold = 0.00639.

The threshold is meaning that: if the change in error during an iteration is less than 1%, then no further optimization will be carried out by the model. For comparison MSE between GLM and NN: MSE (GLM) is less than MSE (NN), so GLM is the best for fitting these data.

Figure 13 explains fitting a network with (2,1) hidden layers:

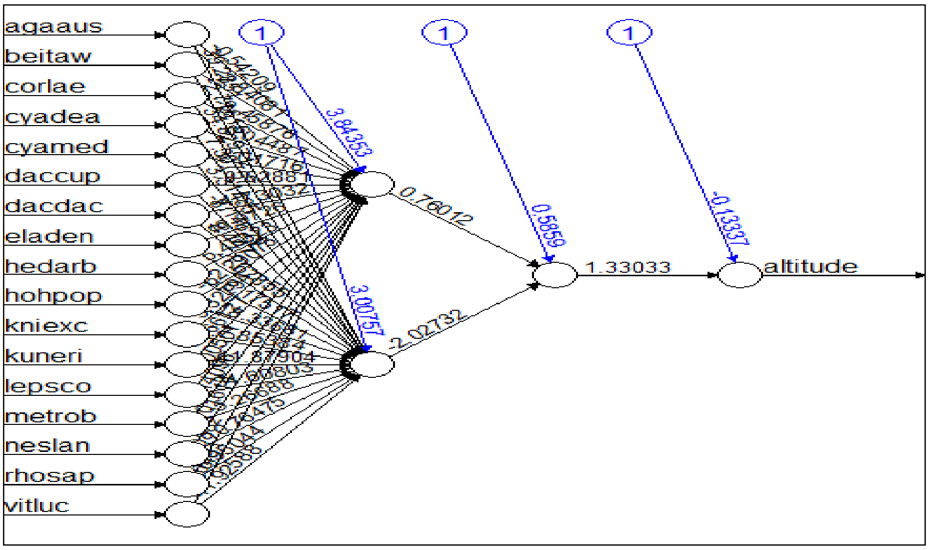


Figure 13: The network with (2,1) hidden layers.

From figure 13 we have:

Intercept to layer hidden 1:1 = 3.844, and the coefficients for the explanatory variables :

Agaaus	Beitaw	Corlae	Cyadea	Cyamed	Daccup	Dacdac	Eladen	Hedar
-0.54	-2.84	-9.46	-0.045	2.017	-0.629	-4.73	10.07	5.463

Kuneri	Lepsco	Metrob	Neslan	Rhosap	Vitluc	Hohpop	Kniexc
-5.66	-9.005	-2.141	-11.016	4.81	-15.049	-62.29	3.328

Intercept to layer hidden 1:2 = 3.00757138 , and the coefficients for the explanatory variables :

Agaaus	Beitaw	Corlae	Cyadea	Cyamed	Daccup	Dacdac	Eladen	Hedar
39.285	-2.716	34.53	7.307	3.315	-8.146	8.447	-1.679	-6.175

Kuneri	Lepsco	Metrob	Neslan	Rhosap	Vitluc	Hohpop	Kniexc
41.879	34.608	-5.267	10.765	6.95	41.926	18.337	7.854

Intercept to layer hidden 2:1 = 0.58589582

1 to l = 0.76011852

2 to 1 = -2.02731967

Intercept to altitude = -0.13336983

1 to altitude = 1.33033

Figure 14 displays the comparison between real and predicted values for GLM and NN (2,1).

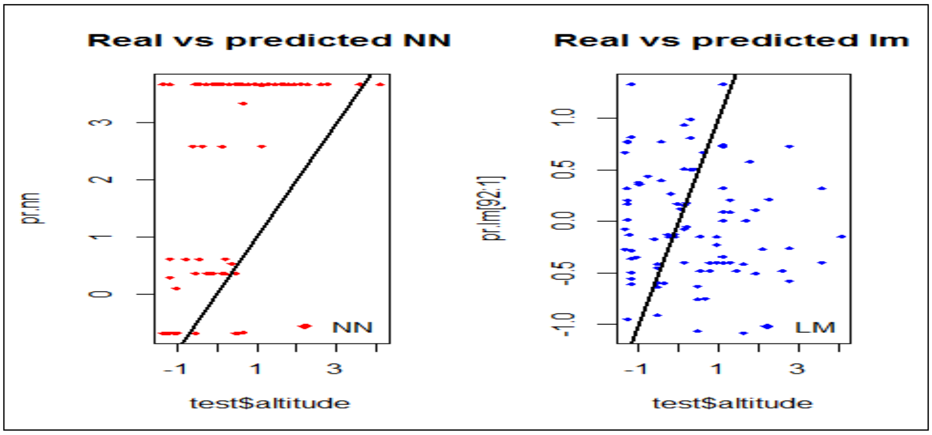


Figure 14: Predicted and real values using GLM and NN (2,1)

Figure 14 indicates the distribution of real values around the predicted values using GLM method are closed. But using neural networks is dispersed.

Classification Process:

A confusion matrix is created with a table to compare the number of true/false positives (1's) and negatives (0's). We have used the test set (92 objects) to test the accuracy of classification model using the networks model. The model generates 31 true, while there are 61 false. Ultimately, we yield 34% accuracy rate. This rate is too low. Since the accuracy rate determines whether the presence of species effect on the level of altitude or not.

The next table displayed the results of validation model. Indicates the coefficients, estimates, standard error, t-value and p-value for testing the significance of estimates.

Coefficients	Estimate	S. Error	t value	P-value
Intercept	196.847	13.974	14.087	0.00 ***
Agaaus	2.692	14.641	0.184	0.85422
Beitaw	28.252	13.052	2.165	0.03106 *
Corlae	-43.138	24.073	-1.792	0.07395
Cyadea	-19.658	12.032	-1.634	0.10314
Cyamed	7.572	12.816	0.591	0.55503
Daccup	-14.593	13.143	-1.110	0.26757
Dacdac	-24.506	15.537	-1.577	0.11558
Eladen	22.660	27.851	0.814	0.41638
Hedarb	70.736	16.412	4.310	0.000092 ***

Hohpop	-107.091	73.528	-1.456	0.14610
Kniexc	14.519	11.484	1.264	0.20691
Kuneri	-62.262	13.022	-4.781	0.00000251***
Lepsco	-73.704	23.877	-3.087	0.00217 **
Metrob	24.505	29.497	0.831	0.40664
Neslan	-85.980	26.884	-3.198	0.00150 **
Rhosap	-23.115	15.328	-1.508	0.13239
Vitluc	-92.042	14.940	-6.161	0.0 ***

Null deviance = 5802056 , Residual deviance = 3908552

AIC = 4759.8

Figure 16 displays the neural network corss valodation method with (2,1) hidden layers.

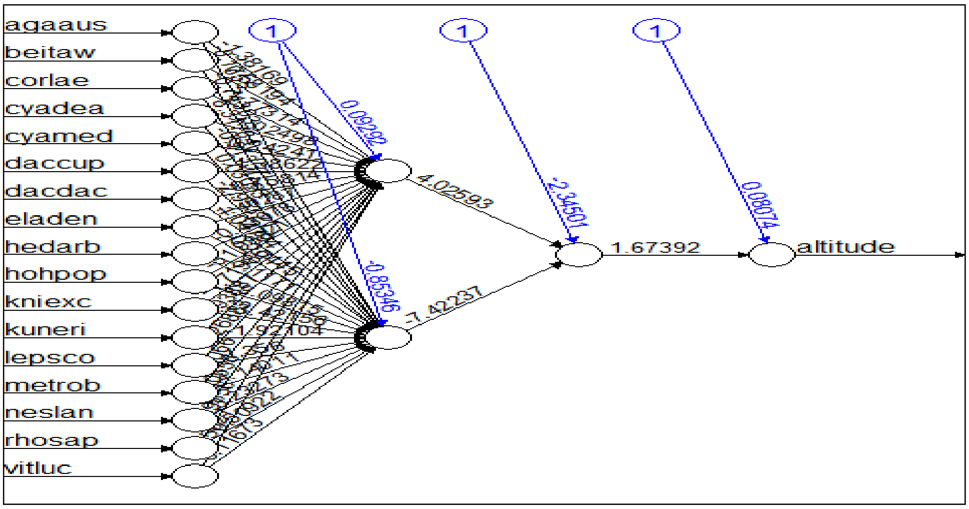


Figure 15: Fitting Neural Networks CV with (2,1) hidden layers

We calculate the average MSE and plot the results as a boxplot in figure 16 as shown below:

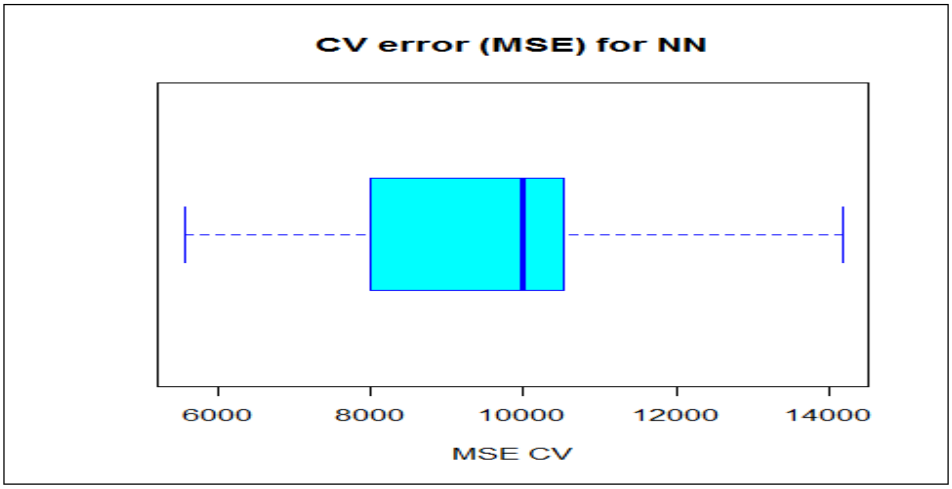


Figure 16: Boxplot of MSE Neural networks CV with (2,1) hidden layers

MSE (GLM) = 9857.305, MSE CV (NN) = 10785.44

As we see, the average MSE CV for the neural network (10785.44) is more than MSE of a linear model (9857.305) although there seems to be a certain degree of variation in MSEs of the cross validation. This may depend on the splitting of the data or the random initialization of the weights in the net.

5. The Model With (5,2) Hidden Layers

What is happened if we use (5,2) hidden layers in the neural network model? Is MSE value changes? From the R results we have:

Error = 1.45 , MSE = 42.215 and Threshold = 0.0١97.

Figure 17 indicates the distribution of real values around the predicted values using the GLM method are closed. But using the neural networks is far away.

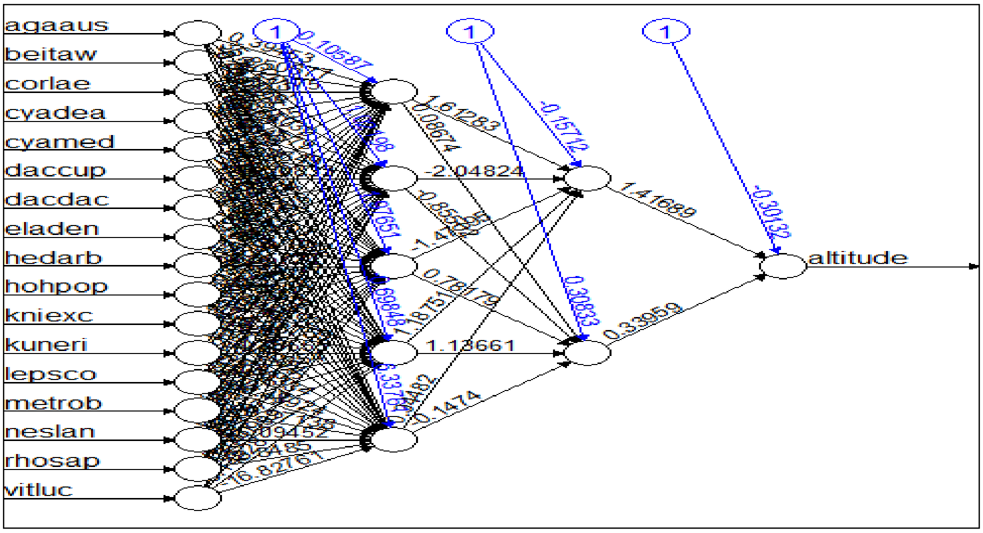


Figure 17: Fitting Neural Networks CV with (5,2) hidden layers

From Figure 17, we have the next results:

Intercept to layer hidden 1:1= 0.10587, and the coefficients for the explanatory variables :

Agaaus	Beitaw	Corlae	Cyadea	Cyamed	Daccup	Dacdac	Eladen	Hedar
0.39	-2.507	1.172	0.73	1.654	0.3	-1.007	-1.57	2.849

Kuneri	Lepsco	Metrob	Neslan	Rhosap	Vitluc	Hohpop	Kniexc
-0.878	1.64	-0.886	-1.215	0.186	-0.527	-0.459	-1.113

Intercept to layer hidden 1:2 =1.021978, and the coefficients for the explanatory variables :

Agaaus	Beitaw	Corlae	Cyadea	Cyamed	Daccup	Dacdac	Eladen	Hedar
35.80205	3.607	0.357	3.046	0.346	0.362	6.06	-9.88	-4.488

Kuneri	Lepsco	Metrob	Neslan	Rhosap	Vitluc	Hohpop	Kniexc
1.50	6.88	4.561	1.42	3.284	-0.64	5.7	3.57

Intercept to layer hidden 1:3 = 1.976509, and the coefficients for the explanatory variables :

Agaaus	Beitaw	Corlae	Cyadea	Cyamed	Daccup	Dacdac	Eladen	Hedar
-5.78	-7.58	6.301	-4.38	-0.73	4.77	9.157	-0.29	5.657

Kuneri	Lepsco	Metrob	Neslan	Rhosap	Vitluc	Hohpop	Kniexc
-0.18	-2.85	-0.1218	1.117	2.29	1.274	-8.94	1.012

Intercept to layer hidden 1:4 = -0.6984803, and the coefficients for the explanatory variables :

Agaaus	Beitaw	Corlae	Cyadea	Cyamed	Daccup	Dacdac	Eladen	Hedar
-6.82	1.73	-2.4	-2.26	-5.05	2.886	6.348	-3.05	-0.205

Kuneri	Lepsco	Metrob	Neslan	Rhosap	Vitluc	Hohpop	Kniexc
-0.429	-2.268	-4.76	-2.54	5.78	4.767	-2.11	6.148

Intercept to layer hidden 1:5 = 6.337606 , and the coefficients for the explanatory variables :

Agaaus	Beitaw	Corlae	Cyadea	Cyamed	Daccup	Dacdac	Eladen	Hedar
-0.936	14.66	-214.245	-5.067	-8.234	-4.03	-10.73	15.23	-15.1

Kuneri	Lepsco	Metrob	Neslan	Rhosap	Vitluc	Hohpop	Kniexc
-1.44	-9.06	-5.88	66.49	20.37	-6.09	-2.85	-16.83

Intercept to 1 = -0.157

1 to 1 = 1.613

2 to 1 = -2.048

3 to 1 = -1.47

4 to 1 = 1.188

5 to 1 = 0.945

Intercept to 2 = 0.3083269

1.to.2 = 0.087

2.to.2 = -0.855

3.to.2 = 0.782

4.to.2 = 1.137

5.to.2 = -0.147

Intercept to altitude = -0.3013

1 to altitude = 1.417

2 to altitude = 0.34

Figure 18 displays the comparison between real and predicted values for GLM and NN (5,2).

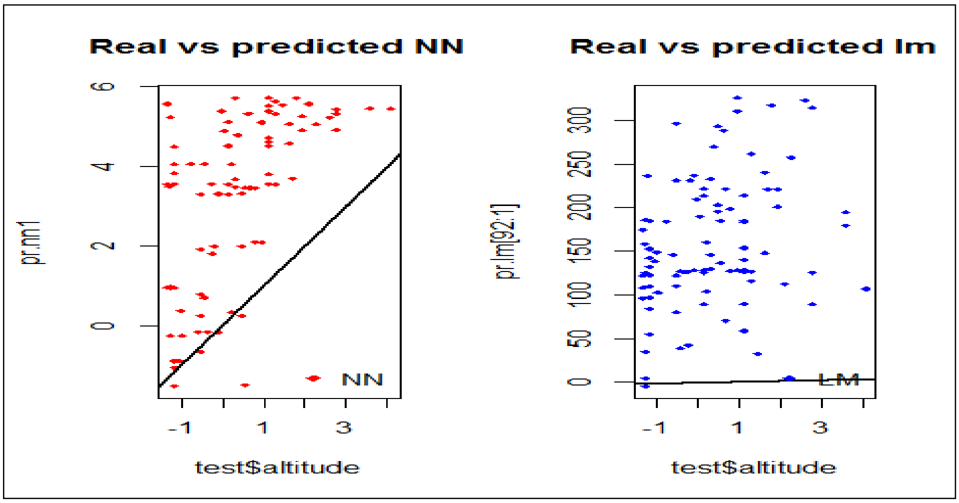


Figure 18: Predicted and real values for GLM and NN with (5,2) hidden layers

The model generates 27 true and 65 false. The accuracy rate is 29% is yielded (27/92). This rate determines whether the presence of species effects on the altitude or not. We see that the classification process with (2,1) hidden layers is better than the process with (5,2) hidden layers, since the accuracy rate in the second is small.

Figure 19: displays Cross validation model with (5,2) hidden layers:

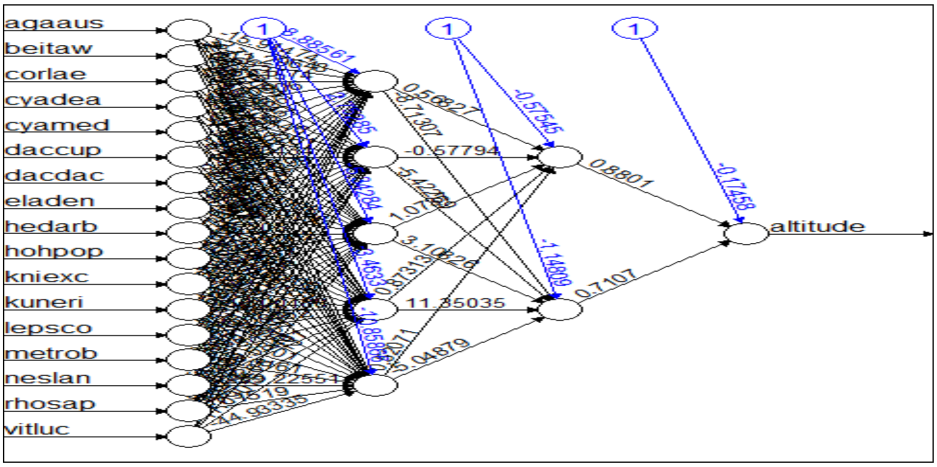


Figure 19:Fitting Neural Networks CV with (5,2) hidden layers

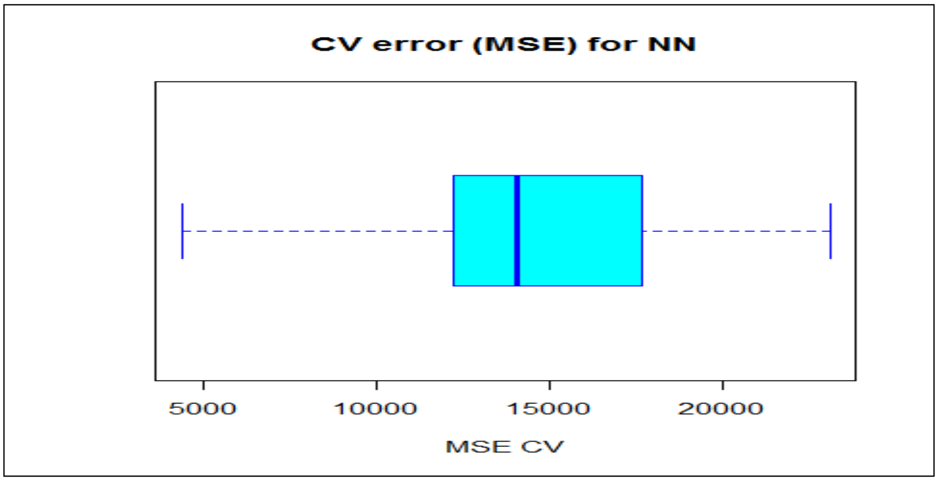


Figure 20: MSE of CV for Neural networks with (5,2) hidden layers

MSE (GLM) = 10684.44, MSE CV error (NN) = 14278.69.

The average of MSE CV error of the neural networks (14278.69) is more than MSE of linear model (10684.44). The GLM is the best for fitting these data. Also, we can see that the MSEs are increased when increase the hidden layers for either GLM or NN.

6. Results and Discussion

From the previous results, we have next comments:

1)-For VGAM model: some of plant species are affected significantly by the altitude, and another plants does not affected by the altitude significantly.

2)-For comparison between GLM models, we have:

Model	Null deviance	Residual deviance	AIC
GLM	224.27	158.92	698.74
GLM.CV	5802056	3908552	4759.8

Untraditionally, the results indicate that the GLM model without CV is better for these data.

3)-For comparison between models, we have:

	MSE	Threshold	Error
GLM	1.29	none	none
NN with (2,1) hidden layers	37.757	0.00639	2.117
NN with (5,2) hidden layers	42.215	0.01970	1.45

The MSE(GLM) is less than MSE(NN) with (2,1), (5,2) hidden layers. But MSE(NN) is increased with (5,2) hidden layers. The threshold is increased with (5,2) hidden layers. Finally, the error of classification is decreased with (5,2) hidden layers.

4)-For comparison between CV models, we have:

	MSE.CV(GLM)	MSE.CV(NN)
With (2,1) hidden layers	9857.305	10785.44
With (5,2) hidden layers	10684.44	14278.69

The MSE.CV(GLM) and MSE.CV(NN) are increased when moving to (5,2) hidden layers. But MSE.CV(GLM) is less than MSE.CV(NN) for both cases. So, GLM with CV is the best for these data.

5)-For comparison between (2,1) and (5,2) for Classification

	False	True	Total	Accuracy rate
With (2,1) hidden layers	61	31	92	34%
With (5,2) hidden layers	65	27	92	29%

The accuracy rate with (2,1) in more than the accuracy rate with (5,2) hidden layers. So, the configuration 17:2:1:1 is the best for these data.

7. Conclusions

In this paper, we have used the Hunua Ranges Data as environmental data. These data were collected from the Hunua Ranges, that it is a small forest in southern Auckland, New Zealand. Some of algorithms were applied on these data to determine how much these algorithms are better than another one for fitting and classifying these data. So, we used the VGAM measure for fitting these data, have considered the binary variables (17 plant species) as response correlated variables, and the altitude (the meters above the sea) is a explanatory variable, to construct the effect of altitude on these plant species simultaneously. This is done using three-three plant species together. Also, we fitted the data using VGLM measure as a linear model comparing with fitting data using the neural networks (NN). We concluded that the best model for these data was GLM model. This is because of the MSE for GLM is smaller, whether using cross validation or not.

For the classification and regression processes, we used two configurations to compare between them and conduct that the configuration (17:2:1:1) is the best for these data, because of the accuracy rate is better. In brief, for the regression process, the GLM model is better than the Neural network (NN) for the MSE measure. Also, the NN model is better with (2,1) hidden layers. Finally, for the classification process using the NN model, with the configuration (17:2:1:1), is better than with the configuration (17:5:2:1), since the accuracy rate is small in the second one.

References

1- Riedmiller M. and Braun H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. Proceedings of the IEEE International Conference on Neural Networks (ICNN), p. 586-591. San Francisco.

2- Intrator O. and Intrator N. (1993). Using Neural Nets for Interpretation of Nonlinear Models. Proceedings of the Statistical Computing Section, p. 244-249 San Francisco: American Statistical Society (eds).

3- Bishop, C.M. (1995). Neural Networks for Pattern Recognition, Oxford: Oxford University Press.

4- Ripley, Brian D. (1996) Pattern Recognition and Neural Networks, Cambridge

5- Yee, T. W. and Wild, C. J. (1996). Vector generalized additive models. Journal of the Royal Statistical Society, Series B, Methodological, **58**, p. 481–493.

6- Wood, S. N. (2004). Stable and efficient multiple smoothing parameter estimation for generalized additive models. J. Amer. Statist. Assoc., **99**(467), p. 673–686.

7- Anastasiadis A. et. al. (2005). New globally convergent training scheme based on the resilient propagation algorithm. Neurocomputing 64, p. 253-270.

8- Yee, T. W. (2008). The VGAM Package. *R News*, **8**, p. 28–39.

9- Yee, T. W. (2015). Vector Generalized Linear and Additive Models: With an Implementation in R. New York, USA: *Springer*.

10- Yee, T. W. (2016). Comments on “Smoothing parameter and model selection for general smooth models” by Wood, S. N. and Pya, N. and Safken, N., *J. Amer. Statist. Assoc.*, **110**(516).