Using Bayesian Regression Neural Networks Model to Predict Thrombosis for Covid-19 Patients

By

Hindreen Abdullah Taher
Department of Information Technology College of Commerce Sulaimani University, Kurdistan
Email: hindreen.taher@univsul.edu.iq

Nawzad Muhammed Ahmed
Department of Statistics and Informatics College of Administration and Economics Sulaimani University, Kurdistan

Abstract

As it is clear that corona-virus (COVID-19) has a direct danger on the humanity in the world, at the beginning of appearing this virus till seven months later that medicine science was unable to understand the behavior of this virus, in the study we focused on the Thrombosis for 73 patients that have covid-19 as a response variable and Age, Sodium, Blood pressure as factors. We aimed to study the effect of these factors on the thrombosis for covid-19 patients, in this situation for such a response of this type Bayesian regression neural network model to predict thrombosis of the patients that have covid-19, the contribution in this study is using Bayesian regression neural network model to predict the thrombosis for the first time. Age, Sodium and Blood Pressure are capable of explaining 84% of Thrombosis. The Mse is 0.1824136 and the AIC= 14.6867 with R2 = 0.84 which is mean that age, sodium and blood pressure are able to explain 84%.

Keywords: Covid-19, Bayesian inference, Regression Neural Networks, Thrombosis.

Introduction

The World Health Organization (WHO) has dubbed this disease coronavirus disease 2019 (COVID-19). Angiotensin-converting enzyme 2 (ACE-2) is mostly located in alveolar epithelium and endothelium, which has been demonstrated to be a binding site for coronaviruses. Thrombosis, an increasingly common consequence, is now understood to be caused mostly by activation of endothelial cells. It has been found that viral inclusion bodies can be found in a wide range of organs, from the lungs to the digestive tract [1]. Pro-inflammatory "pyroptosis" in macrophages may be responsible for the immunological dysregulation that characterizes severe COVID-19 infection, with fast viral multiplication leading to enormous inflammatory mediator release. A higher D-dimer concentration is one of the most reliable indicators. There are a number of inflammatory events that can impact D-dimer levels, however in patients with COVID-19 this almost certainly indicates intravascular embolism [3, 4]. Studies in China have found a link between an elevated D-dimer level (>1000 ngmL1) at admission and an increased risk of death while in the hospital [5]. Because of the lack of rigorous and extensive examination techniques, the real prevalence of thrombosis in patients with COVID-19 infection is unknown. We want to measure the impact of Age, Sodium, Blood pressure on Thrombosis for those patients that suffer from covid-19. The purpose behind using Bayesian regression neural networks model is to get answers for the questions that are said does each of age, sodium, and blood pressure has an effect on thrombosis and how much the amount of their effect.

Related Work

By training neural networks to forecast the parameters from artificially created data, they investigated automating the process of creating summary statistics. Summary statistics are roughly posterior means of the parameters. The method creates summary statistics with moving-average modeling and model with little model-specific adjustment that are as accurate as or more accurate than justified summary statistics[1]. For flexible Bayesian regression, spline-based methods are straightforward to compute and explain. Using prior distributions that penalize complexity, we may simulate smooth bivariate interactions in an economical and seemingly new method. Choosing a model or averaging several models can be used to make predictions. In terms of computation, interpretation and predictive performance, the technique appears to perform well when applied to simulated and real datasets[2]. To get around this problem, he looks at Gaussian approximations, Markov chain Monte Carlo simulation routines, and a group of non-Gaussian but "deterministic" approximations known as variation approximations. Approximation theories such as Laplace's approximation and MCMC have been used in Bayesian work on neural networks and related models in the past. However, new approximation ideas such as variational approaches have also been developed.. As a result, statisticians are strongly urged to investigate the relevant journals and conference proceedings in computer science, as well as to play an active role in the development of the field of computer science [3].

Different economic traits were used in the development of a prediction equation for lifetime milk yield (LTMY). In India, 1210 Holstein Friesian crossbred dairy cattle were analyzed for their first lactation length, first peak yield, first lactation total milk yield, and three lactation total milk yield. The multiple linear regression model was used to compare feed-forward back propagation algorithm variants. In terms of LTMY prediction, the BR algorithm outperformed the other algorithms by a wide margin. The milk yield could be predicted by the BR neural network model with an R^2 of 71.18 [4].

Materials and Methiods

Multiple Linear Regression Model

Linear regression is the most statistical model used in practical applications because these types of models are linearly dependent on their unknown parameters. This can be fitted much more easily than the other models which response have a non-linear relationship with their unknown parameters and because the properties of statistical estimators are easier to explain [23].

The linear model of single regression can be shown as:

$Y_i = B_0 + B_1 X_{i1} + B_2 X_{i2} + \dots + B_{p-1} X_{ip-1} + \varepsilon i$ (1) Where i = 0, 1, 2, n.

$\underline{Y}$: is the vector of response variable that distributed normally of n*1 dimensions.

$\underline{X}$: is matrix of explanatory variables of n*p-1 dimensions.

$\underline{B}$: is vector of unknown parameters of p-1*1 dimensions.

$\underline{\varepsilon}$: is vector of residual of n*1 dimensions.

Neural Networks

In order to understand NNs from a Bayesian perspective, it is necessary to review the basics of neural computation and to establish the notation that will be used throughout the section. The Multi-Layer Perceptron (MLP) network will be the primary focused [8].

Convolutional networks, like MLPs, are based on the MLP, which serves as the foundation for NNs. The network's output can be represented as follows for this network's input x of dimension N_1 [9].

$$\phi_j = \sum_{i=1}^{N_1} a(x_i\, w_{ij}^1), \ (2)$$
$$f_k = \sum_{j=1}^{N_2} g(\phi_j w_{jk}^2) \ (3)$$

Overlapping neural connections are denoted by the superscripted number after w in the parameters. The output of the hidden layer, which has dimensions N_2, is depicted in Equation (2). After the N_2 hidden layer, the k_{th} output is a summing of all N_2 outputs. There are numerous hidden layers that may be added to this model [11]. To keep things simple, this section does not provide a bias value for each layer. Each buried layer neuron (or node) is referred to in Equation (2)[10]. This is commonly referred to as an activation and it is stated a transformation followed by a non-linear element wise transformation $\phi(.)$. The sign(.) function was originally used as the activation function for perceptron’s, but this function is no longer used because its derivative is equal to zero[5]. In addition to the Sigmoid and Hyperbolic Tangent (TanH), the Rectified Linear Unit (ReLU) and the Leaky-ReLU are more advantageous activation functions.

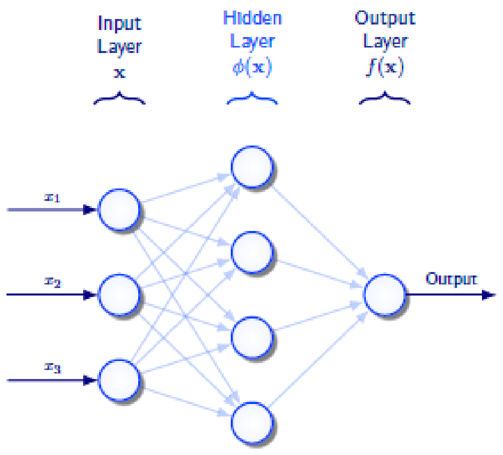


Figure (1) *shows the structure of a neural networks.*

For binary classification or 1-D regression, a 1-layer neural network (NN) design is shown in Figure 1. Input states are summarized and activated at each node, which symbolizes a neuron or a state.

A neuron's strength of connection is indicated by the weight of the arrows. derivatives that are able to reply. Explanation one of the Sigmoid function means that the network's output is comparable to a sum of logistic regression models. The identity function will be applied to the output g(.) in a regression model, and a Sigmoid will be used for binary classification.

A matrix representation is a convenient tool for quickly and accurately implementing equations 2 and 3. The input vector is stacked in the data set as a column in X to obtain the result. The following is an example of how to execute forward propagation:

$$\Phi = a(X^T W^1) \ (4)$$
$$F = g(\Phi W^2) \ (5)$$

In spite of the fact that this matrix notation is more compact, the decision to adopt summation notation to express the network was not made lightly. In the aim that the summation notation would make the connections between kernel and statistical theory, J(x,y) is a non-convex cost function for NN learning in the frequents context, and it is minimized by the network weights to get the MLE or MAP estimate for a given set of values. After a model's output has been computed for the current parameter values, a partial derivative w.r.t. the parameters is obtained, and these partial derivatives are used to update each parameter in order to minimize the cost function's value.

$$w_{t+i} = w_t - \alpha \frac{\partial J(x,y)}{\partial wt} \quad (6)$$

Equation 6 shows how back-propagation changes model parameters, with the learning rate and the subscripts indicating the training iteration. Using a chain rule, we may get partial derivatives for distinct parameters at different layers of the network. Thus, discontinuous non-linearity like the ReLU is preferred for deep training NNs because its greater gradient helps prevent early layer gradients from disappearing during training.

Bayesian Neural Networks

For example, we suppose that weights have a genuine value that is only unknown and the data we've seen is regarded as a random variable in our frequenters setup above. This may appear to be at odds with our goals. Based on the information we have, we would like to find out what the weights of the unknown models. For statistical modeling, we have access to data in the form of the collected information. Because we have no idea what the initial weights will be, it seems sense to treat them like a random variable[6]. In a Bayesian statistical approach, the unidentified (or latent) parameters are viewed as random variables, and our goal is to infer a distribution of these parameters based on the training data. Unknown model weights are inferred from what we already know or can observe during the "learning" phase of BNNs. The Bayes Theorem is used to tackle the problem of inverse probability. We can't see the real distribution of the model's weights because they're latent variables [7]. For example, Bayes' Theorem allows us to define a distribution over these weights in terms observable probabilities, which results in a distribution of model parameters conditional on the data we have observed p(w|D). We can see the joint distribution p(w/D) between the weights and the data before training. According to the previous beliefs of p(w) and the choice of model likelihood p(D|w), this joint distribution is characterized.

$$p(w, \mathcal{D}) = p(\mathcal{D} \mid w) \ (7)$$

The probability term in Equation 7 is determined by the network design and loss function. It is reasonable to assume that, for a 1-D homoscedastic regression issue, the likelihood is distributed normally, with $p(w, \mathcal{D}) = \mathcal{N}(f^w(\mathcal{D}), \sigma^2)$, as the mean value indicated by network output. All samples from D are considered to be identical in this model, which means that the probability may be expressed as a product of the contributions from each of the (n) distinct terms in the data set; however, this assumption is not always true.

$$p(w, \mathcal{D}) = \prod_{i=1}^{N} \mathcal{N}(f^w(x_i), \sigma^2) \ (8)$$

Previous to receiving any data, the distribution of weights should be described in the prior distribution. It is difficult to provide a relevant prior for NNs because of their black-box nature. Under the frequenters technique, many practical NNs are trained with low-magnitude weights centered around zero. A zero-mean Gaussian with a small variance, or a spike-slab prior with a zero center, can be used as a prior in the model to increase sparsity. The posterior distribution of the model weights is obtained by applying Bayes theory to the prior and likelihood distributions.

$$\pi(\omega \mid \mathcal{D}) = \frac{p(\omega)p(\mathcal{D} \mid \omega)}{\int p(\omega)p(\mathcal{D} \mid \omega)d\omega} = \frac{p(\omega)p(\mathcal{D} \mid \omega)}{p(\mathcal{D})} \ (9)$$

The evidence or marginal likelihood is the denominator of the posterior distribution. The posterior distribution using normalized value, which is independent of the model's weights. We can make predictions about any quantity of interest based on the posterior distribution. According to the posterior distribution, we may make predictions.

$$E_{\pi}[f] = \int (\omega)\pi(\omega \mid Ɗ)d\omega \ (10)$$

There will be an expectation of this type for all predicted variables of interest. The predictive quantity is an expectation over the posterior, regardless of whether it is a predictive mean, variance, or interval. Only the function f(w) with expectation applied will be different. A weighted average of function f(w) may then be used to get the predict (w). Here, we see how marginalization (integration) across uncertain model weights drives Bayesian inference. We may learn about a model's generating process rather than an optimization method utilized in the frequents environment by employing this marginalization technique. These correct conditional probabilities may be accessed using this generative model. Many parameters, such as the noise variance for any previous values, were presumptively known in this description. Usually, this isn't the case, thus we have to deduce these unknown factors for ourselves. For these extra variables, we use the Bayesian framework to infer them as latent variables, give a prior distribution (or occasionally a hyper-prior) and then marginalize over them to arrive at the posterior, just as we do for the weights. Detailed instructions on how this may be applied to BNNs are available in [12,13]. The computation of the posterior (Equation 8) remains problematic for many models of interest. This is partly because of how the marginal likelihood was figured out. For non-conjugate models or those with non-linear latent variables (like NNs), this quantity can be computationally intractable. A quadrature approximation of this integral can be computationally challenging in high-dimensional models. As a result, it is necessary to approximate the posterior. The next sections go into depth on how BNNs can approximate Bayesian inference.

Origin of Bayesian Neural Networks

Based on the results of this and previous assessments [14], the first instance of what may be deemed a BNN [15] was built. The statistical interpretation of the loss functions utilized in this research emphasizes the statistical features of NNs. The MLE of a Gaussian may be found by minimizing a squared error term, as it was previously demonstrated. Bayes Theorem may be used to find an acceptable posterior by giving a prior over the weights of the network. This paper gives important insights into the Bayesian perspective of NNs, but no method for obtaining the marginal probability (evidence) is provided, which means that no practical methods of inference is proposed. Using the Laplace approximation, Denker and LeCun give a feasible method for executing approximate inference, but only provide basic experimental data [16]. A NN is a function that approximates a wide range of functions. In a single hidden layer network, every function may be represented as the number of parameters approaches infinite [16,17,18]. For the practical example, this means that as long as the model has enough trainable parameters, it can accurately imitative the finite training data set. The number of parameters in the NN or the degree of polynomial used increases the model complexity, leading to overfitting difficulties, even though it is possible to represent any function and even match the training data with a NN. An important question in neural network (NN) modeling is, "How complicated should I construct my model?" [20] MacKay shows how a Bayesian framework easily lends itself to the process of model construction and comparison of generic statistical models [19]. There are two types of inference discussed here: inference used to fit a model and inference used to judge a model's appropriateness. For the initial level of inference, Bayes' rule is commonly used to update the model parameters.

$$P(\omega \mid Ɗ, H) = \frac{P(Ɗ|\omega, H_i)P(\omega|H_i)}{P(Ɗ|H_i)}, \ (11)$$

Where w is the set of parameters in the generic statistical model, D is the data and Hi represents the ith model used for this level of inference [7]. This is then described as,

$$Posterior = \frac{Likelihood \times Prior}{Evidence}$$

Important to remember is that the normalizing constant in Equation 11 refers to data supporting a certain model as evidence hi As a result, approximations must be used when attempting to evaluate the posterior for most relevant models. Laplace approximation is employed in this study. Although it is necessary to compute the posterior over parameters, the primary goal of this study is to present techniques for evaluating the posterior over the model hypothesis. The design of the posterior over model is depicted as

$$P(H_i|\mathcal{D}) \propto P(\mathcal{D}|H_i)P(H_i) \quad (12)$$

Model posterior equal to evidence multiplied by x, which is equivalent to Prior to the model, the data dependent component in Equation 12 functions as evidence for it. Most BNNs are unable to evaluate the posterior normalization constant, despite its positive interpretation as discussed before. The evidence's Laplace approximation, assuming a Gaussian distribution, is,

$$P(\mathcal{D}|H_i) = \int P(\mathcal{D}|\omega, H_i)P(\omega|H_i)d\omega \quad (13)$$
$$\approx P(\mathcal{D}|\omega_{MAP}, H_i)[P(\omega_{MAP}|H_i)\Delta\omega] \quad (14)$$
$$= P(\mathcal{D}|\omega_{MAP}, H_i)\left[P(\omega_{MAP}|H_i)(2\pi)^{\frac{k}{2}}\det^{-\frac{1}{2}}A\right] \quad (15)$$
$$= \text{Best Likelihood Fit } \times \text{Occam Factor}$$

As a single Riemann approximation to the model evidence, the peak of the evidence is represented by the highest likelihood fit, and Occam's factor is the width that is defined by the curvature around this peak. It is possible to calculate the Occam factor by dividing the posterior w range by the prior w_0 range for the given model H_i.

$$\text{Occam Factor} = \frac{\Delta\omega}{\Delta\omega_0} \quad (16)$$

As a result, the Occam factor is defined as the ratio of change in plausible parameter space between the prior and posterior. Complex models capable of representing vast ranges of data will have greater Occam factors using this approach. A smaller set of data may be described with better accuracy, resulting in a lower Occam Factor, even while a simple model is less capable of capturing a complicated generative process. As a result, a model's complexity is naturally regularized. Models with excessive complexity have a tendency to have a broad posterior distribution, which leads to a high Occam factor and weak support for the model under consideration. An Occam factor decreases with a wider or less informative prior, which provides more insight into the Bayesian context of regularization.

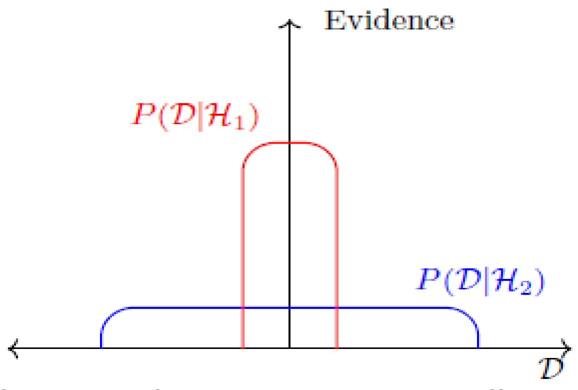

Evidence
$P(\mathcal{D}|\mathcal{H}_1)$
$P(\mathcal{D}|\mathcal{H}_2)$
$\mathcal{D}$

Figure (2) *clarify the role of evidence in producing different model hypotheses.*

The evidence plays a significant role in analyzing various model hypotheses, as seen in this graphic. It's easier to anticipate a narrow range of data using the basic model H_1, but it has a lower likelihood of success than the more sophisticated model H_2. According to [20,21]. This evidence paradigm necessitates the costly (and crucial) task of calculating the marginal likelihood. Many designs may not be viable to compare because of their high investment in estimating the marginal likelihood. A methodology for assessing solutions for BNNs, however, can be applied. Objective functions for most NN designs of relevance are non-convex and have several local minimums. The inference issue can be solved by any of the local minima. The evidence function for each local minimum is used by MacKay to compare the solutions [22]. There are no prohibitive computing requirements, therefore it is possible to evaluate the model complexity at each solution.

Results and Discussion

Data Description

The data were collected by through interviewing and monitoring 73 patients by the researchers from Shahid Aso Hospital at Sulemani governorate, the dataset contains the variables which are Thrombosis as response variable and Age, Sodium and Blood pressure as explanatories variable.

Application Steps of BRNN Model

In order to designate the best architecture for an (BRNN) model that gives fits suitable model for the data under study, the following steps have been applied:

First step

After preparing the data set it has been read in R-Language software and then normalized it to be ready for applying BRNN model on it.

Second step

We prepared the program by installing the brnn package into the R software and then partitioning the data set into %70 for training and %15 for each of testing and validation.

Third step

After running the written program with seventy epochs for training the model, the suitable network was [3 – 3 – 1] the below table shows that the best performance occurred at epoch70 for training dataset:

Table(1) *represents the occurring of best performance of the model*

Epoch	E_d	E_w	gradient	gamma	alpha	beta
70	9.3031	161.9008	0.230639	11.1244	0.0344	23.1131

In the above table E_d is the sum square`s of difference between observed data and predicted data for the training dataset, E_w is the sum of the squares of the bias and weights, gradient is the simply measure of the change in all weights with regard to the change in error , gamma is the effective number of parameters, α is the half of inverse of dispersion parameter for weights and biases and β is the standard error of the of the model. The Mse = 0.1824136 and the AIC= 14.6867 with $R^2 = 0.84$ which is mean that Age, Sodium and Blood Pressure are capable of explaining 84% of Thrombosis.

Table(2) *demonstrate the performance measurements*

Datasets	MSE	AIC
Training	0.1824136	14.6866924
Testing	0.3131967	19.5516853
Validation	0.21536764	16.1813211

Table(3) *shows the actual and the predicted values of training dataset*

	Training			Training	
No.	**Actual**	**predicted**	**No.**	**Actual**	**predicted**
1	-0.398797316	-0.0941	29	0.50597804	0.82106
2	-0.355578555	-0.25949	30	-0.414205744	-0.97325
3	-0.209950123	0.14208	31	0.048422902	-0.11429
4	-0.341485481	-0.19789	32	6.719144634	5.38638
5	0.048422902	0.66158	33	-0.411762944	-0.98818
6	-0.350129233	-0.0762	34	-0.375308859	-0.55799
7	-0.361215785	0.0014	35	2.867037718	2.14034
8	-0.017814547	0.20232	36	-0.349095741	-0.69253
9	-0.209950123	0.39517	37	-0.327392407	-0.84009
10	-0.374369321	0.2463	38	-0.386113549	-0.69035
11	2.038271008	1.86383	39	-0.346558988	-0.22208
12	0.330284383	0.38383	40	-0.148880136	-0.34684
13	0.424050303	0.86239	41	-0.363094862	0.25839
14	-0.292629491	0.14608	42	-0.333499406	-0.36636
15	-0.374369321	-0.00865	43	-0.092507839	0.30283
16	0.001445988	0.44394	44	-0.375308859	-0.70537
17	-0.355578555	-0.2478	45	-0.343646419	-0.75819
18	-0.308601642	-0.14402	46	-0.252229345	-0.44196
19	0.039027519	-0.49994	47	-0.33115056	-0.42197
20	0.396052062	0.85751	48	-0.360276247	-0.69774
21	-0.167670901	0.29807	49	-0.388274487	0.23312
22	-0.327392407	-0.64622	50	0.026061891	-0.32532
23	-0.33678779	-0.51382	51	0.518192038	0.83956
24	-0.252229345	-0.00547	52	-0.405186176	-0.82973
25	1.711405643	1.7679	53	-0.41608482	-0.11216
26	0.612145865	0.37632	54	-0.308601642	-0.2071
27	-0.374369321	-0.0187	55	-0.255987499	-0.06083
28	1.927499446	1.44199			

Table(3) *shows the actual and the predicted values of testing dataset*

	Testing Dataset	
No.	**Actual**	**Predicted**
1	-0.341485481	-0.24441
2	0.17995826	-0.32937
3	-0.148880136	-0.63921
4	-0.350880864	-0.88336
5	-0.30390395	-0.08429
6	-0.337727328	-0.10934
7	-0.364973938	-0.87878
8	-0.409320145	-0.56906
9	-0.327768223	0.29133

Sum to up table clarify the performance of postulated model for training, testing and validation dataset.

Table(3) *shows the actual and the predicted values of validation dataset*

	Validation Dataset	
No.	Actual	Predicted
1	-0.420716744	-0.86432
2	-0.421083164	-0.603
3	-0.415051328	-0.90619
4	-0.278536417	-0.01866
5	-0.271020111	-0.07813
6	-0.416742497	-0.81411
7	-0.412326667	-0.25955
8	-0.404528499	-0.14568
9	-0.374369321	0.12838

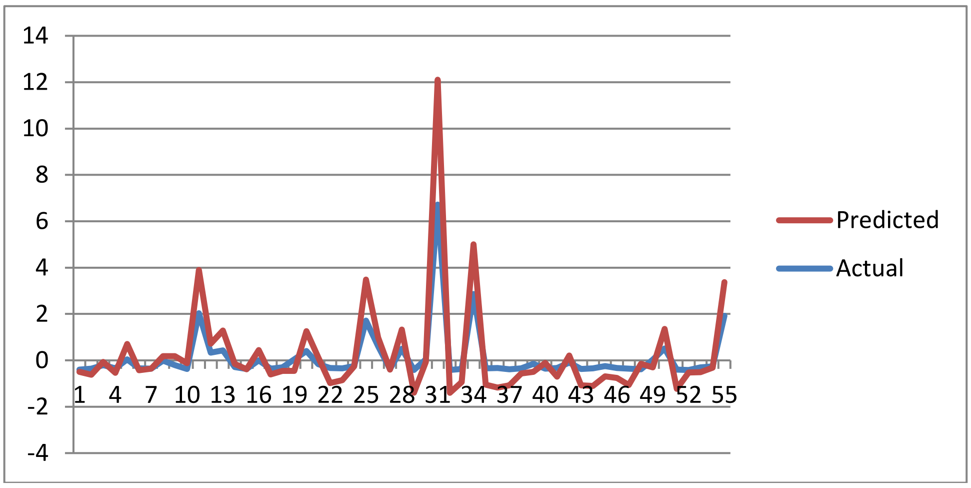
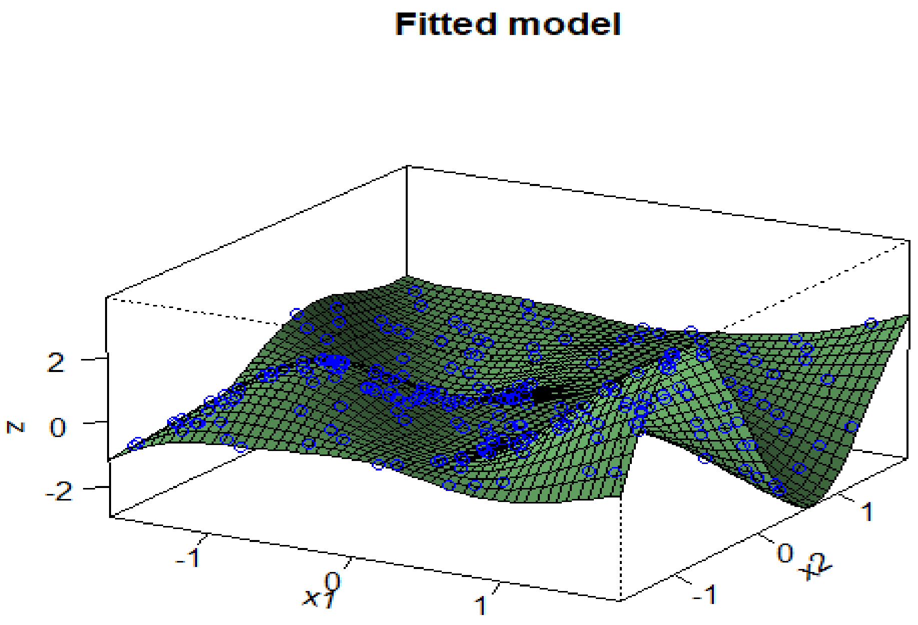


Figure (3) *shows the actual and the predicted values.*



Figure(4) *the 3D plot represents the*

Conclusion

In this study we focused on the thrombosis for 73 patients that have covid-19 as a response variable and Age, Sodium, Blood pressure as factors. Bayesian regression neural network model was used to predict thrombosis for the first time. Age, Sodium and Blood Pressure are capable of explaining 84% of Thrombosis. The Mse is 0.1824136 and the AIC= 14.6867 with R2 = 0.84 which is mean that age, sodium and blood pressure are able to explain 84%.

Acknowledgments

We would want to express our gratitude to all parties who have provided valuable contributions in the form of feedback suggestions in order to make this study as successful and hassle-free as possible. We would also want to extend our gratitude to those who have contributed financially to assist make this study a success. We are of the opinion that in the absence of support from all relevant stakeholders, the outcome will not be as dependable as we had expected

References

Jiang, B., Wu, T. Y., Zheng, C., & Wong, W. H. (2017). Learning summary statistic for approximate Bayesian computation via deep neural network. Statistica Sinica, 1595-1618.

Gustafson, P. (2000). Bayesian regression modeling with interactions and smooth effects. Journal of the American Statistical Association, 95(451), 795-806.

Titterington, D. M. (2004). Bayesian methods for neural networks and related models. Statistical science, 128-139.

Bhosale, M. D., & Singh, T. P. (2017). Development of lifetime milk yield equation using artificial neural network in Holstein Friesian crossbred dairy cattle and comparison with multiple linear regression model. Current Science, 951-955.Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).Dahl, George E., Dong Yu, Li Deng, and Alex Acero. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition." IEEE Transactions on audio, speech, and language processing 20, no. 1 (2011): 30-42.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ... & Zhu, Z. (2016, June). Deep speech 2: End-to-end speech recognition in english and mandarin. In International conference on machine learning (pp. 173-182). PMLR.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Hassabis, D. (2017). Mastering the game of go without human knowledge. nature, 550(7676), 354-359.

Lampinen, J., & Vehtari, A. (2001). Bayesian approach for neural networks—review and case studies. Neural networks, 14(3), 257-274.

Neal, R. M. (2012). Bayesian learning for neural networks (Vol. 118). Springer Science & Business Media.

Denker, J., & LeCun, Y. (1990). Transforming neural-net output levels to probability distributions. Advances in neural information processing systems, 3.

Cybenko, G. (1992). Approximation by superpositions of a sigmoidal function. Math. Control. Signals Syst., 5(4), 455.

Funahashi, K. I. (1989). On the approximate realization of continuous mappings by neural networks. Neural networks, 2(3), 183-192.

Honik, K. (1991). Approximation capabilities of multilayer feedforward network. Neural Networks, 4(2), 251-257.

Gull, S. F., & Skilling, J. (1991). Quantified Maximum Entropy: Mem Sys 5. Users' Manual.

MacKay, D. J. (1992). Bayesian interpolation. Neural computation, 4(3), 415-447.

Mackay, D. J. (1992). Bayesian methods for adaptive methods. PhD thesis. California Institute of Technology.

MacKay, D. J. (1992). A practical Bayesian framework for backpropagation networks. Neural computation, 4(3), 448-472.

Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. arXiv preprint arXiv:1701.02434.

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid monte carlo. Physics letters B, 195(2), 216-222.

Ahmad, N. M. (2010). Bayesian Classifier for a Gaussian Distribution, Decision Surface Equation, with Application. Iraqi Journal of Statistical Science, 18, 35-58.

Ahmed, N. M., & Taher, H. A. (2018). Multi-response Regression Modeling for an Agricultural Experiment. Journal of University of Human Development, 4(2), 46-52.