AMSI VACATION RESEARCH SCHOLARSHIPS 2020–21

Get a Thirst for Research this Summer

When does machine learning work in time series forecasting?

Gaurangi Gupta

Supervised by A/Professor Georgy Sofronov and Dr. Nan Zou

Macquarie University

Vacation Research Scholarships are funded jointly by the Department of Education, Skills and Employment and the Australian Mathematical Sciences Institute.

Machine Learning in time series forecasting

ABSTRACT

Machine learning methods use information and patterns regarding historical values to predict future activity. This project mainly focuses on the applications of machine learning techniques like Artificial Neural Network and Random forests in time series forecasting which is a sequence of observations measured over time.

We will be simulating a time series and will be making predictions using machine learning methods like Artificial Neural Network and Random Forest. We want to check and compare the accuracy and robustness of the machine learning methods in making the predictions.

Particularly, we want to see how well these methods work in autoregressive time series with additional non stationarity. We will test these methods on simulated data which is generated according to a true model. This true model has a trend component, seasonality component and the error term. We will be using monthly data as monthly time series are the most common type of data in businesses and economics and they are also difficult to predict as compared to other seasonal time series such as the quarterly time series.

LITERATURE REVIEW

Time Series

A time series is a sequence of observations measured over time. Time series analysis is the use of a model to predict future values based on previously observed values. The objectives of time series analysis include forecasting future values. In a time series it is assumed that each point is independent of each other. A stationary time series is one which represents such a condition. In a stationary time series, properties like mean, variance and autocorrelation are constant over time.

A time series consists of four components; *trend*; which is the linear or non-linear component that changes over time and doesn't repeat itself, *seasonality*; patterns of change that occur at specific regular intervals, *cyclic fluctuations*; which correspond to periodic fluctuations in

the data which is not seasonal and *irregular variations*; which are the non-random variations of the series. It is due to the belief that seasonality can hinder the measurement of other components and therefore need to be separated.

Sometimes, the seasonal component is not independent of the non-seasonal component and therefore it is not advisable to separate the two components.

The multiplicative model of the time series through which we will be simulating data can be written as:

$$Y_t = T_t.SI_t + E_t;$$

where T_t represents the *trend component*, SI_t represents the *seasonal variation* and E_t represents the irregular or the residual variations.

If the trend is linear, then it can be written by:

$$T_t = a + bt :$$

where

$'a'$ and $'b'$ are constants and $'t'$ is a given time.

A *lag* in a time series represents a fixed amount of passing time. The *'k^{th}'* lag is the time period that happened '*k*' time points before time '*i*'.

These multiplicative time series models are parametric in nature which means that these models have some finite number of parameters associated with them. Parameters are quantities that summarizes the data and provides some information about the data. A limitation of these models is that the model form must be specified without knowing true the data generating process (see, Qian (2017)).

A linear relationship is assumed between the values and their respective historic values. Assuming linear relationship limits the classical method's ability to model complex non-linear problems which are encountered in reality.

Among the traditional predicting methods, for example the Box-Jenkins ARIMA makes use of information underlying in past values of the variable itself and errors in the past. Machine

learning methods makes use of artificial algorithms to learn pattern, underlying given data without knowledge of any kind of underlying relationships (see, Gupta (2020)).

Machine Learning Techniques

Machine learning is the practice of using algorithms to analyze data, learn from it and then make a prediction about something.

Artificial Neural Network

Artificial neural networks are complex multivariable statistical models for the estimation of an unknown expectation on function of a random variable 'y' given explanatory variables 'x'. They have been proved to model any type of relationship with high accuracy.
ANN can be written as nonlinear regression models containing input variables **x** (which can be random variables) and output or dependent variables '**y'**. These models are either used to determine unknown relations between the 2 variables.

A general type of the neural network model is the feed forward network, where the information only passes from the left to the right. It comes into the network through the input layer, goes through the hidden layers and comes out through the output layer. It does not loop between the hidden layers; neither does it loop within a single hidden layer nor in a neuron itself.

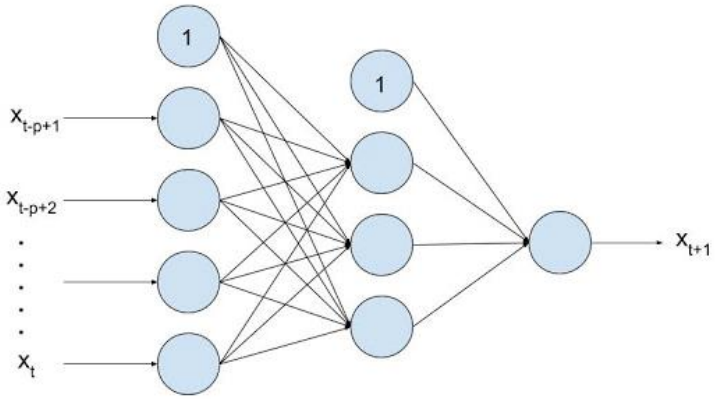

Figure 1. Feed forward mechanism

The x variables or the parameters that determine the y variable form the input layer. The output layer consists of the y variable. The input variables are weighted up and the output variable is calculated. All of the neurons from the input variables have synopsis connecting them to the hidden layer, and these synopses have weights. Some of these weights will have zero weights and some of them will have non-zero weights indicating that some weights are important whereas some of them are not that important. The output variable can be calculated as the weighted sum of all the inputs.

In time series forecasting, the input nodes are formed by the previous lagged observations, while the output layer provides the forecast for the future value. The hidden layers process the information received by the input layer with appropriate nonlinear transfer function. The model can be written as:

$$y_t = \alpha_0 + \sum_{j=1}^{n} \alpha_j f\left(\sum_{i=1}^{m} \beta_{ij}\, y_{t-i} + \beta_{0j}\right) + \varepsilon_t$$

where,

m is the number of input nodes, *n* is the number of hidden nodes, *f* is a sigmoid transfer function, α_j is a vector of weights from the hidden to output nodes and β_{ij} are weights from input to hidden nodes. α_0 and β_{0j} are bias terms. The above equation indicates that a linear transfer function is employed in the output node as desired for forecasting problems.

Random Forests

Random forests are one of the most popular and powerful machine learning algorithms used both in regression and classification. Random forest is a collection of different decision trees. The aim to random forest is to combine a set of decision trees, each of which is constructed using a bootstrap sample.

This algorithm creates the forests with a number of decision trees. More the trees in the forest, more robust is the prediction. In this technique, multiple trees are grown. To classify each object based on attributes, each tree gives its prediction. The forest chooses the classification with the most votes and in the case of regression takes the output of the different trees. This process is known as *bagging* and it improves the stability and accuracy of the model, reduce the variance and helps avoiding overfitting.

Studies have shown that random forest does not overfit as more trees are added but produce a limiting value of the generalization error. The random forest generalization error is estimated by an out-of-bag error, i.e. the error of training points not contained in the bootstrap training sets.

The algorithm of random forest for regression is (see, Dudek(2015)):

1. For k = 1 to K:
    - 1.1. Draw a bootstrap sample L of size N from the training data.
    - 1.2. Grow a random-forest tree T_k to the bootstrapped data, by recursively repeating the following steps for each node of the tree, until the minimum node, size m is reached.
        - 1.2.1. Select F variables at random from the n variables.
        - 1.2.2. Pick the best variable/split-point among the F.
        - 1.2.3. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_k\}_{k=1,2,....K}$.

To make a prediction at a new point x:

The two main parameters of random forest are the number of trees in the forest K and the number of input variables at each split F.

Indicator of efficiency

We used the root mean square error (RMSE) as the indicator of accuracy of the prediction. It is the square root of the mean of the difference between the actual y_t values and the values predicted by the machine learning methods. The higher the error, the less accurate is the technique.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

where,

$Predicted_i$ = The value predicted by the machine learning methods for the i_{th} observation.

$Actual_i$ = The true simulated i_{th} value.

N = Total number of observations or the total number of simulated true values.

The difference between the actual value and observed value is also known as the residual.

Training, Validation and Test dataset

The dataset is generally split into the training, validation and the test set. The training set is the set through which the machine learning technique learns about the data. The algorithms find relations between variables using this data which further can be used to predict values when we have a different dataset. Different models are created with different parameter values. These parameter values are assessed by the validation set.

This is the next portion of the split. It is used to assess how well the model has been trained. This set is used to pick the best model among all the models that are trained. Using this dataset, we choose the model with the best parameters. This is indicated by different indicators like the Root Mean Square error (RMSE), Mean Square Error (MSE), Mean Absolute percentage error (MAPE) etc.

The test set tells about the accuracy of the chosen model. The training dataset cannot be used to measure the performance of the model as the model may have memorized some of the training data and since the validation set was used to pick the best model, it may cause some bias while measuring the performance of the model. The test set behaves as an out-of-sample dataset which is completely new to the model.

Applications of Machine Learning

1. Image Recognition

   Image recognition is one of the most popular applications of machine learning. It is used to identify objects, persons, places etc. The most popular one of them is ‘automatic friend tagging suggestion’. This is when social media applications ask the user automatically suggests the name of the person in the photograph uploaded.

2. Speech Recognition

   This is the process of converting voice instructions to text. Google assistant, Siri, Alexa are all using speech recognition technology.

3. Product Recommendations

It is through machine learning algorithms that we get suggestions for entertainment series or movies on Netflix after we have watched a film or suggestions on any product we should buy when we search something on Amazon.

METHODOLOGY

GENERATING THE TIME SERIES.

A time series was simulated in order to carry out all the tests. We have simulated a monthly time series as it is very common for forecasting. Moreover, these are difficult to deal with than other types of seasonal data such as yearly or quarterly.
Simulated time series was generated according to the multiplicative model:

$$Y_t = T_t.SI_t + E_t,$$

where,

Y_t is a function of time $'t'$ and is an observation at a given time $'t'$.

$T_t = 100 + 0.6t$; is the linear trend,

SI_t is the seasonal index as given in Table 1,

E_t is the error term following a normal distribution, N(0,σ^2).

The errors are drawn from a standard normal distribution.

A total of 228 points were generated.

The table below gives the seasonal indices:

Month	January	February	March	April	May	June	July	August	September	October	November	December
SI	0.75	0.80	0.82	0.90	0.94	0.92	0.91	0.99	0.95	1.02	1.20	1.80

The plots for the seasonal time series that was simulated is as shown below:

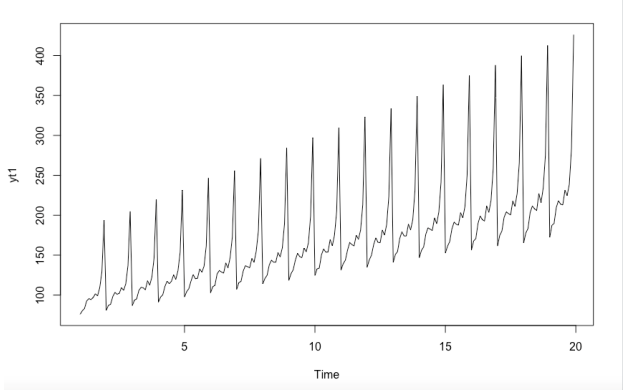

Figure 2: Simulated seasonal time series with noise level (σ^2) = 1.

Machine Learning Techniques for forecasting

We consider 10 lag numbers as the independent variable or in other words we would like to find the association of 'y' at a particular time with its different lagged values. Lagged values are exactly the same as the original time series but they are shifted by the given time period. For example: y_{t-1} is the same time series as y_t but shifted back by one time period.
The lag numbers that we consider here are: 1-4, 12-14, 24, 25 and 36. We will see how a given observation y_t is related to its past observations (y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-12}, y_{t-13}, y_{t-14}, y_{t-24}, y_{t-25}, y_{t-36}). These lagged observations will be inputs to the model. Each model with the respective number of lags will also include the previous lags in the model. For example, if the machine learning model considers lag 12 then lags 1, 2, 3 and 4 will also be included in the model (see, Zhang (2005)).

We also created lag plots to see whether the time series is random or not. We want to see whether a particular value has any correlation with its previous values. In a lag plot values y_t are plotted against values y_{t-k}, where *'k'* represents the time period. These are the lag plots of our time series:

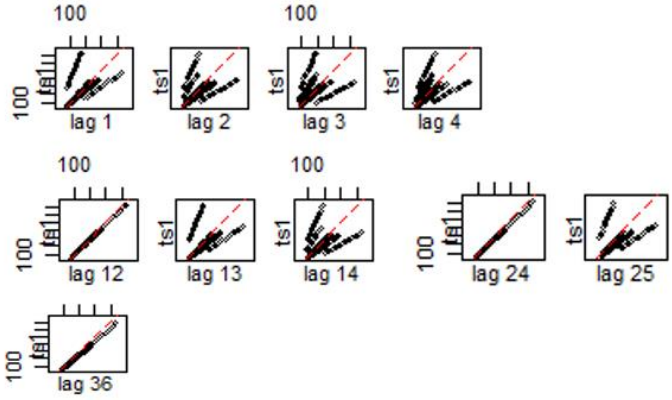

Figure 3. Lag plots of simulated time series

A linear lag plot indicates that the time series values are correlated to its previous time series values. We expect to see a high correlation between observations that are 12, 24 and 36 months apart due to seasonality.

Therefore, we see that the lag plots of lags 12, 24 and 36 are linear. This is because the time series that was simulated was seasonal and since the frequency of the time series was 12, every 12^{th} observation will be correlated to each other. Hence, we see that the lag 12, lag 24 and lag 36 plots are highly correlated to each other.

In the software R, we have used the *embed* function to generate lagged data. The dataset now has a total of 192 rows.

After the formation of the dataset, we split the data into training, validation and test dataset. The last 12 values of the dataset are assigned to the test set, second last 12 values are assigned to the validation set and the rest 168 values are assigned to the training set.

In using the neural networks, the number of hidden nodes for each model vary from 2-14 with an increment of 2. Thus, the hidden layers we considered are (2,4,6,8,10,12,14).

Therefore, there will be a total of 70 models for training the data.

The *neuralnet* function was used in order to generate the models. The hyper parameters that were tuned in the function were the number of hidden layers and the number of parameters. All the 70 models were trained on the training set. These models were then evaluated on the validation set under the criterion of root mean square error (RMSE). The

best model is the one which has the lowest RMSE and this best model was then applied to the testing data.

Artificial Neural networks work on a 'rolling window' approach, where we predict value that is one step ahead and then use that prediction to predict the next value. The idea is to predict x(t+1), given x(t) and then using x(t+1), predict x(t+2). In order to perform that in the software, we had to standardize the data using the *scaled* function before running the artificial neural network. This is to prevent larger values from dominating the result and producing the same values.

The model which had the lowest RMSE when evaluated on the validation set had lags 1,2,3,4,12,13,14 and 24 as the covariates and had 8 hidden layers in it.

In Random Forest, we wrote a code to automatically select the best 'number of variables at each split' or '*mtry*' in the *randomforest* function in R. This was done to select the '*mtry*' which has the least out-of-bag error.

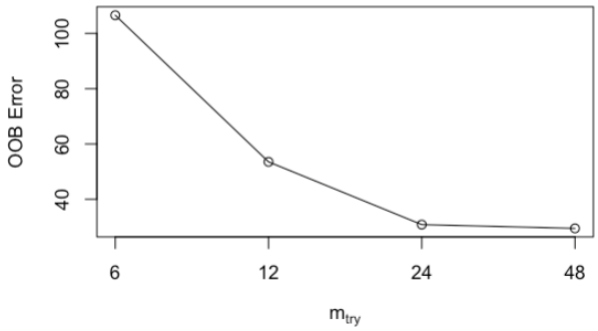

Figure 4. Plot of OOB error against mtry

The best mtry was automatically plugged in the random forest function in R and the ten models were trained on the training data. These 10 models were then evaluated on the validation data and the model with the lowest RMSE was selected. The best model selected also had lags 1,2,3,4,12,13,14 and 24 as the covariates. This model was then evaluated on the testing data and a final RMSE was evaluated.

RESULTS

The table below shows the comparison between the actual simulated time series values and the values predicted by the two machine learning algorithms: ANN and Random Forest.

Original simulated testing set	Predicted values by ANN	Predicted values by Random Forest
170.7464	169.4577	174.0609
185.3155	183.4479	183.5681
189.5513	187.7231	188.6169
211.0320	208.3834	208.9889
219.4648	217.4687	219.0205
213.3588	212.4005	216.0470
209.8160	212.9261	214.3617
233.2104	230.3436	234.2761
222.9176	222.7398	230.5797
240.9976	242.7966	253.8221
286.2513	284.3057	305.2757
425.5202	420.0849	379.2284

Table 1 Comparison between original simulated time series values and the predictions by the Machine learning algorithms

Looking at the values we can conclude that artificial neural network predicts values slightly closer to the original simulated time series values as compared to the values predicted by the random forest algorithm.

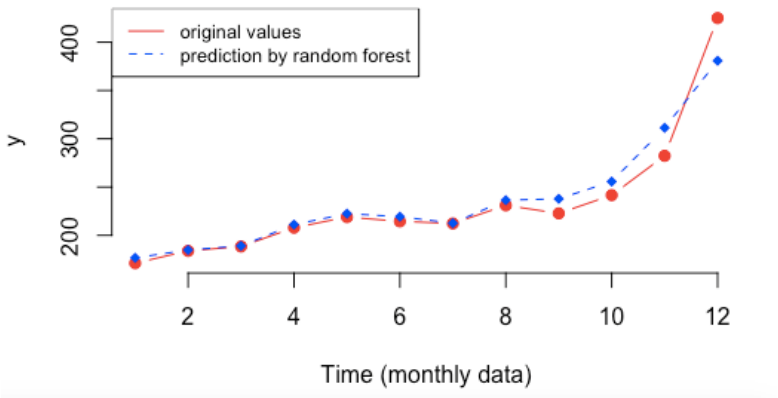

Figure 5: Comparison between the original and the values predicted by random forest.

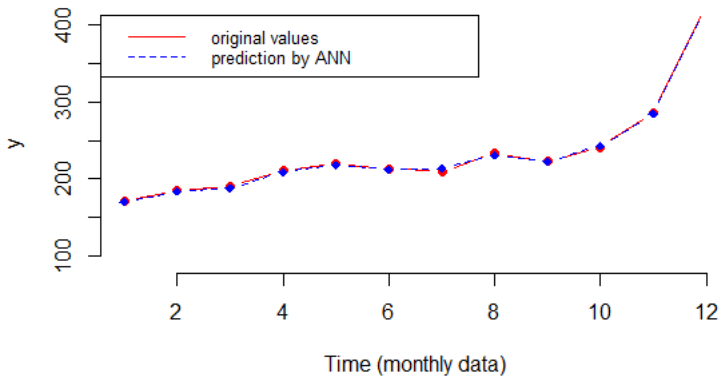

Figure 6: Comparison between the original and the values predicted by artificial neural network.

Also, looking at the figures 5 and 6, we see that although the points predicted by the random forests in figure 5 lie close to the original time series values but artificial neural networks have given phenomenal results in predicting the original simulated time series as can be seen in figure 6. Thus, the predictions made by artificial neural network are more robust in comparison to the predictions made by random forest.

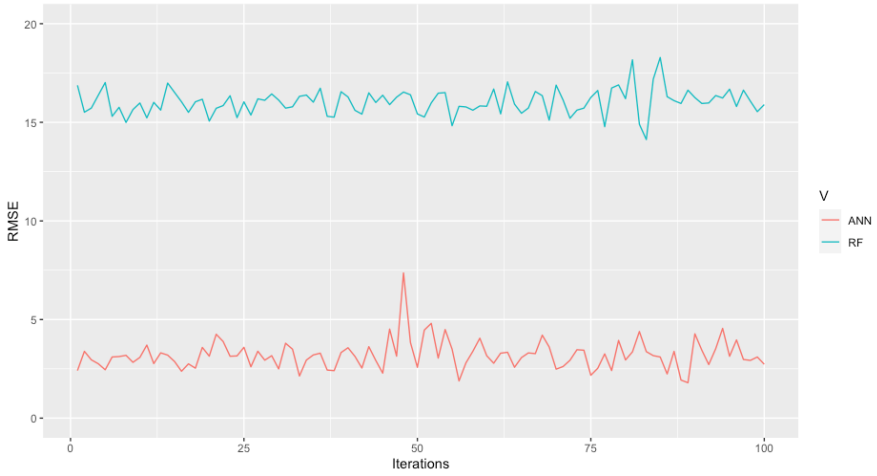

Figure 7: RMSE values across 100 iterations for both the algorithms

On iterating the simulations 100 times, we find that the RMSE between original Y_t values and the values predicted by the two machine learning algorithms vary fairly across a constant mean. The average RMSE given by artificial neural network is **3.07427**, whereas the average RMSE between the original Y_t values and the values predicted by the random forest is **15.90248**, thus giving evidence that artificial neural network is performing better than random forest.

CONCLUSIONS AND REMARKS

- Seasonal and trend variations are the most encountered phenomena in sectors of business and economics. In this paper, we examine the capability of two popular machine learning algorithms; artificial neural network and random forest to predict the time series with trend and seasonality.
- We found in this brief study, that for the dataset that was simulated in this study, artificial neural network proves to be better in predicting the time series as compared to random forest.
- Future work concerns tuning other parameters of random forest like the number of decision trees.
- We shall also consider backpropagation mechanism in artificial neural network wherein the weights are adjusted to give the most accurate prediction.
- We shall also consider using a real-world time series to see which of the two methods perform better.

REFERENCES

- Qian, X.Y. and Gao, S., 2017. Financial series prediction: Comparison between precision of time series models and machine learning methods. *arXiv preprint arXiv:1706.00948*.

- Gupta, A. and Kumar, A., 2020, June. Mid Term Daily Load Forecasting using ARIMA, Wavelet-ARIMA and Machine Learning. In *2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)* (pp. 1-5). IEEE.

- Zhang, G.P. and Qi, M., 2005. Neural network forecasting for seasonal and trend time series. *European journal of operational research*, *160*(2), pp.501-514.

- Dudek, G., 2015. Short-term load forecasting using random forests. In *Intelligent Systems' 2014* (pp. 821-828). Springer, Cham.

APPENDIX

R CODE:

```
#R CODE FOR AMSI VRS

#**SIMULATION OF THE TIME SERIES** yt = T.SI +E

#generating the vector 't'
t<- c(1:228)
#generating the trend vector 'T'
T<- 100+(0.6*t)
#seasonal indices as given in table 1
si<- c(0.75,0.80,0.82,0.90,0.94,0.92,0.91,0.99,0.95,1.02,1.20,1.80)
#multiplying the seasonal and trend component
mx<- si*T
#computing the error term
e1<- rnorm(228, mean = 0, sd = 1)
yt1<- mx+e1

ts1<- ts(data = yt1, frequency = 12)
plot(ts1)

#**USING ARTIFICIAL NEURAL NETWORK**

#CREATING LAGS
#mydf<- data.frame(yt1) #converting yt1 to data frame
lag <- embed(yt1, 37) #creating lags
df_lag<- data.frame(lag) #converting matrix to data frame.
colnames(df_lag)<- c("original", "lag1", "lag2", "lag3", "lag4", "lag5", "lag6", "lag7", "lag8", "lag9", "lag10",
"lag11", "lag12", "lag13", "lag14", "lag15", "lag16", "lag17", "lag18", "lag19", "lag20", "lag21", "lag22", "lag23",
"lag24", "lag25","lag26", "lag27", "lag28", "lag29", "lag30", "lag31", "lag32", "lag33", "lag34", "lag35", "lag36")

#SPLITTING THE DATA IN TRAINING, VALIDATION AND TEST SET.
training_set<- df_lag[1:168,] #the embed() function has reduced the size of the data.
validation_set<- df_lag[169:180,]
testing_set<- df_lag[181:192,]

training_set <- scale(training_set)
```

```
training_set<-as.data.frame(training_set)
validation_set <- scale(validation_set)
validation_set <-as.data.frame(validation_set)

#fitting neural network to the 70 models
# 10 models for 2 hidden nodes
library(neuralnet)
NN1<- neuralnet(original ~ lag1, data = training_set, hidden = 2, linear.output = T)
NN2<- neuralnet(original ~ lag1 +lag2, data = training_set, hidden = 2, linear.output = T)
NN3<- neuralnet(original ~ lag1 + lag2 +lag3, data = training_set, hidden = 2, linear.output = T)
NN4<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4, data = training_set, hidden = 2, linear.output = T)
NN5<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12, data = training_set, hidden = 2, linear.output = T)
NN6<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13, data = training_set, hidden = 2, linear.output = T)
NN7<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14, data = training_set, hidden = 2, linear.output = T)
NN8<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24, data = training_set, hidden = 2, linear.output = T)
NN9<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25, data = training_set, hidden = 2, linear.output = T)
NN10<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25 +lag36, data = training_set, hidden = 2, linear.output = T)

# 10 models for 4 hidden nodes
NN11<- neuralnet(original ~ lag1, data = training_set, hidden = 4, linear.output = T)
NN12<- neuralnet(original ~ lag1 +lag2, data = training_set, hidden = 4, linear.output = T)
NN13<- neuralnet(original ~ lag1 + lag2 +lag3, data = training_set, hidden = 4, linear.output = T)
NN14<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4, data = training_set, hidden = 4, linear.output = T)
NN15<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12, data = training_set, hidden = 4, linear.output = T)
NN16<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13, data = training_set, hidden = 4, linear.output = T)
NN17<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14, data = training_set, hidden = 4, linear.output = T)
NN18<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24, data = training_set, hidden = 4, linear.output = T)
NN19<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25, data = training_set, hidden = 4, linear.output = T)
```

```
NN20<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25 +lag36, data = training_set, hidden = 4, linear.output = T)

#10 models for 6 hidden nodes
NN21<- neuralnet(original ~ lag1, data = training_set, hidden = 6, linear.output = T)
NN22<- neuralnet(original ~ lag1 +lag2, data = training_set, hidden = 6, linear.output = T)
NN23<- neuralnet(original ~ lag1 + lag2 +lag3, data = training_set, hidden = 6, linear.output = T)
NN24<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4, data = training_set, hidden = 6, linear.output = T)
NN25<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12, data = training_set, hidden = 6, linear.output = T)
NN26<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13, data = training_set, hidden = 6, linear.output = T)
NN27<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14, data = training_set, hidden = 6, linear.output = T)
NN28<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24, data = training_set, hidden = 6, linear.output = T)
NN29<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25, data = training_set, hidden = 6, linear.output = T)
NN30<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25 +lag36, data = training_set, hidden = 6, linear.output = T)

#10 models for 8 hidden nodes
NN31<- neuralnet(original ~ lag1, data = training_set, hidden = 8, linear.output = T)
NN32<- neuralnet(original ~ lag1 +lag2, data = training_set, hidden = 8, linear.output = T)
NN33<- neuralnet(original ~ lag1 + lag2 +lag3, data = training_set, hidden = 8, linear.output = T)
NN34<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4, data = training_set, hidden = 8, linear.output = T)
NN35<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12, data = training_set, hidden = 8, linear.output = T)
NN36<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13, data = training_set, hidden = 8, linear.output = T)
NN37<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14, data = training_set, hidden = 8, linear.output = T)
NN38<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24, data = training_set, hidden = 8, linear.output = T)
NN39<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25, data = training_set, hidden = 8, linear.output = T)
NN40<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25 +lag36, data = training_set, hidden = 8, linear.output = T)

#10 models for 10 hidden nodes
```

```
NN41<- neuralnet(original ~ lag1, data = training_set, hidden = 10, linear.output = T)
NN42<- neuralnet(original ~ lag1 +lag2, data = training_set, hidden = 10, linear.output = T)
NN43<- neuralnet(original ~ lag1 + lag2 +lag3, data = training_set, hidden = 10, linear.output = T)
NN44<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4, data = training_set, hidden = 10, linear.output = T)
NN45<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12, data = training_set, hidden = 10, linear.output = T)
NN46<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13, data = training_set, hidden = 10, linear.output = T)
NN47<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14, data = training_set, hidden = 10, linear.output = T)
NN48<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24, data = training_set, hidden = 10, linear.output = T)
NN49<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25, data = training_set, hidden = 10, linear.output = T)
NN50<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25 +lag36, data = training_set, hidden = 10, linear.output = T)

#10 models for 12 hidden nodes
NN51<- neuralnet(original ~ lag1, data = training_set, hidden = 12, linear.output = T)
NN52<- neuralnet(original ~ lag1 +lag2, data = training_set, hidden = 12, linear.output = T)
NN53<- neuralnet(original ~ lag1 + lag2 +lag3, data = training_set, hidden = 12, linear.output = T)
NN54<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4, data = training_set, hidden = 12, linear.output = T)
NN55<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12, data = training_set, hidden = 12, linear.output = T)
NN56<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13, data = training_set, hidden = 12, linear.output = T)
NN57<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14, data = training_set, hidden = 12, linear.output = T)
NN58<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24, data = training_set, hidden = 12, linear.output = T)
NN59<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25, data = training_set, hidden = 12, linear.output = T)
NN60<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25 +lag36, data = training_set, hidden = 12, linear.output = T)


#10 models for 14 hidden nodes
NN61<- neuralnet(original ~ lag1, data = training_set, hidden = 14, linear.output = T)
```

```
NN62<- neuralnet(original ~ lag1 +lag2, data = training_set, hidden = 14, linear.output = T)
NN63<- neuralnet(original ~ lag1 + lag2 +lag3, data = training_set, hidden = 14, linear.output = T)
NN64<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4, data = training_set, hidden = 14, linear.output = T)
NN65<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12, data = training_set, hidden = 14, linear.output = T)
NN66<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13, data = training_set, hidden = 14, linear.output = T)
NN67<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14, data = training_set, hidden = 14, linear.output = T)
NN68<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24, data = training_set, hidden = 14, linear.output = T)
NN69<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25, data = training_set, hidden = 14, linear.output = T)
NN70<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24 +lag25 +lag36, data = training_set, hidden = 14, linear.output = T)

# subsetting the validation set for all the lag variables to compute the result.
x_test1 <- subset(validation_set, select = c("lag1"))
x_test2 <- subset(validation_set, select = c("lag1", "lag2"))
x_test3 <- subset(validation_set, select = c("lag1", "lag2", "lag3"))
x_test4 <- subset(validation_set, select = c("lag1", "lag2", "lag3", "lag4"))
x_test12 <- subset(validation_set, select = c("lag1", "lag2", "lag3", "lag4", "lag12"))
x_test13 <- subset(validation_set, select = c("lag1", "lag2", "lag3", "lag4", "lag12","lag13"))
x_test14 <- subset(validation_set, select = c("lag1", "lag2", "lag3", "lag4","lag12", "lag13", "lag14"))
x_test24 <- subset(validation_set, select = c("lag1", "lag2", "lag3", "lag4", "lag12", "lag13", "lag14","lag24"))
x_test25 <- subset(validation_set, select = c("lag1", "lag2", "lag3", "lag4", "lag12", "lag13", "lag14","lag24", "lag25"))
x_test36 <- subset(validation_set, select = c("lag1", "lag2", "lag3", "lag4", "lag12", "lag13", "lag14", "lag24", "lag25","lag36"))

validation_set<-data.frame(validation_set)
testing_set<-data.frame(testing_set)
training_set<-data.frame(training_set)

#computing the results for all 70 models on the Validation set.
nn.results1 <- compute(NN1, x_test1)
result1 <- data.frame(actual = validation_set$original, prediction = nn.results1$net.result)
rmse1 <- sqrt(mean((result1$actual - result1$prediction)^2))
```

```
nn.results2 <- compute(NN2, x_test2)
result2 <- data.frame(actual = validation_set$original, prediction = nn.results2$net.result)
rmse2 <- sqrt(mean((result2$actual - result2$prediction)^2))

nn.results3 <- compute(NN3, x_test3)
result3 <- data.frame(actual = validation_set$original, prediction = nn.results3$net.result)
rmse3 <- sqrt(mean((result3$actual - result3$prediction)^2))

nn.results4 <- compute(NN4, x_test4)
result4 <- data.frame(actual = validation_set$original, prediction = nn.results4$net.result)
rmse4 <- sqrt(mean((result4$actual - result4$prediction)^2))

nn.results5 <- compute(NN5, x_test12)
result5 <- data.frame(actual = validation_set$original, prediction = nn.results5$net.result)
rmse5 <- sqrt(mean((result5$actual - result5$prediction)^2))

nn.results6 <- compute(NN6, x_test13)
result6 <- data.frame(actual = validation_set$original, prediction = nn.results6$net.result)
rmse6 <- sqrt(mean((result6$actual - result6$prediction)^2))

nn.results7 <- compute(NN7, x_test14)
result7 <- data.frame(actual = validation_set$original, prediction = nn.results7$net.result)
rmse7 <- sqrt(mean((result7$actual - result7$prediction)^2))

nn.results8 <- compute(NN8, x_test24)
result8 <- data.frame(actual = validation_set$original, prediction = nn.results8$net.result)
rmse8 <- sqrt(mean((result8$actual - result8$prediction)^2))

nn.results9 <- compute(NN9, x_test25)
result9 <- data.frame(actual = validation_set$original, prediction = nn.results9$net.result)
rmse9 <- sqrt(mean((result9$actual - result9$prediction)^2))

nn.results10 <- compute(NN10, x_test36)
result10 <- data.frame(actual = validation_set$original, prediction = nn.results10$net.result)
rmse10 <- sqrt(mean((result10$actual - result10$prediction)^2))

nn.results11 <- compute(NN11, x_test1)
```

```
result11 <- data.frame(actual = validation_set$original, prediction = nn.results11$net.result)
rmse11 <- sqrt(mean((result11$actual - result11$prediction)^2))

nn.results12 <- compute(NN12, x_test2)
result12 <- data.frame(actual = validation_set$original, prediction = nn.results12$net.result)
rmse12 <- sqrt(mean((result12$actual - result12$prediction)^2))

nn.results13 <- compute(NN13, x_test3)
result13 <- data.frame(actual = validation_set$original, prediction = nn.results13$net.result)
rmse13 <- sqrt(mean((result13$actual - result13$prediction)^2))

nn.results14 <- compute(NN14, x_test4)
result14 <- data.frame(actual = validation_set$original, prediction = nn.results14$net.result)
rmse14 <- sqrt(mean((result14$actual - result14$prediction)^2))

nn.results15 <- compute(NN15, x_test12)
result15 <- data.frame(actual = validation_set$original, prediction = nn.results15$net.result)
rmse15 <- sqrt(mean((result15$actual - result15$prediction)^2))

nn.results16 <- compute(NN16, x_test13)
result16 <- data.frame(actual = validation_set$original, prediction = nn.results16$net.result)
rmse16 <- sqrt(mean((result16$actual - result16$prediction)^2))

nn.results17 <- compute(NN17, x_test14)
result17 <- data.frame(actual = validation_set$original, prediction = nn.results17$net.result)
rmse17 <- sqrt(mean((result17$actual - result17$prediction)^2))

nn.results18 <- compute(NN18, x_test24)
result18 <- data.frame(actual = validation_set$original, prediction = nn.results18$net.result)
rmse18 <- sqrt(mean((result18$actual - result18$prediction)^2))

nn.results19 <- compute(NN19, x_test25)
result19 <- data.frame(actual = validation_set$original, prediction = nn.results19$net.result)
rmse19 <- sqrt(mean((result19$actual - result19$prediction)^2))

nn.results20 <- compute(NN20, x_test36)
result20 <- data.frame(actual = validation_set$original, prediction = nn.results20$net.result)
rmse20 <- sqrt(mean((result20$actual - result20$prediction)^2))
```

```
nn.results21 <- compute(NN21, x_test1)
result21 <- data.frame(actual = validation_set$original, prediction = nn.results21$net.result)
rmse21 <- sqrt(mean((result21$actual - result21$prediction)^2))

nn.results22 <- compute(NN22, x_test2)
result22 <- data.frame(actual = validation_set$original, prediction = nn.results22$net.result)
rmse22 <- sqrt(mean((result22$actual - result22$prediction)^2))

nn.results23 <- compute(NN23, x_test3)
result23 <- data.frame(actual = validation_set$original, prediction = nn.results23$net.result)
rmse23 <- sqrt(mean((result23$actual - result23$prediction)^2))

nn.results24 <- compute(NN24, x_test4)
result24 <- data.frame(actual = validation_set$original, prediction = nn.results24$net.result)
rmse24 <- sqrt(mean((result24$actual - result24$prediction)^2))

nn.results25 <- compute(NN25, x_test12)
result25 <- data.frame(actual = validation_set$original, prediction = nn.results25$net.result)
rmse25 <- sqrt(mean((result25$actual - result25$prediction)^2))

nn.results26 <- compute(NN26, x_test13)
result26 <- data.frame(actual = validation_set$original, prediction = nn.results26$net.result)
rmse26 <- sqrt(mean((result26$actual - result26$prediction)^2))

nn.results27 <- compute(NN27, x_test14)
result27 <- data.frame(actual = validation_set$original, prediction = nn.results27$net.result)
rmse27 <- sqrt(mean((result27$actual - result27$prediction)^2))

nn.results28 <- compute(NN28, x_test24)
result28 <- data.frame(actual = validation_set$original, prediction = nn.results28$net.result)
rmse28 <- sqrt(mean((result28$actual - result28$prediction)^2))

nn.results29 <- compute(NN29, x_test25)
result29 <- data.frame(actual = validation_set$original, prediction = nn.results29$net.result)
rmse29 <- sqrt(mean((result29$actual - result29$prediction)^2))

nn.results30 <- compute(NN30, x_test36)
```

```
result30 <- data.frame(actual = validation_set$original, prediction = nn.results30$net.result)
rmse30 <- sqrt(mean((result30$actual - result30$prediction)^2))

nn.results31 <- compute(NN31, x_test1)
result31 <- data.frame(actual = validation_set$original, prediction = nn.results31$net.result)
rmse31 <- sqrt(mean((result31$actual - result31$prediction)^2))

nn.results32 <- compute(NN32, x_test2)
result32 <- data.frame(actual = validation_set$original, prediction = nn.results32$net.result)
rmse32 <- sqrt(mean((result32$actual - result32$prediction)^2))

nn.results33 <- compute(NN33, x_test3)
result33 <- data.frame(actual = validation_set$original, prediction = nn.results33$net.result)
rmse33 <- sqrt(mean((result33$actual - result33$prediction)^2))

nn.results34 <- compute(NN34, x_test4)
result34 <- data.frame(actual = validation_set$original, prediction = nn.results34$net.result)
rmse34 <- sqrt(mean((result34$actual - result34$prediction)^2))

nn.results35 <- compute(NN35, x_test12)
result35 <- data.frame(actual = validation_set$original, prediction = nn.results35$net.result)
rmse35 <- sqrt(mean((result35$actual - result35$prediction)^2))

nn.results36 <- compute(NN36, x_test13)
result36 <- data.frame(actual = validation_set$original, prediction = nn.results36$net.result)
rmse36 <- sqrt(mean((result36$actual - result36$prediction)^2))

nn.results37 <- compute(NN37, x_test14)
result37 <- data.frame(actual = validation_set$original, prediction = nn.results37$net.result)
rmse37 <- sqrt(mean((result37$actual - result37$prediction)^2))

nn.results38 <- compute(NN38, x_test24)
result38 <- data.frame(actual = validation_set$original, prediction = nn.results38$net.result)
rmse38 <- sqrt(mean((result38$actual - result38$prediction)^2))

nn.results39 <- compute(NN39, x_test25)
result39 <- data.frame(actual = validation_set$original, prediction = nn.results39$net.result)
rmse39 <- sqrt(mean((result39$actual - result39$prediction)^2))
```

```
nn.results40 <- compute(NN40, x_test36)
result40 <- data.frame(actual = validation_set$original, prediction = nn.results40$net.result)
rmse40 <- sqrt(mean((result40$actual - result40$prediction)^2))

nn.results41 <- compute(NN41, x_test1)
result41 <- data.frame(actual = validation_set$original, prediction = nn.results41$net.result)
rmse41 <- sqrt(mean((result41$actual - result41$prediction)^2))

nn.results42 <- compute(NN42, x_test2)
result42 <- data.frame(actual = validation_set$original, prediction = nn.results42$net.result)
rmse42 <- sqrt(mean((result42$actual - result42$prediction)^2))

nn.results43 <- compute(NN43, x_test3)
result43 <- data.frame(actual = validation_set$original, prediction = nn.results43$net.result)
rmse43 <- sqrt(mean((result43$actual - result43$prediction)^2))

nn.results44 <- compute(NN44, x_test4)
result44 <- data.frame(actual = validation_set$original, prediction = nn.results44$net.result)
rmse44 <- sqrt(mean((result44$actual - result44$prediction)^2))

nn.results45 <- compute(NN45, x_test12)
result45 <- data.frame(actual = validation_set$original, prediction = nn.results45$net.result)
rmse45 <- sqrt(mean((result45$actual - result45$prediction)^2))

nn.results46 <- compute(NN46, x_test13)
result46 <- data.frame(actual = validation_set$original, prediction = nn.results46$net.result)
rmse46 <- sqrt(mean((result46$actual - result46$prediction)^2))

nn.results47 <- compute(NN47, x_test14)
result47 <- data.frame(actual = validation_set$original, prediction = nn.results47$net.result)
rmse47 <- sqrt(mean((result47$actual - result47$prediction)^2))

nn.results48 <- compute(NN48, x_test24)
result48 <- data.frame(actual = validation_set$original, prediction = nn.results48$net.result)
rmse48 <- sqrt(mean((result48$actual - result48$prediction)^2))

nn.results49 <- compute(NN49, x_test25)
```

```
result49 <- data.frame(actual = validation_set$original, prediction = nn.results49$net.result)
rmse49 <- sqrt(mean((result49$actual - result49$prediction)^2))

nn.results50 <- compute(NN50, x_test36)
result50 <- data.frame(actual = validation_set$original, prediction = nn.results50$net.result)
rmse50 <- sqrt(mean((result50$actual - result50$prediction)^2))

nn.results51 <- compute(NN51, x_test1)
result51 <- data.frame(actual = validation_set$original, prediction = nn.results51$net.result)
rmse51<- sqrt(mean((result51$actual - result51$prediction)^2))

nn.results52 <- compute(NN52, x_test2)
result52 <- data.frame(actual = validation_set$original, prediction = nn.results52$net.result)
rmse52 <- sqrt(mean((result52$actual - result52$prediction)^2))

nn.results53 <- compute(NN53, x_test3)
result53 <- data.frame(actual = validation_set$original, prediction = nn.results53$net.result)
rmse53 <- sqrt(mean((result53$actual - result53$prediction)^2))

nn.results54 <- compute(NN54, x_test4)
result54 <- data.frame(actual = validation_set$original, prediction = nn.results54$net.result)
rmse54 <- sqrt(mean((result54$actual - result54$prediction)^2))

nn.results55 <- compute(NN55, x_test12)
result55 <- data.frame(actual = validation_set$original, prediction = nn.results55$net.result)
rmse55 <- sqrt(mean((result55$actual - result55$prediction)^2))

nn.results56 <- compute(NN56, x_test13)
result56 <- data.frame(actual = validation_set$original, prediction = nn.results56$net.result)
rmse56 <- sqrt(mean((result56$actual - result56$prediction)^2))

nn.results57 <- compute(NN57, x_test14)
result57 <- data.frame(actual = validation_set$original, prediction = nn.results57$net.result)
rmse57 <- sqrt(mean((result57$actual - result57$prediction)^2))

nn.results58 <- compute(NN58, x_test24)
result58 <- data.frame(actual = validation_set$original, prediction = nn.results58$net.result)
rmse58 <- sqrt(mean((result58$actual - result58$prediction)^2))
```

```
nn.results59 <- compute(NN59, x_test25)
result59 <- data.frame(actual = validation_set$original, prediction = nn.results59$net.result)
rmse59 <- sqrt(mean((result59$actual - result59$prediction)^2))

nn.results60 <- compute(NN60, x_test36)
result60 <- data.frame(actual = validation_set$original, prediction = nn.results60$net.result)
rmse60 <- sqrt(mean((result60$actual - result60$prediction)^2))

nn.results61 <- compute(NN61, x_test1)
result61 <- data.frame(actual = validation_set$original, prediction = nn.results61$net.result)
rmse61 <- sqrt(mean((result61$actual - result61$prediction)^2))

nn.results62 <- compute(NN62, x_test2)
result62 <- data.frame(actual = validation_set$original, prediction = nn.results62$net.result)
rmse62 <- sqrt(mean((result62$actual - result62$prediction)^2))

nn.results63 <- compute(NN63, x_test3)
result63 <- data.frame(actual = validation_set$original, prediction = nn.results63$net.result)
rmse63 <- sqrt(mean((result63$actual - result63$prediction)^2))

nn.results64 <- compute(NN64, x_test4)
result64 <- data.frame(actual = validation_set$original, prediction = nn.results64$net.result)
rmse64 <- sqrt(mean((result64$actual - result64$prediction)^2))

nn.results65 <- compute(NN65, x_test12)
result65 <- data.frame(actual = validation_set$original, prediction = nn.results65$net.result)
rmse65 <- sqrt(mean((result65$actual - result65$prediction)^2))

nn.results66 <- compute(NN66, x_test13)
result66 <- data.frame(actual = validation_set$original, prediction = nn.results66$net.result)
rmse66 <- sqrt(mean((result66$actual - result66$prediction)^2))

nn.results67 <- compute(NN67, x_test14)
result67 <- data.frame(actual = validation_set$original, prediction = nn.results67$net.result)
rmse67 <- sqrt(mean((result67$actual - result67$prediction)^2))
nn.results68 <- compute(NN68, x_test24)
result68 <- data.frame(actual = validation_set$original, prediction = nn.results68$net.result)
```

```
rmse68 <- sqrt(mean((result68$actual - result68$prediction)^2))

nn.results69 <- compute(NN69, x_test25)
result69 <- data.frame(actual = validation_set$original, prediction = nn.results69$net.result)
rmse69 <- sqrt(mean((result69$actual - result69$prediction)^2))

nn.results70 <- compute(NN70, x_test36)
result70 <- data.frame(actual = validation_set$original, prediction = nn.results70$net.result)
rmse70 <- sqrt(mean((result70$actual - result70$prediction)^2))

#creating a data frame of RMSE
rmse_df<- data.frame(rmse1, rmse2, rmse3, rmse4, rmse5, rmse6, rmse7, rmse8, rmse9, rmse10, rmse11,
rmse12, rmse13, rmse14, rmse15, rmse16, rmse17,rmse18, rmse19, rmse20, rmse21, rmse22, rmse23,
rmse24, rmse25, rmse26, rmse27, rmse28, rmse29,rmse30,rmse31, rmse32, rmse33, rmse34, rmse35,
rmse36, rmse37, rmse38, rmse39, rmse40, rmse41, rmse42, rmse43, rmse44, rmse45, rmse46, rmse47,
rmse48, rmse49, rmse50, rmse51, rmse52, rmse53, rmse54, rmse55, rmse56, rmse57, rmse58, rmse59,
rmse60, rmse61, rmse62, rmse63, rmse64, rmse65, rmse66, rmse67, rmse68, rmse69, rmse70)
View(rmse_df)

#selecting the minimum rmse from rmse_df
which.min(rmse_df) #this gives NN38 as the model with lowest rmse


# code to perform 100 iterations to calculate avergae RMSE from ANN
t<- c(1:228)

#generating the trend vector 'T'
T<- 100+(0.6*t)

#seasonal indices as given in table 1
si<- c(0.75,0.80,0.82,0.90,0.94,0.92,0.91,0.99,0.95,1.02,1.20,1.80)

#multiplying the seasonal and trend component
mx<- si*T
rmse.mw <-c()
library(neuralnet)

for (i in c(1:100)){
```

```
yt1 <- mx+rnorm(228)
initial.data <- yt1[1:216]
########$%#$%^$@%@$^$@^@^@$$^$^@$%$##%^$#^
mean.initial.data <- mean(initial.data)
sd.initial.data <- sd(initial.data)
scale.initial.data <- scale(initial.data)
scale.initial.data <- c(scale.initial.data,0)
embed.scale.initial.data <- embed(scale.initial.data,37)
colnames(embed.scale.initial.data)<- c("original", "lag1", "lag2", "lag3", "lag4", "lag5", "lag6", "lag7", "lag8", "lag9", "lag10", "lag11", "lag12", "lag13", "lag14", "lag15", "lag16", "lag17", "lag18", "lag19", "lag20", "lag21", "lag22", "lag23", "lag24", "lag25","lag26", "lag27", "lag28", "lag29", "lag30", "lag31", "lag32", "lag33", "lag34", "lag35", "lag36")

#%$^#%@$^$##@#%#$#%#@^#@^@#%@$%$^%&^%$&*^%^
NN38<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24, data = embed.scale.initial.data, hidden = 8, linear.output = T)

for (q in c(1:12)){
  mean.initial.data <- mean(initial.data)
  sd.initial.data <- sd(initial.data)
  scale.initial.data <- scale(initial.data)
  scale.initial.data <- c(scale.initial.data,0)
  embed.scale.initial.data <- embed(scale.initial.data,37)
  colnames(embed.scale.initial.data)<- c("original", "lag1", "lag2", "lag3", "lag4", "lag5", "lag6", "lag7", "lag8", "lag9", "lag10", "lag11", "lag12", "lag13", "lag14", "lag15", "lag16", "lag17", "lag18", "lag19", "lag20", "lag21", "lag22", "lag23", "lag24", "lag25","lag26", "lag27", "lag28", "lag29", "lag30", "lag31", "lag32", "lag33", "lag34", "lag35", "lag36")

  #NN38<- neuralnet(original ~ lag1 + lag2 +lag3 +lag4 +lag12 + lag13 +lag14 + lag24, data = embed.scale.initial.data, hidden = 8, linear.output = T)
  #x_38 <- embed.scale.initial.data[181,c("lag1", "lag2", "lag3", "lag4", "lag12", "lag13", "lag14","lag24")]
  x_38 <- embed.scale.initial.data[dim(embed.scale.initial.data)[1],c("lag1", "lag2", "lag3", "lag4", "lag12", "lag13", "lag14","lag24")]
  x_38_mat <- matrix(NA, nrow = 1, ncol = 8)
  x_38_mat[1,] <- x_38
  scale.predict <- compute(NN38,x_38_mat)$net.result
  rescaled.predict <- (scale.predict*sd.initial.data)+mean.initial.data
  initial.data <- c(initial.data,rescaled.predict)
```

```
  }
  plot.original <- yt1[217:228]
  plot.predicted <- initial.data[217:228]
  final_result.mw <- data.frame(actual = plot.original, prediction = plot.predicted)
  rmse.mw[i] <- sqrt(mean((final_result.mw$actual - final_result.mw$prediction)^2))
}
#plot of ann and original
plot( plot.original, type = "b", frame = FALSE, pch = 19,
      col =  "red", xlab = "Time (monthly data)",ylab = "y",ylim = c(90,400))
# Add a second line
lines( plot.predicted, pch = 18, col = "blue", type = "b", lty = 2)
# Add a legend to the plot
legend("topleft", legend=c("original values", "prediction by ANN"),
       col=c("red", "blue"), lty = 1:2, cex=0.8)

avg.rmse.final.ann <- mean(rmse.mw)
#############################################################################
#**USING RANDOM FOREST **

training_setR<- df_lag[1:168,] #the embed() function has reduced the size of the data.
validation_setR<- df_lag[169:180,]
testing_setR<- df_lag[181:192,]
library(randomForest)
#Automating the process of selecting the best mtry
#1. Tuning the training dataset to get the best mtry
tuning.RF<- tuneRF(training_setR[,-1], training_setR[,1], stepFactor = 0.5, plot = TRUE, ntreeTry = 300, trace
=TRUE, improve = 0.05)

#Storing this result as a file
write.table(tuning.RF, file = "TuneMtry.csv", row.names = F, sep = ",")

#Reading this table
TuneMtry <- read.csv("TuneMtry.csv", header = T)

#Selecting the best mtry from the result
library(sqldf)
```

```
trym<- sqldf("SELECT mtry FROM TuneMtry WHERE OOBError IN (SELECT MIN(OOBError) FROM TuneMtry)")

#Converting the result into "numeric" datatype as it is stored as a "data.frame"
trym <- as.numeric(trym)


library(randomForest)

rmse.vector <- c()
a <- "original ~ lag1"
b<-as.formula(a)
rf <- randomForest(b, data = training_setR, mtry = trym, importance = TRUE)
predict.validation <- predict(rf, validation_setR)
rmse_r <- sqrt(mean((validation_setR$original - predict.validation)^2))
rmse.vector <- c(rmse.vector,rmse_r)
lags.paste <- c( "+lag2", "+lag3", "+lag4", "+lag12", "+lag13", "+lag14", "+lag24", "+lag25", "+lag36")
for (k in lags.paste) {
  a <- paste(a, k)
  b<-as.formula(a)
  rf <- randomForest(b, data = training_setR, mtry = trym, importance = TRUE)
  predict.validation <- predict(rf, validation_setR)
  rmse_r <- sqrt(mean((validation_setR$original - predict.validation)^2))
  rmse.vector <- c(rmse.vector,rmse_r)
}
which.min(rmse.vector)

best.model <- randomForest(original ~ lag1 + lag2 + lag3 +lag4 +lag12 + lag13 +lag14, data = training_setR,
mtry = trym, importance = TRUE)
predict.test <- predict(best.model, testing_setR)
rmse_final <- sqrt(mean((testing_setR$original - predict.test)^2))

# Performing 100 iterations to compute average RMSE from random forest

vec_rmse_rf<-c()
for (j in c(1:100)){
  e1<- rnorm(228, mean = 0, sd = 1)
  yt1<- mx+e1
```

```
  ts1<- ts(data = yt1, frequency = 12)
  lag <- embed(yt1, 37) #creating lags
  df_lag<- data.frame(lag) #converting matrix to data frame.
  colnames(df_lag)<- c("original", "lag1", "lag2", "lag3", "lag4", "lag5", "lag6", "lag7", "lag8", "lag9", "lag10",
"lag11", "lag12", "lag13", "lag14", "lag15", "lag16", "lag17", "lag18", "lag19", "lag20", "lag21", "lag22", "lag23",
"lag24", "lag25","lag26", "lag27", "lag28", "lag29", "lag30", "lag31", "lag32", "lag33", "lag34", "lag35", "lag36")
  training_set<- df_lag[1:168,] #the embed() function has reduced the size of the data.
  validation_set<- df_lag[169:180,]
  testing_set<- df_lag[181:192,]

  training_set <-as.data.frame(training_set)
  testing_set <-as.data.frame(testing_set)
  ############

  best.model <- randomForest(original ~ lag1 + lag2 + lag3 +lag4 +lag12 + lag13 +lag14, data = training_set,
mtry = trym, importance = TRUE)
  predict.test <- predict(best.model, testing_set)
  rmse_final <- sqrt(mean((testing_set$original - predict.test)^2))
  vec_rmse_rf[j]<-rmse_final
}
avg.rmse.final.rf <- mean(vec_rmse_rf)
# Plot of random forest and original
plot( testing_setR$original, type = "b", frame = FALSE,pch = 19,
      col =  "red",xlab = "Time (monthly data)", ylab = "y")
 # Add a second line
lines( predict.test, pch = 18, col = "blue", type = "b", lty = 2)
 # Add a legend to the plot
legend("topleft", legend=c("original values", "prediction by random forest"),
                col=c("red", "blue"), lty = 1:2, cex=0.8)
```