Forecasting Using Neural Networks

Mihail Mihaylov

Reg. No: 201888925

Introduction

This assignment involves training a neural network to make predictions for a time series given a set of values in a time series object. The object will be presented by a data set which would have to be prepared and then turned into a lagged matrix which would be passed onto the neural network and every row from the matrix would correspond to a set of values which would be used to predict the next one (which would be held in the last column of the matrix). The matrix will be split into two parts (training and testing parts). The training set is used for training the neural network and the testing set would be used to see how the neural network works on future unseen data. More details about the implementation can be seen below.

Background to the data set(s)

The data set chosen to do this assignment is the monthly log stock returns for Disney. The data is taken from: http://faculty.chicagobooth.edu/ruey.tsay/teaching/fts/m-dis6299.dat This data set contains 456 monthly values, which have been cast into a time series object with a frequency of 12. A plot of the data can be seen in figure 1 below.

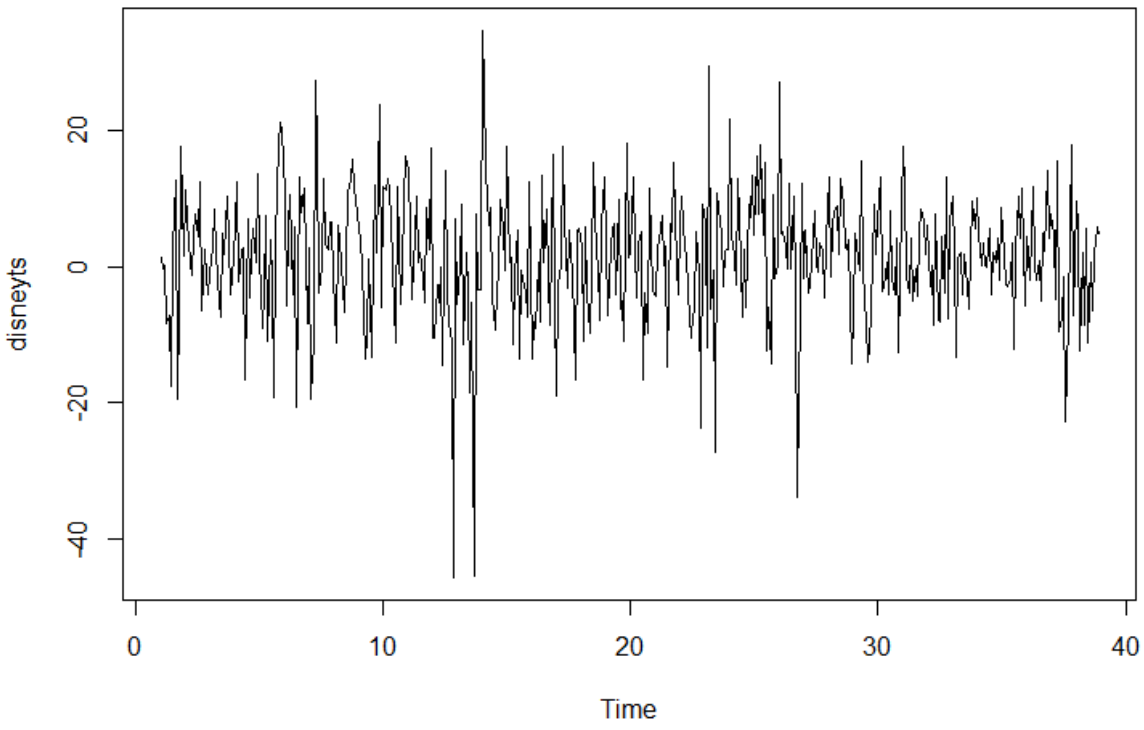

Figure 1: Time series plot of the original data

As it can be seen from the figure above the stock returns variate quite a lot through time so it would be difficult to build a model to generalize well and make good predictions over the time series. In order to prepare the data for the neural network the time series would need to be smoothed. To do that a built-in function in R called SMA was used. This method takes as an input the time series and an integer which points to the number of periods to average over. This function would calculate the arithmetic mean of the series over the past 12 observations. Once the smoothing is done and the null values have been removed there are 445 values left in the time series. Figure 2 below shows a plot of the original time series against the smoothed time series.

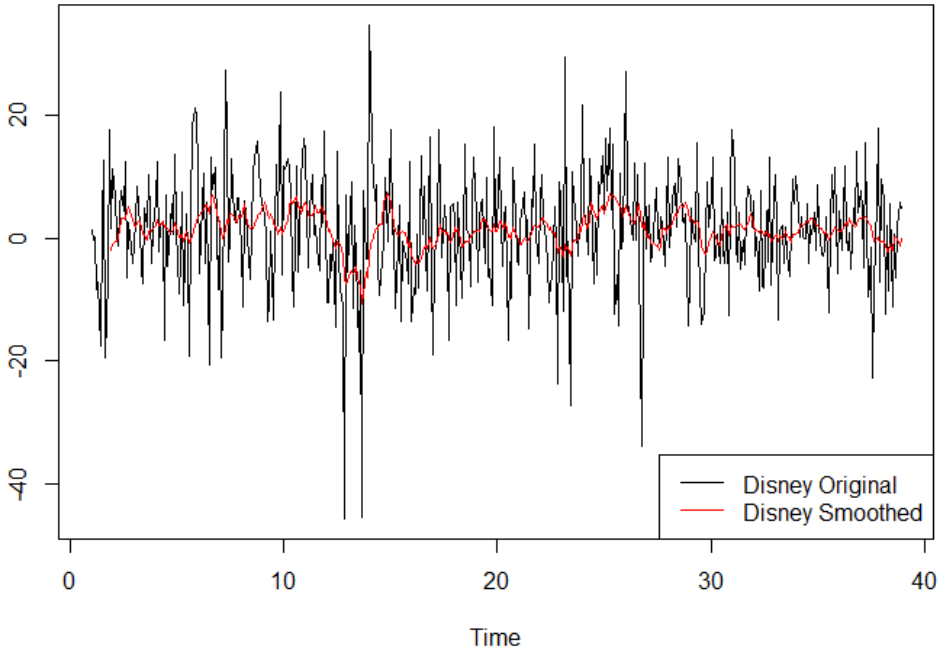

Figure 2: Time series plot of the original and the smoothed values.

As it can be seen from the figure above the smoothed stock levels are a lot less variate and should do well to perform forecasting on them. The next step in the process is to create a lagged matrix from the smoothed time series and build the neural network, details of which are described in the next section of this report.

Details of the neural network and results from different runs

Once the data has been prepared it is time to create the lagged matrix. A lagged matrix is used to split the data into different consecutive windows. A part of the matrix can be seen in Figure 3 below.

Input1	Input2	Input3	Input4	Input5	Input6	Input7	Input8	Input9	Input10	Output
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	-1.924667
NA	NA	NA	NA	NA	NA	NA	NA	NA	-1.924667	-1.092083
NA	NA	NA	NA	NA	NA	NA	NA	-1.924667	-1.092083	-0.489333
NA	NA	NA	NA	NA	NA	NA	-1.924667	-1.092083	-0.489333	-0.456250
NA	NA	NA	NA	NA	NA	-1.924667	-1.092083	-0.489333	-0.456250	0.148333
NA	NA	NA	NA	NA	-1.924667	-1.092083	-0.489333	-0.456250	0.148333	1.110000
NA	NA	NA	NA	-1.924667	-1.092083	-0.489333	-0.456250	0.148333	1.110000	3.224583
NA	NA	NA	-1.924667	-1.092083	-0.489333	-0.456250	0.148333	1.110000	3.224583	3.371833
NA	NA	-1.924667	-1.092083	-0.489333	-0.456250	0.148333	1.110000	3.224583	3.371833	3.346500
NA	-1.924667	-1.092083	-0.489333	-0.456250	0.148333	1.110000	3.224583	3.371833	3.346500	4.434333
-1.924667	-1.092083	-0.489333	-0.456250	0.148333	1.110000	3.224583	3.371833	3.346500	4.434333	5.201250
-1.092083	-0.489333	-0.456250	0.148333	1.110000	3.224583	3.371833	3.346500	4.434333	5.201250	3.635167
-0.489333	-0.456250	0.148333	1.110000	3.224583	3.371833	3.346500	4.434333	5.201250	3.635167	3.163750
-0.456250	0.148333	1.110000	3.224583	3.371833	3.346500	4.434333	5.201250	3.635167	3.163750	2.139833
0.148333	1.110000	3.224583	3.371833	3.346500	4.434333	5.201250	3.635167	3.163750	2.139833	1.801083
1.110000	3.224583	3.371833	3.346500	4.434333	5.201250	3.635167	3.163750	2.139833	1.801083	2.459500
3.224583	3.371833	3.346500	4.434333	5.201250	3.635167	3.163750	2.139833	1.801083	2.459500	2.717250
3.371833	3.346500	4.434333	5.201250	3.635167	3.163750	2.139833	1.801083	2.459500	2.717250	1.921000
3.346500	4.434333	5.201250	3.635167	3.163750	2.139833	1.801083	2.459500	2.717250	1.921000	0.651417

Figure 3: First 19 rows from the lagged matrix with 10 inputs.

The figure above shows the first 19 lines of the lagged matrix with a window size of 11. That is 10 inputs and the output. In this case the time series from the stock level for Disney have been lagged 10 times in order to create the matrix and the first line with no null values holds the 1st to the 11th numbers from the data set. The next line holds 2nd to the 12th values and so on. Now that the matrix has been created the rows with null values need to be omitted because the neural network can not work with missing values. Once this is done the matrix is split into 2 sets a training and a testing set. The first 80% of the matrix has been saved as a training set to be used in the neural network to build a model and the remaining 20% has been saved as the testing set to test if the model build works well on future unseen data. Since forecasting involves dealing with time series the split is done in consecutive rows and not random sampling. The training set in this case has a size of 348 rows and the testing set is the remaining 87 rows of data.

Once the data has been split it is time to build the neural network. In R there are multiple packages that can do that. The package chosen is called neuralnet and the function for neural networks is called the same way. The function takes numerous arguments as an input. Some of those are pretty straightforward and were determined from the start others required multiple runs in order to fine tune the model to work well with the current data set. The first argument is the formula which consists of a symbolic description of the model to be fitted. In this case it is Output ~ Input1 + Input2 + Input3 + Input4 + Input5 + Input6 + Input7 + Input8 + Input9 + Input10. The number of inputs summed depends on the number of inputs that are put into the neural network. For this assignment the chosen number of inputs was 10. Many different runs were performed with 6, 8, 10 and 12 inputs, the results from which can be seen in Table 1. The next argument for the neuralnet function is the data in this case that is the training set split from the lagged matrix. After that it is important to specify the number of hidden layers and the number of neurons in each layer. The best one was determined to be 3 layers of 15 neurons in each. This can also be seen in Table 1. Other arguments for the neuralnet function involve the algorithm (different runs were performed with resilient backpropagation with and without weight backtracking, results for which can be seen from Table 1) which was chosen to be rprop+ which is the resilient backpropagation with weight backtracking, the stepmax number which is the maximum steps for the training of the neural network before stopping the training process. The stepmax value chosen was 1e+06. The default value is 1e+05 which turned out to be too little in some cases for the neural network to converge to a good solution and another value considered was 1e+07 which proved to take too long in some cases to run and did not provide much better solutions. Another argument for the function is the threshold which holds a numeric value specifying the threshold for the partial derivatives of the error function as stopping criteria. The values considered were 0.01 and 0.1, where 0.1 was the chosen one because 0.01 caused the model to overfit and underperform on the testing set.

Run #	Algorithm	Inputs	Layers	MSE Training	MSE Testing
1.	rprop-	10	15,15,15	0.18114	1.649
2.	rprop-	10	15,15,15,15	0.01653	1.319
3.	rprop-	10	15,15	0.95192	2.71
4.	rprop-	10	10,10	30.3405	1.505
5.	rprop-	10	15,10	3.00772	3.38
6.	rprop+	10	15,15,15	0.28175	1.012
7.	rprop+	10	15,15,15,15	0.21907	1.379
8.	rprop+	10	15,15	1.33923	1.216
9.	rprop+	10	15,10	8.58322	1.135

10.	rprop+	10	10,10	17.04644	1.303
11.	rprop+	10	10,10,10	dnc	dnc
12.	rprop+	10	15,10,10	0.45702	1.392
13.	rprop+	10	15,15,10	0.33218	1.589
14.	rprop+	10	15,10,5	0.96801	1.316
15.	rprop+	10	20,10,5	0.8687	1.278
16.	rprop+	10	25,15,10	0.1982	1.384
17.	rprop+	10	20,15,10	0.47457	1.384
18.	rprop+	12	15,15,15	0.54819	1.252
19.	rprop+	12	25,15,10	0.3407	1.073
20.	rprop+	12	25,15,15	0.19288	1.267
21.	rprop+	12	20,15,15	0.15424	1.193
22.	rprop+	8	15,15,15	0.3938	1.396
23.	rprop+	8	15,10,10	2.25308	1.872
24.	rprop+	8	15,15,10	0.19136	1.466

Table 1: Table of the results from fine tuning the neural network.

In order to determine the algorithm, the number of inputs and the layers and neurons of the neural network it was run 24 times with different values. The best ones were chosen based on the mean squared error for the training and the testing set. The best values can be seen on line 6 which was performed using the rprop+ algorithm with 10 inputs and 3 hidden layers with 15 neurons in each. The results obtained were a MSE of 0.28175 for the training set and 1.012 for the testing set.

Comparing the rprop+ and rprop-, it can be seen that the runs with the same number of inputs and the same number of layers and neurons in the network provided better results when using the resilient backpropagation with weight backtracking. Once that was chosen different runs were performed with 10 inputs and different number of layers and neurons. As it can be seen on line 7 the results for the training set when using 4 layers of 15 neurons is better than with 3 layers with 15 neurons each, but the testing set results are worse which would mean that the model overfits the training data more and overall it is not a better solution. The key to finding a good model in this case requires a balance between the mean squared error from the training and the testing set which was best at the neural network using 10 inputs and 3 layers of 15 neurons. Lines 18 to 21 of the table show the runs performed with 12 inputs. In some cases the results on the training set are better at the expense at the results from the testing set. The same applies for 8 inputs which proved to be not enough to find a good solution.

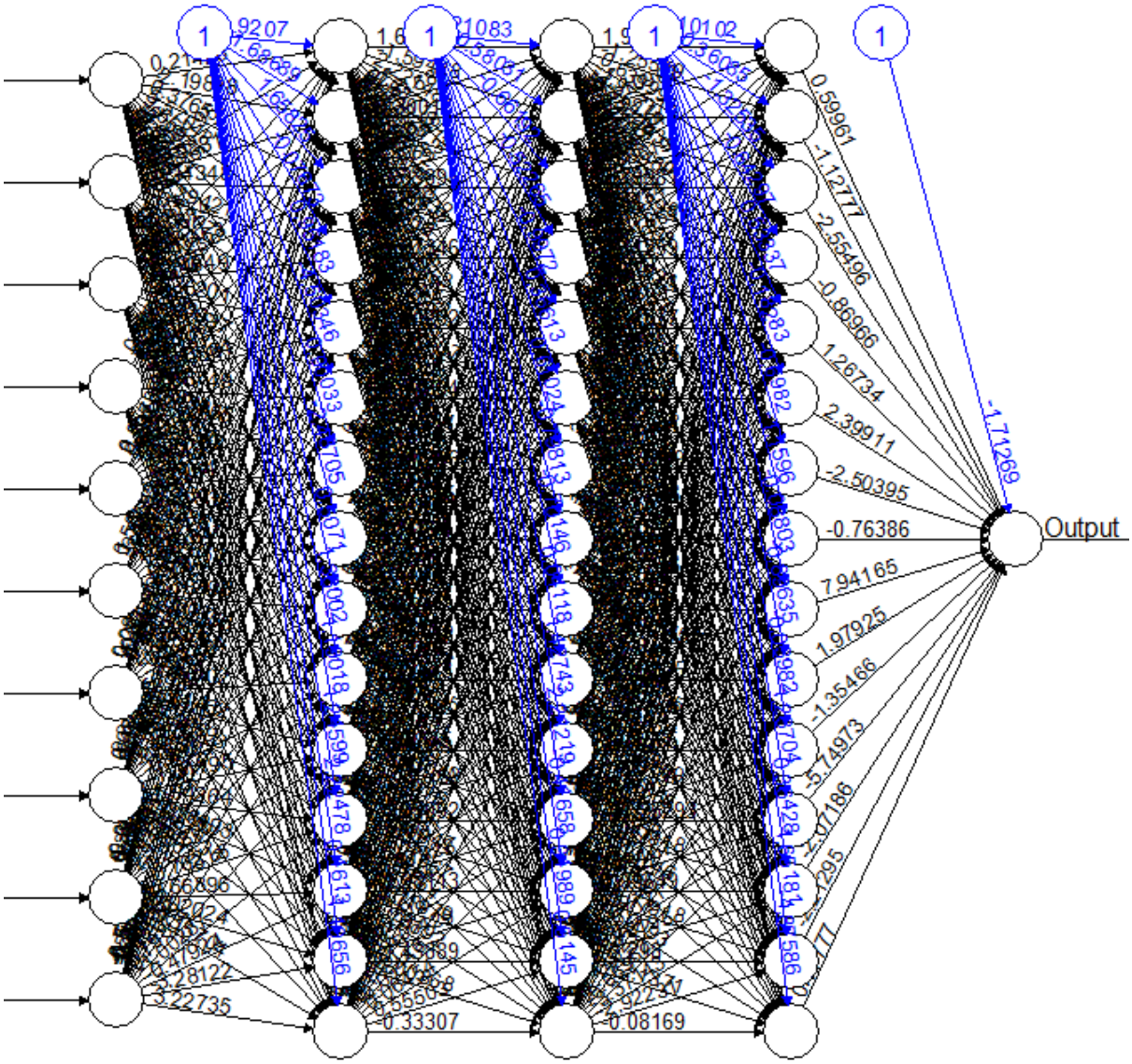

Figure 4: Neural network with 10 inputs and 3 hidden layers of 15 neurons.

The figure above shows what the neural network looks like with 10 inputs, 3 hidden layers and 15 neurons in each layer. That was the chosen setup, yielding the best results for the chosen data set.

The predictive abilities of the neural network over the test (unseen) data

Once the neural network has been configured and set to the desired values it was run again to provide better visualisation of the results and the model was used to compute the values for the test data and the whole dataset. Figure 5 below shows a plot of the smoothed data and the predictions of the neural network for the test set. As it can be seen in the figure the predictions are not exactly the same as the actual values for the test set but there are a lot of similarities. The main point in this is to show that the predictions are in the same direction as the actual values, which means that this model can be used to estimate the direction that the stock levels are going to take in the future. After obtaining the expected values they were compared against the actual values for the test set and the mean squared error was calculated. The mean squared error in this case is: 1.012

The figure below also shows that the differences between the actual values and the predicted values is rarely more than 1, which also proves that the model is accurate enough to make predictions of the direction the stock prices are going to go.

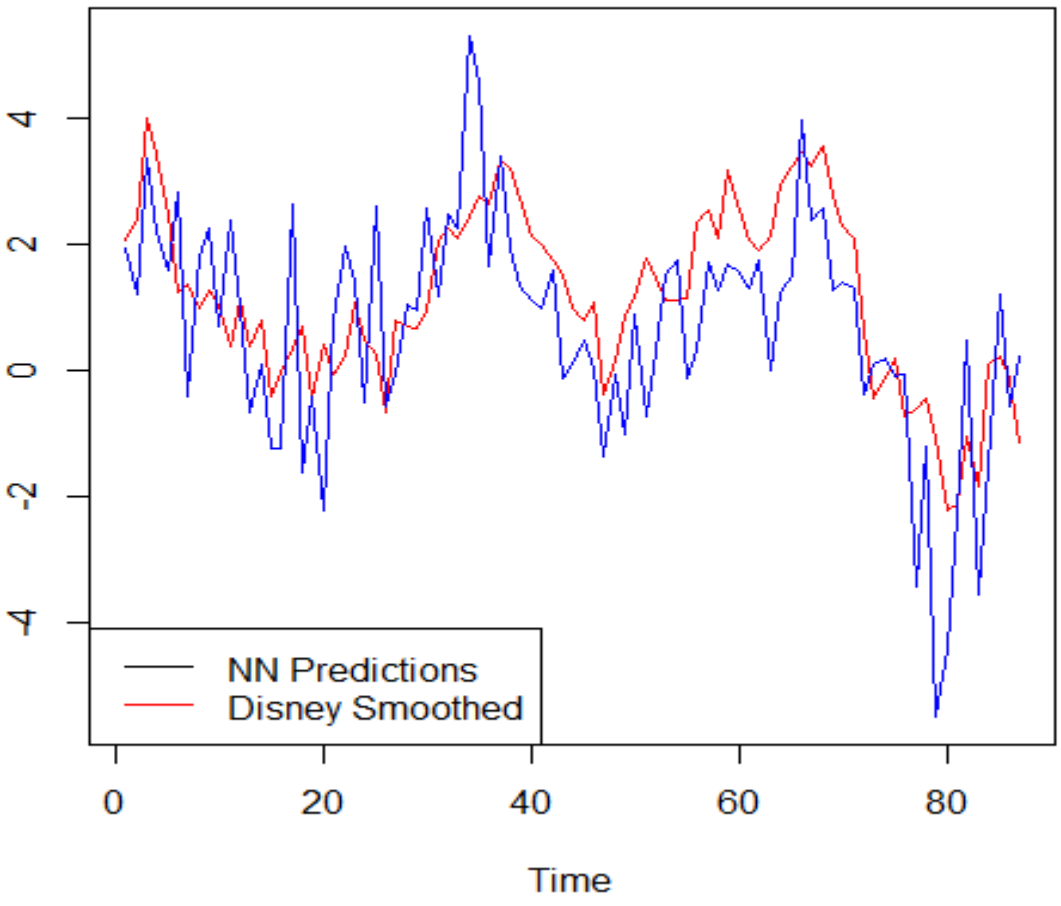

Figure 5: Plot of the predicted values from the neural network versus the actual values from the test set.

The next section of the report show the comparison of the network performance with other approaches. It also contains a plot of the actual values, predicted values from the neural network and other predicted values from other methods.

Comparisons of the network performance with other approaches

The comparisons performed for this assignment is between the neural network predictions for the test set and the predictions for the same set using the Holt-Winters technique and baseline performance of means.

The best way to compare the different techniques for forecasting would be to take a look at the plot provided below. It holds the plots for the actual values for the test set (blue), with the neural network predictions (red), the baseline mean predictions (green) and the Holt-Winters predictions (yellow). As it can be seen from the figure below the base mean line lags behind the actual values

because of the means being calculated on previous values. The Holt-Winters technique provides an exponential prediction based on the values from the training set and only the neural network tries to calculate the actual values using a model trained on previous data. From those 3 techniques the neural network is the only one which could indicate that the stock value would go up before it actually does that rather than after like the base mean.

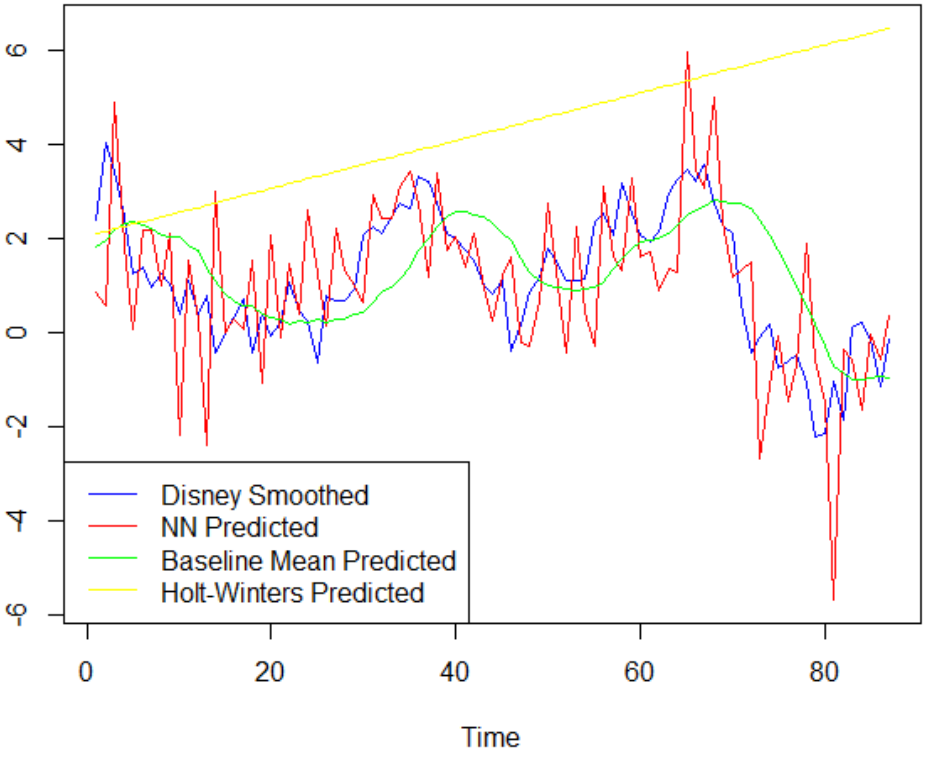

Figure 6: Multi plot over test set of NN, HW, Base Mean predictions and actual values.

The next figure below shows a sample from the 3 techniques which are being compared and the actual values from the test set. That is used to show the similarities between the actual values and the neural network which lack in any of the other 2 techniques.

	Disney Smoothed	NN Predicted	Mean Predicted	HW Predicted
1	2.39933	0.843676	1.8366	2.102
2	4.01958	0.563504	1.9713	2.153
3	3.43692	4.892114	2.1752	2.204
4	2.46508	1.870852	2.3307	2.255
5	1.24192	0.077143	2.3706	2.305
6	1.38142	2.195985	2.2935	2.356
7	0.97783	2.176466	2.2017	2.407
8	1.26733	0.996921	2.0787	2.458
9	1.04958	2.118876	2.0540	2.508
10	0.38758	-2.182905	2.0291	2.559
11	1.12233	1.526777	1.8627	2.610
12	0.38525	0.301570	1.7350	2.661
13	0.78225	-2.403503	1.3715	2.711
14	-0.42142	2.984781	1.1061	2.762
15	-0.02458	-0.004989	0.8174	2.813
16	0.34142	0.288277	0.6908	2.864
17	0.70417	0.062435	0.5868	2.914
18	-0.43208	1.529031	0.5594	2.965
19	0.40217	-1.072951	0.3895	3.016
20	-0.06742	2.070370	0.3247	3.067

Figure 7: Sample results from the test set with NN, HW, Base Mean predictions and actual values.

By this it can be concluded that the neural network is better at predicting values than Holt-Winters and the base mean predictions.