# Assignment 3: Forecasting and Prediction

David Smith

February 2019

## 1 Introduction

This project involved the use of neural networks for the application of forecasting and predicting financial data. The project was concerned with data available from a web page for the Analysis of Financial Time Series, which consisted of monthly returns on The Hershey Company stock between 1977 and 2015.

The entry for each month was a percentage difference representing the return on investment on the stock that month. We can see the variation of returns over the 38 years below.

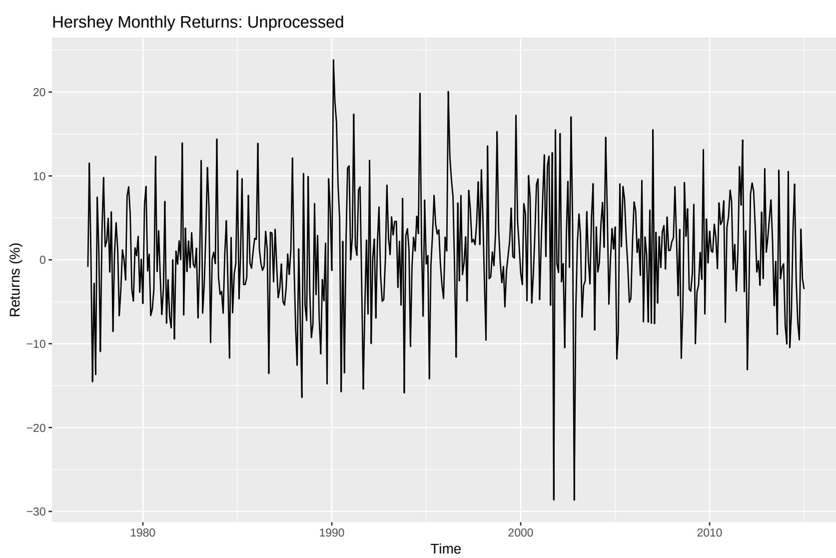


Figure 1: Pre-processed monthly returns on Hershey stock between 1997 and 2015.

The dataset was chosen for the task as it was fairly large and covered an extensive time span. The dataset size meant processing and cleaning could be carried out whilst still maintaining data integrity for analysis. As can be seen above, the data was noisy, contained many spikes and had no distinct trends. Therefore, it was decided to subset the data down to a period of 30 years and to perform some smoothing. Smoothing would decrease the influence of the spikes whilst still maintaining data integrity. This was done by taking the average of every five monthly returns for the 30 year period.

Neural networks are highly sensitive to the scale of their inputs, therefore scaling of the monthly returns was required. Normalization was carried out to re-scale the returns between 0 and 1. The processed data can be seen below.

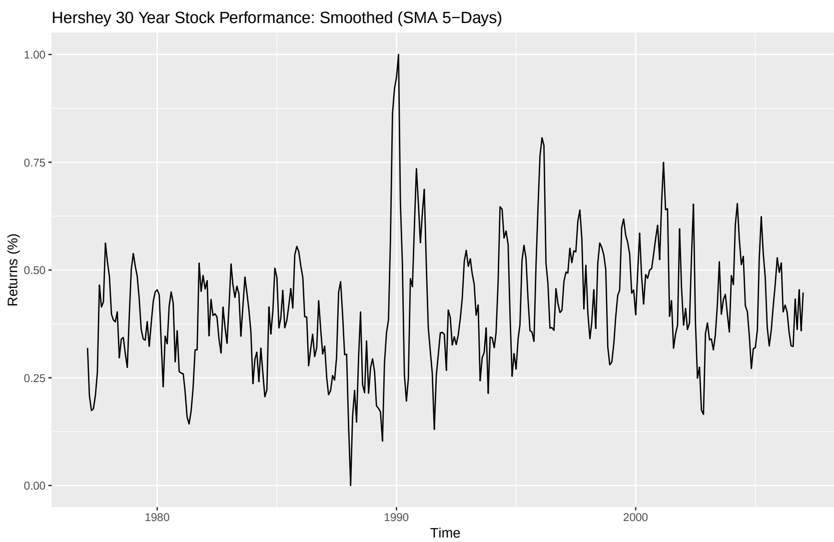


Figure 2: Processed monthly returns on Hershey stock between 1997 and 2007.

After processing, we can now see a slight upwards trend and a large spike around 1990. It is hard to detect any substantial seasonality in the data, therefore decomposition of the data was carried out. This is a statistical task which deconstructs a time series into components representing underlying categories of patterns. The decomposition of the monthly returns can be seen below.

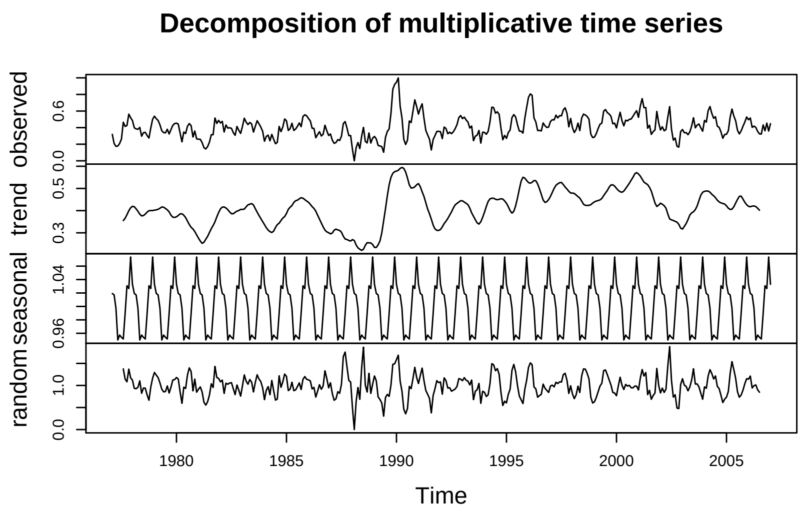


Figure 3: Observed, trend, seasonal and random components to Hershey monthly returns time series.

The decomposition demonstrates more clearly the observable patterns in the time series. The observed and trend components indicate the slight upwards movement of the stock across the period as well as the spike in return between approximately 1988 and 1992. The seasonal component shows a small spike, ranging between 0.96 and 1.04%, telling us the stock maintains a positive return. The random component is fairly substantial throughout, indicating high degrees of randomness in the data.

To prepare our data for further analysis, we split the data into training and testing subsets. The magnitude of the testing subset would determine how far into the future we were aiming to predict. Due to the time span of the data, this was set initially as 10 years. The split between training and testing data can be seen below.

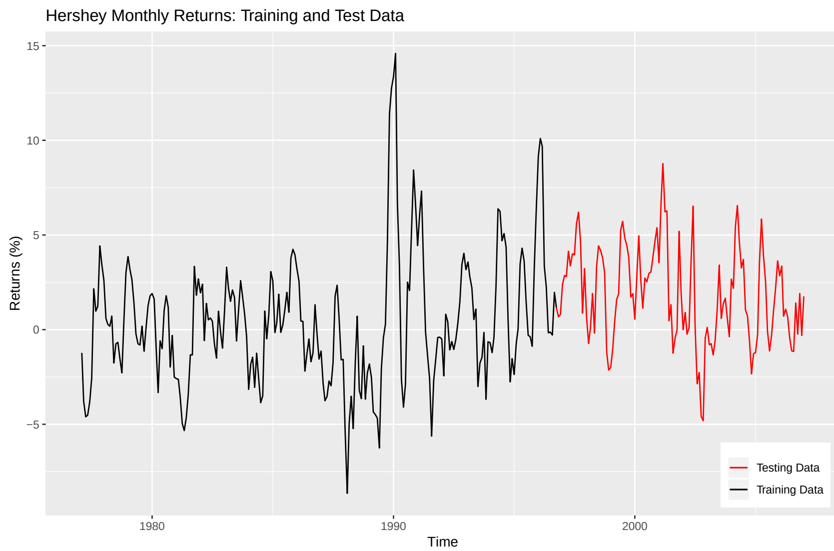


Figure 4: Dividing the data: 20 years worth of returns for training to produce a 10 year forecast.

The model would be trained over the monthly returns between 1977 and 1997 to provide an initial forecast of 10 years. It can be seen there is no strong trend over this period, with many peaks and troughs. This could prove difficult to predict, therefore it was decided to experiment with varying forecast sizes. However this would also affect the amount of data the neural network would be training on, so a trade-off must be reached.

# 2 Representation

Before implementing the neural network, the time series data was transformed to be more suitable for analysis. A lagged matrix was created, where each matrix row consisted of a monthly return and a number of previous returns. Moving down one row in the matrix meant shifting forward in time by one entry. An example excerpt of the lagged matrix can be seen below.

| t | Return | Return (t-1) | Return (t-2) |
|---|---|---|---|
| 2005-01 | 0.04 | N/A | N/A |
| 2005-02 | 0.07 | 0.04 | N/A |
| 2005-03 | -0.03 | 0.07 | 0.04 |
| 2005-04 | 0.80 | -0.03 | 0.07 |
| 2005-05 | -0.50 | 0.80 | -0.03 |

Table 1: Example lagged matrix with a window size of 2.

As can be seen above, the first entry has no available previous data entries. With each progressing month, the previous return(s) are then available to use as inputs. We were aiming to use the neural network to give us predictions for the Return column, based on an aggregation of that row's previous returns. The window size, i.e. how many previous returns would be considered each time, was variable and would be altered to observe its effect on prediction.

# 3 Neural Network Package

The neural network was implemented using the package *neuralnet*. The package allowed for high customization of the model hyperparameters. For this project the number of hidden layers, including the number of nodes per layer, the step max function and error threshold would be considered, as well as the window size of the inputs and the range of the prediction, i.e. the forecast size.

With the input transformed into a lagged matrix, we could begin experimenting with different network configurations. Performance would be evaluated based on the root mean squared error (RMSE), calculated between the actual monthly returns and the networks predictions. The RMSE was a suitable metric as it gives more weighting to larger errors, appropriate given the financial context of the network's predictions.

# 4 Results & Configurations

| Forecast (yr) | Norm. | Window | Hidden Layers | Train RMSE | Test RMSE |
|---|---|---|---|---|---|
| 10 | N | 10 | 10,10,10 | 0.0075 | 0.3013 |
| 10 | Y | 10 | 10,10,10 | 0.0387 | 0.2524 |
| 5 | Y | 10 | 10,10,10 | 0.0340 | 0.1582 |
| 3 | Y | 10 | 10,10,10 | 0.0589 | 0.2031 |
| 1 | Y | 10 | 10,10,10 | 0.1184 | 0.2167 |
| 10 | Y | 10 | 10,10,10,10 | 0.0672 | 0.2996 |
| 10 | Y | 10 | 10,10,10,10,10 | 0.0657 | 0.1423 |
| 10 | Y | 10 | 10,10,10,10,10,10 | 0.0517 | 0.2145 |
| 10 | Y | 10 | 90, 30, 10 | 0.0172 | 0.3363 |
| 10 | Y | 10 | 80, 60, 20 | 0.0142 | 0.2386 |
| 10 | Y | 10 | 20,10,5 | 0.0122 | 0.3383 |
| 10 | Y | 5 | 90,30,10 | 0.0499 | 0.1024 |
| 10 | Y | 5 | 90,30,10,10 | 0.0882 | 0.1799 |
| 10 | Y | 2 | 90,30,10,10 | 0.0860 | 0.2916 |

Table 2: Range of neural network configurations explored.

The network was initially run with both scaled and unscaled data to determine the effect of normalization. We observed a lower error on both training and testing data as a result of normalization, hence the data remained scaled for subsequent models. The forecast size was initially set to 10 years, which seemed an appropriate long-term forecast for a dataset of this size. When decreasing the forecast size, i.e. increasing the train/test ratio, we saw a reduced error on both train and test data. However the error increased once this was decreased to a 1 year prediction, indicating that the network was over-fitted.

Next, the number of layers was increased, which decreased errors up until there were six hidden layers, after which the error increased. As well as increasing the time taken to converge, it is likely that overfitting also occurred. Similarly with the number of nodes, an increase gave higher accuracies but dropped after approximately 100 nodes per layer. Altering the layer number and node count mainly affected the testing error, demonstrating the susceptibility of the network to overfitting if too many features are added.

The size of the lag window was then altered. An initial size of 10 had been set due to the size of the dataset. However after further consideration, with the dataset containing 450 entries, the size was altered. Halving the amount of inputs gave a greater accuracy. This indicated that 10 inputs was likely causing the network to over-fit.

Altering the threshold had no substantial effect on error and only increased time to convergence, so was left at a rate of 1%. Similarly, the step max function was left at the default of 100000 steps. With an error threshold of 1% the network consistently converged before the maximum number of steps was reached.

The final neural network configuration consisted of three hidden layers containing 90, 30 and 10 nodes respectively. This configuration provided the lowest error for testing and a reasonable training error. A lag window of 5 was chosen for a 10 year forecast. With this configuration training and testing RMSEs were 0.0499 and 0.1024 respectively.

## 4.1 Training Performance

We can visualise the performance of the final neural network when predicting on the training data below.

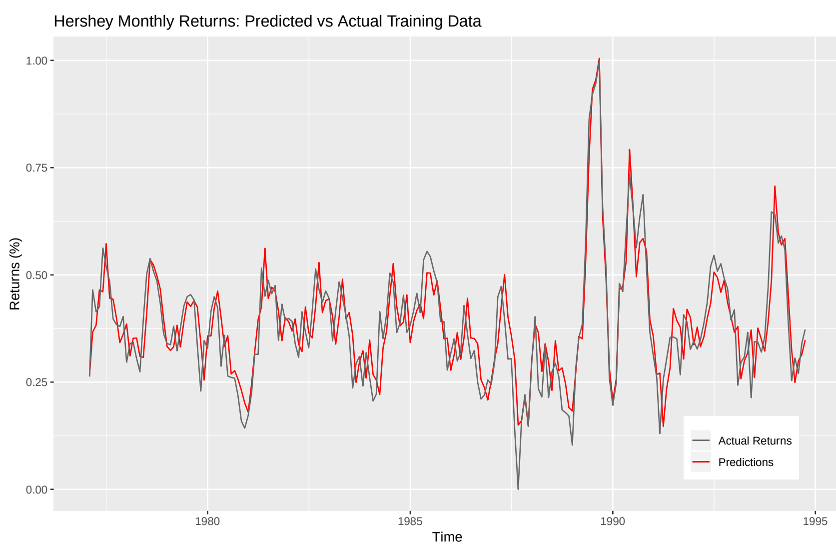


Figure 5: Neural network predictions over the seen training data.

The above figure shows the accuracy of the network when giving predictions for the training period. As the neural network was trained over this period, a high rate of accuracy is to be expected. The neural network exhibited an RMSE of 0.0499 over the training subset. It could be said overfitting is taking place, however it is hard to say until predictions on unseen data have been made. It is likely that displays of high accuracy such as these are usual when evaluating predictions made for data which the model has been trained on.

## 4.2 Predictive Performance

We can visualise the performance of the network when predicting over the unseen test period below.

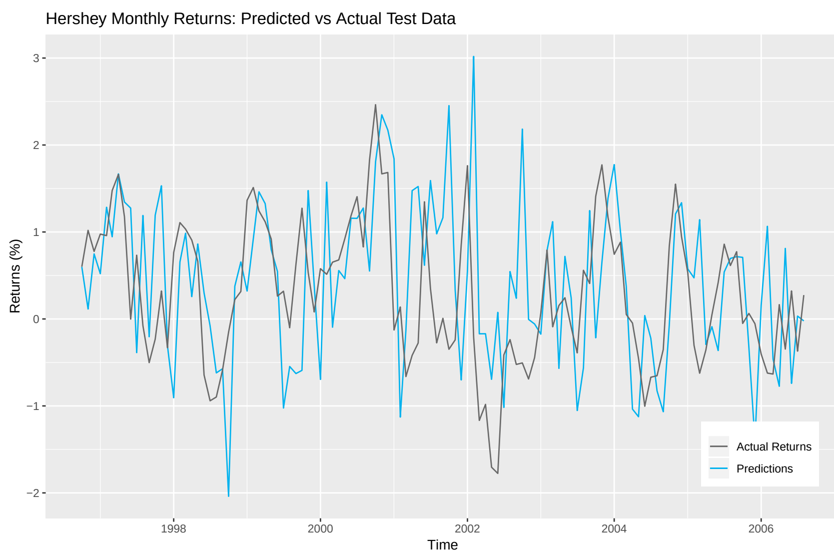


Figure 6: Neural network predictions over unseen testing data.

We can see a clear comparison between the predicted and actual returns above. The predictions are fairly erratic, with steep peaks and troughs above and below the line of the actual returns. However, the predictions are following the trends of the actual returns, largely mirroring their increases and decreases.

Comparing the performance with the training predictions, it could be said that the model has over-fit on the training data to a degree, leading to greater residuals when generalizing on the testing data. The training subset and/or lag window size may have been too large and contributed to over-fitting, however several configurations were experimented with. The final neural network achieved an RMSE of 0.0854 on the test data, which was the lowest error out of all explored configurations.

As shown in initial analysis of the data, there is a high degree of randomness in the data. There was no clear trend, and any trend we can currently see was a result of data smoothing. Therefore the erratic behaviour observable in the neural network predictions could be a result of the data having no clear patterns to model.

# 5 Comparison with Other Approaches

To evaluate the performance of the neural network against other methods, predictions were made using a Holt-Winters forecast and also a 5-Day Moving Average.

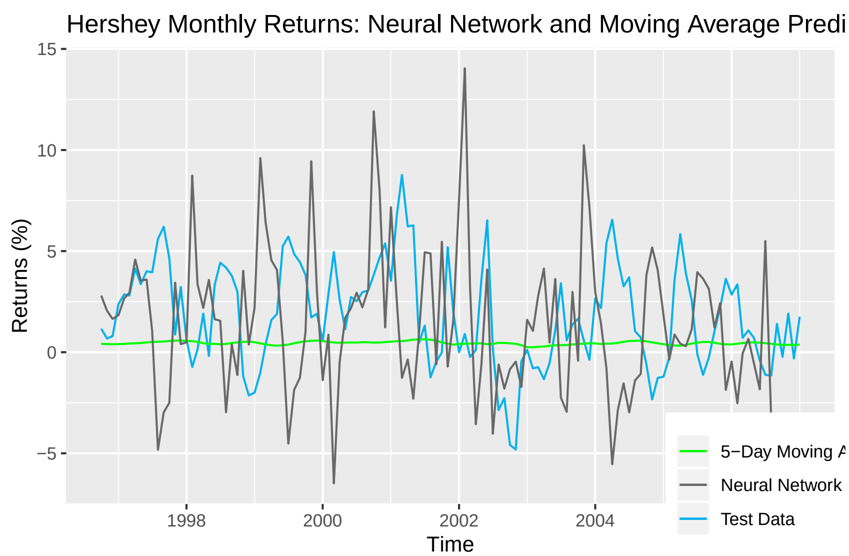


Figure 7: Neural network predictions across the testing data compared with a baseline mean measure.

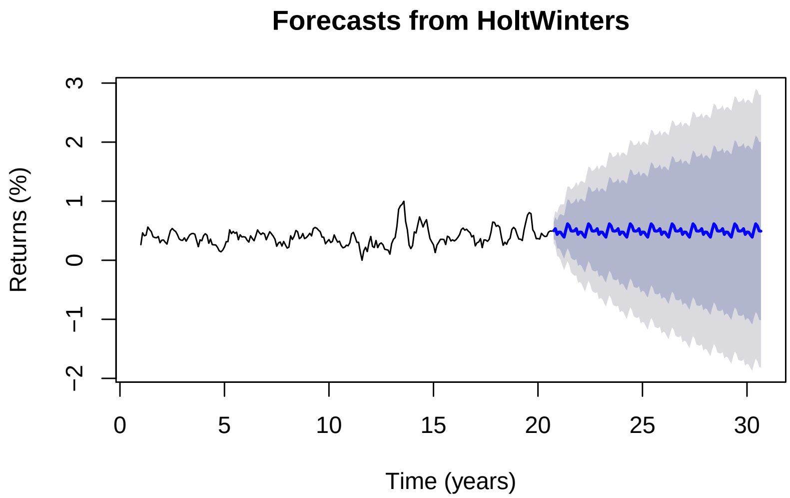


Figure 8: Holt-Winters Forecast over the testing period of 1997-2007, shown here in years. Please note: the forecast begins at approx. 20 years and is in blue. The surrounding buffers represent 85% and 95% likelihood zones respectively.

In Figure 7 we can see the actual test data, the neural network predictions and a 5-day Moving Average. The moving average was calculated from plotting the means of every row in the lagged matrix. We can see that as a predictive model it is weak, simply following the average of the returns and failing to indicate any real features in the data. The superior performance of the neural network is evident, as we can see it's predictions actively changing with the actual data.

Figure 8 shows us the result of the Holt-Winters forecast. The Holt-Winters forecasting method is an algorithm for predicting time-series points. It is hard to determine the shape of its prediction here, but worth noting is the seasonality of the prediction. Holt-Winters forecasts requires a fair degree of seasonality in the data it is forecasting. Whilst the monthly returns may contain some elements of seasonality, it is small at best whilst they contain high degrees of randomness. Therefore, the Holt-Winters may not be a suitable comparison. However, it does highlight the versatility of the neural network, which has no requirement of seasonality when predicting time series data.