Stock Market Prediction using Time Series Analysis and Neural Networks

Misati Nyambane - 670145

Bethlehem Kebede

Hoziyana Rachael

MTH1060: Analytical Computational Foundation

Dr. Francis Oketch

July 29, 2024

Abstract

Table of Contents

Methodology

Data Acquisition

Data was obtained from Yahoo Finance. The data spans from 1st July 2014 to 28th June 2024.

The data is in the general form:

Date,Open,High,Low,Close,Adj Close,Volume

2014-07-01,23.3799991607666,23.517499923706055,23.282499313354492,23.3799991607666,20.680427551269 53,152892000

2014-07-02,23.467500686645508,23.514999389648438,23.272499084472656,23.3700008392334,20.671592712402344,113860000

2014-07-03,23.417499542236328,23.524999618530273,23.299999237060547,23.50749969482422,20.793209075927734,91567200

The data was obtained using the Yahoo finance package in Python. By use of a dictionary the 20 companies selected were iterated through and their data written to CSV files. Yahoo Finance was preferred since it offers data for each company separately.

Data Preprocessing

Data was imported into MATLAB in bulk and stored in an array. This enables the use of iteration and control structures to access the data. Warnings are suppressed for cleaner output and if there are any missing values they are pointed out.

This allows for missing values to be dropped to ensure that operations are not skewed or curtailed by this data.

Furthermore, data is normalized between the range of 0 and 1 for our Time Series Analysis and Neural Network models to be able to work with our data. Only numerical columns are normalized. The column *date* is not affected in any way.

Machine Learning Models

Times Series Analysis

Neural Networks

We decided to use a Feedforward Neural Network that:

- Has 40 hidden nodes
- Trains for a maximum of 1000 epochs
- Uses the adam optimizer
- Utilizes 75% of the data to train
- Utilizes 25% of the data to test
- Uses the Mean Squared Error to test the error of the results
- Utilizes validation data from 1st July 2024 to 26th July 2024

The neural network uses different functions to compute the values. In simplified terms a neural network is a polynomial function whose parameters we do not know. That is, we do not know the degree, variables or coefficients. Nonetheless, what we do know are the weights and biases. These weights and biases are updated using polynomial functions that differentiated and integrated as needed. Furthermore, an activation function and delta variable are employed to control the rate at which the weights and biases are updated. This modification of weights and biases is how the model 'learns'.

Forward Propagation

For forward propagation the functions employed are:

$$h = f(W_1 x + b_1)$$

The function above computes a linear combination of the weights and biases then this linear combination is transformed using a non-linear function. This enables complex relationships to be learnt.

$$\hat{y} = g(W_2 h + b_2)$$

The function receives the output from the entry node and then computes the linear combination of weights and biases which are then transformed by the Sigmoid function (defined below).

$$f(z) = max(0, z)$$

This function is the Rectified Linear Unit (ReLU) activation function which outputs a value directly if it is positive. Otherwise, it outputs zero.

$$g(z) = \frac{1}{1 + e^{-z}}$$

This function is the Sigmoid function which maps any real-valued number into the range (0, 1).

Back Propagation

The function below is the loss function which calculates the Mean Squared Error (MSE) between the actual values and the predicted values. The aim is to minimize the value obtained from the loss function to get a more accurate neural network.

$$L = \frac{1}{2}\sum(y - \hat{y})^2$$

The function below is the derivative of the loss which indicates how much the loss changes with a small change in the predicted output. If $\hat{y}$ is greater than *y*, the derivative will be positive, suggesting that decreasing $\hat{y}$ will reduce the loss. Conversely, if $\hat{y}$ is less than *y*, the derivative will be negative, indicating that increasing $\hat{y}$ will help reduce the loss. This will inform the network on the way weights and biases should be updated.

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$$

This function is the derivative with respect to weights. This term $g'(W_2h + b_2)$ assesses the sensitivity of the output with respect to the input of the activation function. The product with *h* reflects how changes in the weights affect the output based on the current output from the first layer. This derivative is then employed in backpropagation to adjust the weights in the direction that reduces the loss.

$$\frac{\partial \hat{y}}{\partial W_2} = h \cdot g'(W_2h + b_2)$$

The last function is the derivative with respect to bias. Similar to the derivative with respect to weights function this one is used to adjust the bias in the direction that minimizes the loss.

$$\frac{\partial \hat{y}}{\partial b_2} = g'(W_2 h + b_2)$$

Gradients

The functions are the derivatives with respect to the hidden layer output, weights and bias. These derivatives are then used to update their respective variables. This backpropagation aims to reduce the error between the predicted and the actual values.

$$\frac{\partial L}{\partial h} = \left(\frac{\partial L}{\partial \hat{y}}\right) \cdot W_2$$

$$\frac{\partial h}{\partial W_1} = x \cdot f'(W_1 x + b_1)$$

$$\frac{\partial h}{\partial b_1} = f'(W_1 x + b_1)$$

Weight Update Rules

The weights are then updated using the following functions to minimize the value obtained from the loss function.

$$W_2 = W_2 - \eta \frac{\partial L}{\partial W_2}$$

$$W_1 = W_1 - \eta \frac{\partial L}{\partial W_1}$$

Bias Update Rules

Lastly, the biases are updated.

$$b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}$$

$$b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}$$

Results and Discussion

Time Series Analysis

Neural Networks

Training

Testing

Company	MSE
AMD	0.005859265
Alphabet	0.000862567
Amazon	0.000831416
Apple	0.007479319
BHP Group Limited	0.001295159
Coca Cola	0.006657724
East Africa Metals	0.003050332
HSBC Holdings	0.000289854
Intel	0.001157845
Johnson and Johnson	0.077928339
L’Oreal	0.026000365
Meta	0.123486883
Microsoft	0.041602214
Nestle	0.000945337
Novartis AG	0.007599726
Proctor and Gamble	0.001207186
Samsung Electronics	0.000384859
Siemens AG	0.021622344
Tencent Holdings Limited	0.000328486
Toyota Motor Corporation	0.156136595

Validation

Company	MSE
AMD	0.011286
Alphabet	0.036998
Amazon	0.041988
Apple	0.044326
BHP Group Limited	0.015347
Coca Cola	0.051388
East Africa Metals	0.483313
HSBC Holdings	0.102384
Intel	0.072831
Johnson and Johnson	0.049414
L'Oreal	0.174741
Meta	0.028277
Microsoft	0.024886
Nestle	0.049855
Novartis AG	0.028158
Proctor and Gamble	0.126288
Samsung Electronics	0.020738
Siemens AG	0.154387
Tencent Holdings Limited	0.01493
Toyota Motor Corporation	0.017593

Conclusion and Recommendations

Time Series Analysis

Neural Networks

References

Appendices

Company	Testing MSE	Validation MSE
AMD	0.005859265	0.011285854
Alphabet	0.000862567	0.03699829
Amazon	0.000831416	0.041988428
Apple	0.007479319	0.044326173
BHP Group Limited	0.001295159	0.015347291
Coca Cola	0.006657724	0.051387616
East Africa Metals	0.003050332	0.483313323
HSBC Holdings	0.000289854	0.102384032
Intel	0.001157845	0.072831488
Johnson and Johnson	0.077928339	0.049413927
L'Oreal	0.026000365	0.174740765
Meta	0.123486883	0.028276792
Microsoft	0.041602214	0.024885528
Nestle	0.000945337	0.049855178
Novartis AG	0.007599726	0.028157622
Proctor and Gamble	0.001207186	0.126287847
Samsung Electronics	0.000384859	0.020737847
Siemens AG	0.021622344	0.154387357
Tencent Holdings Limited	0.000328486	0.014930324
Toyota Motor Corporation	0.156136595	0.017593062

Fig 1.1: Testing and Validation MSE’s