Artificial Neural Networks for Stock Price Prediction

Akbar Maulana
Dept. Science and Technology
Universitas Teknologi Yogyakarta
Indonesia

Enny Itje Sela
Dept. Science and Technology
Universitas Teknologi Yogyakarta
Indonesia

ABSTRACT

This research is based on a problem that is difficult to predict stock prices, especially for beginners. Stock prices are difficult to predict because stock prices are volatile. By using artificial neural networks, users will find it easier to predict stock prices. The artificial neural network method used is Multilayer Perceptron. Multilayer Perceptron (MLP) is a variant of an artificial neural network and is a development of perceptron. The selection of the Multilayer Perceptron method is based on the ability of MLP to solve various problems both classification and regression. The research conducted by the author is a regression problem because MLP is asked to predict the close price or closing price of shares after seven days. The results of the model built are able to predict stock prices and produce good accuracy because the resulting RMSE value is 0.042649862994352014 and the RMSE value is close to 0.

General Terms

Neural networks, Multilayer Perceptron (MLP), Forecasting

Keywords

Multi-layer perceptron, Stock price prediction, Artificial neural network

1. INTRODUCTION

Shares are a proof of value that shows ownership in a company. Buying shares of a company's stock can mean we have ownership of the company. Buying shares has both short-term and long-term benefits. In the short term, profits can be obtained when stock buyers purchase shares at a low price and sell them when the stock price rises (capital gain). Additionally, owning shares in the long term allows shareholders to receive a portion of the company's profits, although this may require a longer time period.

However, it is important to note that stock prices are highly volatile and can fluctuate due to various factors. These factors include demand and supply dynamics, inflation, interest rates, and the overall performance of the company. As a result, individuals who are new to the stock market or lack knowledge and experience may find it challenging to accurately predict stock prices.

One significant risk that arises when purchasing stocks without proper analysis is the potential dissolution of the company (liquidation) and a subsequent decline in stock prices. To mitigate these risks, it is crucial to analyze the value of the stock before making a purchase. By conducting a thorough analysis, investors can make informed decisions, maximize potential profits, and minimize potential losses.

There are many methods that can be used to predict stock prices. One of these methods is Machine Learning. Machine Learning is a sub of Artificial intelligence (AI) that aims to improve knowledge and performance [1]. Machine Learning also has various algorithms. One of these algorithms is Multilayer Perceptron. The Multilayer Perceptron is an extension of the perceptron, consisting of an input layer, an output layer, and at least one hidden layer [2].

2. LITERATURE REVIEW

The research is not separated from the previous research results that are used as a comparison and study material. The comparison and study of the research results are closely related to the research topic, which is about stock price prediction. Akil et al (2022) stated that in order to accurately predict whether the stock price will rise or fall, technical analysis needs to be conducted. The problem with technical analysis using manual statistical calculations is that it is difficult to obtain real-time predictions because it takes time to perform the calculations, while in the trading world, stock price fluctuations occur very quickly. In this study, the object used is Twitter stock data with a total of 2118 records and consisting of 7 attributes. The method used in this study is LSTM with single-step and multi-step models, and these two models are compared to choose which model is better for predicting Twitter stock prices. This study concludes that LSTM with the single-step model is able to predict Twitter stock prices more accurately with the last obtained loss values from 24 epochs being 0.0770 (Open), 0.0885 (Close). The value of loss is 1.0363 (Open), 0.7981 (Close), and the value of MAE is 0.9300 (Open), 0.8042 (Close). As for the multi-step model, at the 16th epoch, the loss values obtained are 0.3638 (Open) and 0.3864 (Close), the value of loss is 0.4163 (Open) and 5.1493 (Close), and the value of MAE is 2.0540 (Open) and 2.0668 (Close) [3].

Afrianto et al (2022) stated that stocks are an attractive investment because they can provide high levels of profit. Stock investors try to choose good investment companies to obtain profits with relatively small risks. Therefore, stock investors need to be careful and evaluate a company. In this study, the object used is Bank BCA stock data with 5 attributes and public sentiment from Twitter. The method used in this study is LSTM with single-step and multi-step models, and these two models are compared to choose which model is better for predicting Twitter stock prices. This study obtained the lowest error value of 0.094 for MSE and 0.306 for RMSE, as well as the highest Dstat value with a value of 68% [4].

Nopianti et al (2022) stated that stock price prediction is an interesting data mining area because there are many variables that affect stock prices. Stock prices are volatile, especially in the covid-19 era which has an impact on the economy. In this study, the object used is the financial statements of companies listed on the Indonesia Stock Exchange, where data related to the stock prices of telecommunications companies are sampled. The methods used in this research are BiLSTM and VADER (Valence Aware Dictionary and Sentiment Reasoner). This study found that high correlation coefficients were generated from Decision Tree Regression and K-Nearest Regression. Decision Tree Regression produced good results on Train Test Split and K Fold Cross Validation data, 2.99% and 2.98% [5].

Putra et al (2022) state that it is very important to predict stock prices to reduce the risk to investors when making investment decisions. a stock prediction analysis method is needed that has more accurate results and to reduce the risk to investors when making investment decisions. In this study, the object used is the BRI Bank stock price obtained from the investing.com website. The data amounted to 3040 and had 7 attributes. The method used in this research is Linear Regression. This research gets an accuracy of 0.8992 for training and 0.9125 for testing [6].

Fitriadini et al (2020) state that someone who wants to buy shares must first analyze the value of shares to get maximum profit and technology can be used to predict stock prices that are volatile. In this study, the objects used are 3875 BCA Bank stock data, 4026 BRI Bank stock data and 3995 BNI Bank stock data. The data has 7 attributes and is sourced from yahoo finance. The method used in this research is backpropagation. The results of the research conducted are RMSE: 0.01035 for BBCA, 0.017248 for BBNI, 0.0169134 for BBRI [7].

3. RESEARCH METHOD

3.1 Research Framework

The framework for this research begins with the problems faced by users who have difficulty predicting stock prices. Based on the explanation in the background, artificial neural networks or neural networks are able to predict stock prices with a fairly high level of accuracy so that neural networks can be used as a strategy that is applied to predict stock prices. After going through the necessary process, the output produced is in the form of stock predictions which in the future are expected to be able to help users.

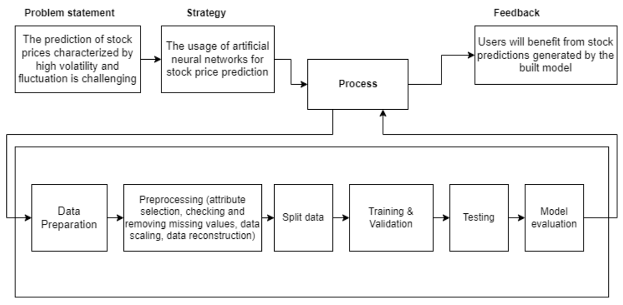

Problem statement
The prediction of stock prices characterized by high volatility and fluctuation is challenging
Strategy
The usage of artificial neural networks for stock price prediction
Process
Feedback
Users will benefit from stock predictions generated by the built model
Data Preparation
Preprocessing (attribute selection, checking and removing missing values, data scaling, data reconstruction)
Split data
Training & Validation
Testing
Model evaluation

Fig. 1 Research Framework

3.2 Data Source

The data utilized in this study comprises The Walt Disney Company (DIS) stock prices obtained from Yahoo Finance. The choice of Disney's stock is grounded in statistical evidence, demonstrating a significant year-over-year growth in Disney+ subscribers, a video-on-demand subscription service owned and operated by Walt Disney, within the United States. The data is accessed through the yfinance library, eliminating the necessity for manual downloads and enabling seamless online access.

The dataset covers the time period from January 1, 2008, to April 14, 2023, encompassing a total of 3847 rows. Key features of the dataset include date, opening price (open), highest price (high), lowest price (low), closing price (close), Adj Close, and Volume. For reference, a sample of the data is presented in Table 1.

Table 1 Example of dataset

No	Date	Open	High	Low	Close	Volume
1	2008-01-02	27,41	27,682	26,885	27,012	9269900
2	2008-01-03	27,046	27,165	26,859	26,953	9681100
3	2008-01-04	26,299	26,876	26,299	26,410	9550700
4	2008-01-07	26,622	26,715	26,223	26,435	10742900
5	2008-01-08	26,511	26,724	25,841	25,909	13014300

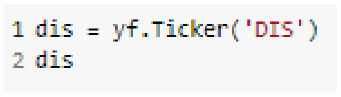
From the provided data, only the "close" column will be used in this research, and all other columns will be removed. The resulting dataset that will be used for this study can be observed in Table 2.

Table 2 Data after select close column

No	Date	Close
1	2008-01-02	27,012
2	2008-01-03	26,953
3	2008-01-04	26,410
4	2008-01-07	26,435
5	2008-01-08	25,909

3.3 How to Collect Data

As previously mentioned in subsection 3.2, the Walt Disney stock data used in this study was obtained by accessing Yahoo Finance using the yfinance library. The code for accessing Yahoo Finance using the yfinance library can be referenced in Figure 2.



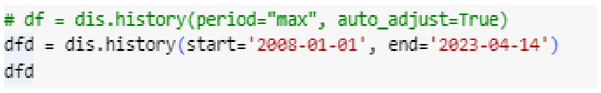
```
dis = yf.Ticker('DIS')
dis
```

Fig. 2 Code for acces walt disney stock price

The purpose of this code is to specify which stock data to access. The symbol for The Walt Disney Company is (DIS), so the stock symbol for Walt Disney Company is provided as a parameter. The data will be stored in the variable 'dis'.

Since the data to be used consists of historical prices, the .history method is used, followed by the parameters 'start'

(2008-01-01) and 'end' (2023-04-14). These parameters are used to define the time period for which data is required. The data will be stored in the variable 'df'. The code for this operation can be seen in Figure 3.



```
# df = dis.history(period="max", auto_adjust=True)
dfd = dis.history(start='2008-01-01', end='2023-04-14')
dfd
```

Fig. 3 Code to get the walt disney stock price data

3.4 Model Architecture

The model architecture illustrating how the system functions in this research can be seen in Figure 4. Figure 4 represents the model architecture employed in this study. The first stage involves users searching for the data to be used. The subsequent stage is the application of this data to the designed system. After the data has been processed within the designed system, the output produced consists of stock price predictions and their comparisons with global stock prices.

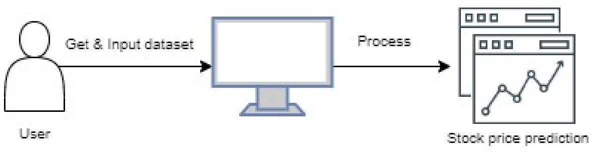


Fig. 4 Model Architecture

3.5 Conceptual Design

The conceptual and physical design of the system constructed by the researcher can be observed in Figure 5. An explanation of Figure 5 comprises the following stages: The initial stage involves reading the downloaded stock data. Subsequent stages encompass data preprocessing, including several steps. These steps involve feature selection or attribute selection, specifically 'close,' removing missing values, scaling the data, determining h+7 (seven days after day h) as the target variable, and dividing the dataset into training, validation, and testing data. The next stage involves the creation of an artificial neural network model that will be employed. Once the artificial neural network model is constructed, the subsequent step is model training using the training data, followed by model validation using the validation data. The trained artificial neural network model is then saved and employed using the test data. Next comes the model evaluation stage. The evaluation process entails computing the error value using Root Mean Square Error. Following the evaluation, graphical representations and tables comparing predicted stock prices with actual stock prices will be presented

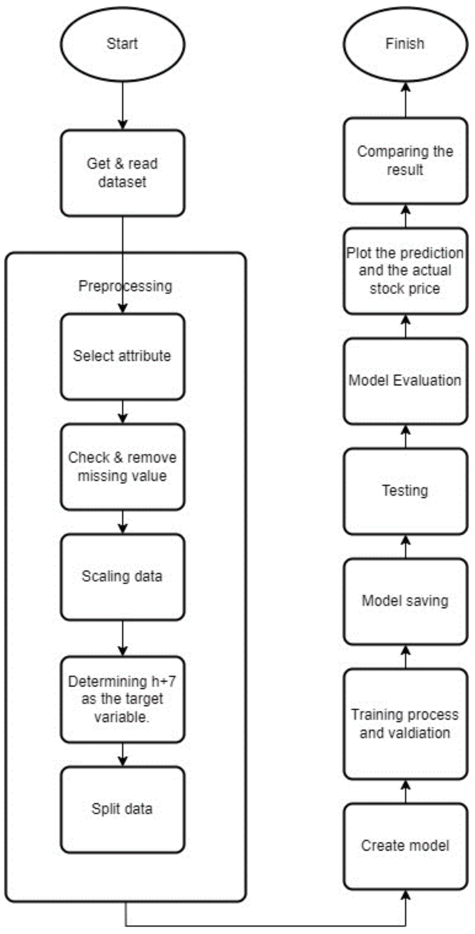
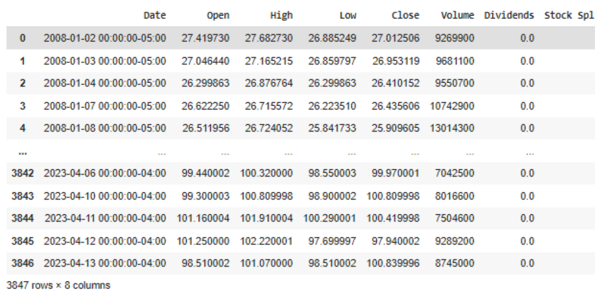


Fig. 5 Flowchart

4. IMPLEMENTATION

The first step involves downloading and importing the necessary libraries. Subsequently, accessing and downloading the Walt Disney Company stock dataset from Yahoo Finance. After that, calling the data into a dataframe and specifying the date parameters for the stock data to be included in the dataframe. Then, the dataframe will be read and displayed



	Date	Open	High	Low	Close	Volume	Dividends	Stock Spl
0	2008-01-02 00:00:00-05:00	27.419730	27.682730	26.885249	27.012506	9269900	0.0	
1	2008-01-03 00:00:00-05:00	27.046440	27.165215	26.859797	26.953119	9681100	0.0	
2	2008-01-04 00:00:00-05:00	26.299863	26.876764	26.299863	26.410152	9550700	0.0	
3	2008-01-07 00:00:00-05:00	26.622250	26.715572	26.223510	26.435606	10742900	0.0	
4	2008-01-08 00:00:00-05:00	26.511956	26.724052	25.841733	25.909605	13014300	0.0	
...	...	...	...	...	...	...	...	
3842	2023-04-06 00:00:00-04:00	99.440002	100.320000	98.550003	99.970001	7042500	0.0	
3843	2023-04-10 00:00:00-04:00	99.300003	100.809998	98.900002	100.809998	8016600	0.0	
3844	2023-04-11 00:00:00-04:00	101.160004	101.910004	100.290001	100.419998	7504600	0.0	
3845	2023-04-12 00:00:00-04:00	101.250000	102.220001	97.699997	97.940002	9289200	0.0	
3846	2023-04-13 00:00:00-04:00	98.510002	101.070000	98.510002	100.839996	8745000	0.0	

3847 rows × 8 columns

Fig. 6 - Dataset

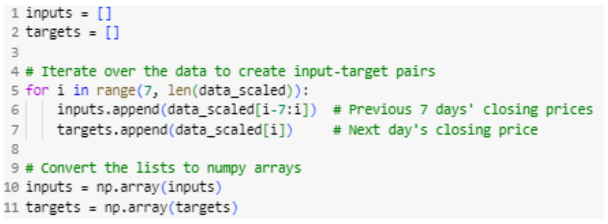
The next step is the preprocessing stage. From the numerous columns available, only the 'close' column will be used. After selecting only the 'close' column, additional preprocessing steps are undertaken, which include checking for missing values, removing any missing values, and normalizing the data through data scaling using MinMaxScaler to facilitate computations in the artificial neural network model. The formula for MinMaxScaler can be observed in the Equation 1

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Equation 1. Minmax scaler formula

From the dataset used, it is known that the maximum value (max) and the minimum value (min) are 201.910 and 13.424, respectively

Subsequently, the data is restructured, where the data from seven days prior will be used as input, and the data from the eighth day will be used as the target. The code can be seen in Figure 7



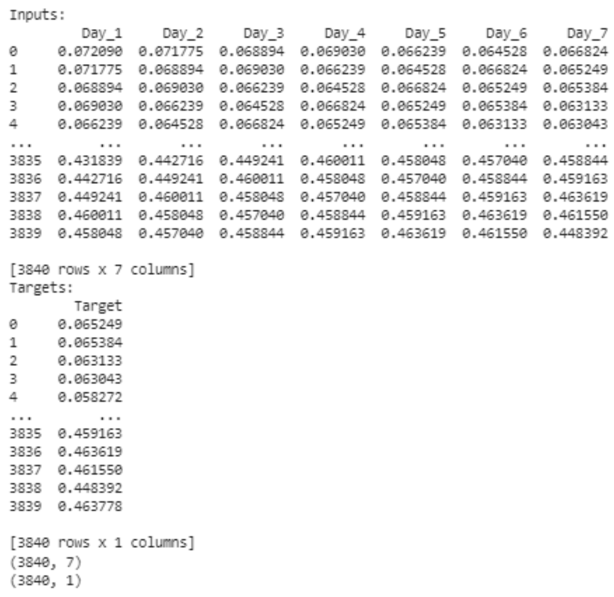
```
inputs = []
targets = []

# Iterate over the data to create input-target pairs
for i in range(7, len(data_scaled)):
    inputs.append(data_scaled[i-7:i])  # Previous 7 days' closing prices
    targets.append(data_scaled[i])     # Next day's closing price

# Convert the lists to numpy arrays
inputs = np.array(inputs)
targets = np.array(targets)
```

Fig. 7 Code to reconstructed the data

The results of the process described above can be observed in Figure 8



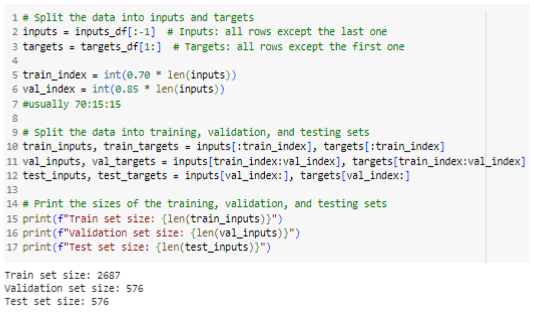
```
Inputs:
         Day_1     Day_2     Day_3     Day_4     Day_5     Day_6     Day_7
0     0.072090  0.071775  0.068894  0.069030  0.066239  0.064528  0.066824
1     0.071775  0.068894  0.069030  0.066239  0.064528  0.066824  0.065249
2     0.068894  0.069030  0.066239  0.064528  0.066824  0.065249  0.065384
3     0.069030  0.066239  0.064528  0.066824  0.065249  0.065384  0.063133
4     0.066239  0.064528  0.066824  0.065249  0.065384  0.063133  0.063043
...        ...       ...       ...       ...       ...       ...       ...
3835  0.431839  0.442716  0.449241  0.460011  0.458048  0.457040  0.458844
3836  0.442716  0.449241  0.460011  0.458048  0.457040  0.458844  0.459163
3837  0.449241  0.460011  0.458048  0.457040  0.458844  0.459163  0.463619
3838  0.460011  0.458048  0.457040  0.458844  0.459163  0.463619  0.461550
3839  0.458048  0.457040  0.458844  0.459163  0.463619  0.461550  0.448392

[3840 rows x 7 columns]
Targets:
        Target
0     0.065249
1     0.065384
2     0.063133
3     0.063043
4     0.058272
...        ...
3835  0.459163
3836  0.463619
3837  0.461550
3838  0.448392
3839  0.463778

[3840 rows x 1 columns]
(3840, 7)
(3840, 1)
```

Fig. 8 Data after being reconstructed

After the data has been restructured, it will be divided into three sets: training data, validation data, and testing data, with a ratio of 70%:15%:15%. The code for data splitting can be seen in Figure 9



```
# Split the data into inputs and targets
inputs = inputs_df[:-1]  # Inputs: all rows except the last one
targets = targets_df[1:]  # Targets: all rows except the first one

train_index = int(0.70 * len(inputs))
val_index = int(0.85 * len(inputs))
#usually 70:15:15

# Split the data into training, validation, and testing sets
train_inputs, train_targets = inputs[:train_index], targets[:train_index]
val_inputs, val_targets = inputs[train_index:val_index], targets[train_index:val_index]
test_inputs, test_targets = inputs[val_index:], targets[val_index:]

# Print the sizes of the training, validation, and testing sets
print(f"Train set size: {len(train_inputs)}")
print(f"Validation set size: {len(val_inputs)}")
print(f"Test set size: {len(test_inputs)}")
```

```
Train set size: 2687
Validation set size: 576
Test set size: 576
```

Fig. 9 Code for splitting the data

The next step is model creation. The model constructed in this research consists of seven neurons in the input layer, two hidden layers, each with seven neurons, and an output layer comprising a single neuron. The architecture of the research model can be observed in Figure 10

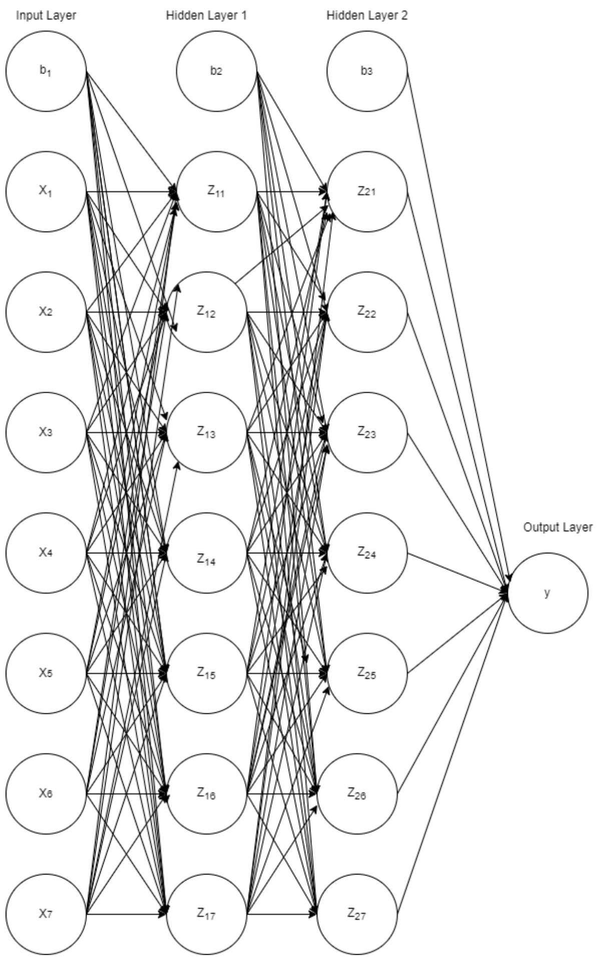


Fig. 10 Neural network model

Next is compiling the model. In this research, the optimizer used is stochastic gradient descent. For the activation function, there are three options: tanh, relu, and sigmoid. Additionally, sliding windows are incorporated in this study with the aim of enhancing model performance. Sliding windows work by dividing the data into fixed-sized overlapping or overlapping windows. The artificial neural network is then trained to predict the target based on input values from the sliding windows. Here is an explanation of the example use of sliding windows with a window size of 4. The neurons in the input layer should receive input as follows:

x1 = 0.07209, x2 = 0.0717, x3 = 0.0688, x4 = 0.0693, x5 = 0.0662,3 x6 = 0.06452, and x7 = 0.0668

And for the target:

y = 0.0652

If we use a sliding window of size 4, the data will be divided according to the window size, which is 4, so each window contains 4 values. The windows generated using the example data from x1 to x7 are as follows:

- [0.07209, 0.0717, 0.0688, 0.0693] for the first window.

- [0.0717, 0.0688, 0.0693, 0.06623] for the second window.
- [0.0688, 0.0693, 0.06623, 0.06452] for the third window.
- [0.0693, 0.06623, 0.06452, 0.0668] for the fourth window.

It's important to note that the window size refers to how many data points are included in a single window, not the number of windows.

Next, we move on to the training phase, where the model will be trained using the previously prepared training data. In Figure 5.1.6, there are seven neurons in the input layer, but after applying sliding windows, each window contains only four values, while there are seven neurons in the input layer. To address this, the first neuron (x1) through the fourth neuron (x4) take values from the first window [0.07209, 0.0717, 0.0688, 0.0693] as input, and the fifth neuron (x5) through the seventh neuron (x7) take values from the second window [0.0717, 0.0688, 0.0693] as input. Thus, in the first iteration, the data used for the first through seventh neurons are [0.07209, 0.0717, 0.0688, 0.0693, 0.0717, 0.0688, 0.0693] with the same target value of 0.0652.

The training phase consists of four process. The first process is forward propagation. The equation of forward propagation can be seen at Figure

$$dot_j = (\sum_i^n w_i * x_i + b)$$
$$a = f(dot_j)$$

Equation 2. Equation of forward propagation

The description of Equation 2 is as follows:

dotj = dot product

n = the number of neurons in the input layer

i = node in the input layer

wi = weight

xi = input value

b = bias

a = output in the hidden layer after activation

f = activation function

There are several activation functions that can be used in Equation 2. The first one is the sigmoid function (σ). The sigmoid activation function and its derivative can be calculated using Equation 3 and Equation 4, respectively.

$$f_{(x)} = \frac{1}{1 + e^{-(x)}}$$

Equation 3. Formula of sigmoid activation function

$$f'_{(x)} = \left(\frac{1}{1 + e^{-(x)}}\right) * \left(1 - \frac{1}{1 + e^{-(x)}}\right)$$

Equation 4. Derivative of sigmoid activation function

In addition to the sigmoid activation, there are other activations such as ReLU and tanh. The equations for the activation function and derivative of ReLU (R) or rectified linear units can be seen in Rquations 5 and Equation 6, respectively.

$$f_{(x)} = \max(0, x)$$

Equation 5. Formula of relu activation function

$$f'_{(x)} = \begin{cases} 0\ if\ x \leq 0 \\ 1\ if\ x > 0 \end{cases}$$

Equation 6. Derivative of relu activation formula

Meanwhile, for the activation function and derivative of Tanh (hyperbolic tangent), you can refer to Equation 7 and Equation 8, respectively

$$f_{(x)} = \frac{e^x - e^{-(x)}}{e^x + e^{-(x)}}$$

Equation 7. Formula of tanh activation function

$$f'_{(x)} = 1 - \left(\frac{(e^x - e^{-(x)})^2}{(e^x + e^{-(x)})^2}\right)$$

Equation 8. Derivative of tanh activation formula

The second process is evaluating the error. The output from the output layer after the forward propagation process will be used to determine the error. The type of error calculation that being used is MSE. The formula for MSE can be seen in Equation 9

$$MSE = {}^{1 * \sum(y - y_{pred})^2}/_{n}$$

Equation 9. Mean Squre Error (MSE) formula

The description of the equations is as follows:

E = error

y = target

ypred = output from the output layer

n = the number of data points used

The third process is backpropagation. This step is used to update each weight in the neural network. The goal of this process is to find optimal weights to minimize the error generated by the model, leading to better model performance. The first step to take is to calculate the delta for each neuron using Equation 10

$$\Delta_j = (a - y) * f'(a)$$

Equation. 10 Delta formula

The description of Equation 10 is as follows:

Δj = delta (the error term)

a = the activation output result at a neuron

y = the target

f'(a) = the derivative function of the activation function

The fourth process is updating the weight using the previously calculated delta and the known learning rate. The equation for weight updates can be seen in Equation 11

$$w' = w - (\mu * \Delta_j * a)$$

Equation 11. New weight formula

The description of Equation 11 is as follows:

w' = new weight

w = old weight

μ = learning rate

The trained model will be validated using the validation data that was prepared during the data splitting process. Afterward, the artificial neural network model will be saved and reused during the testing phase. At this stage, if the loss from the training and validation processes is still relatively high, parameters such as learning rate, epoch, activation function, the size of the sliding window, and so on, may be adjusted accordingly.

The next step is to apply the test data to the saved model. After testing, the data that was scaled using MinMaxScaler will be inverted or transformed back to its original values.

The model evaluation in this study uses RMSE (Root Mean Square Error). The output or predictions from the trained model are subtracted from the actual data, then averaged and square-rooted. Following this, graphs are created to visualize the comparison between the prediction results and the ground truth (original data). The subsequent step involves comparing the predicted values with the actual stock data.

5. RESULT AND DISCUSSION

There were several experiments conducted to find more suitable parameters for the constructed model. These parameters include learning rate, epoch, batch size, window size, and activation function. The results of the experiments can be seen in Table 3. The smallest RMSE obtained was 0.042649862994352014, indicating that the model has achieved good performance

Table 3. Table of experiments

Learning rate	Epoch	Batch size	Windows size	seed	Activation	RMSE
0.05	1000	42	4	42	Relu	0.042649862994352014
0.01	1000	21	4	42	Relu	0.22088035881599877
0.01	1000	42	6	21	Relu	0.060893733828284655
0.08	1000	42	4	42	Relu	0.16104509002174425
0.01	1000	42	6	21	Tanh	0.13048368913533623
0.05	1000	42	4	42	Tanh	0.15935786586165798
0.01	1000	21	4	21	Tanh	0.13400511143257018
0.01	1000	21	4	21	Sigmoid	0.17954298277176617
0.05	1000	42	4	42	Sigmoid	0.16104509002174425
0.01	1000	42	6	21	Sigmoid	0.17875569186199866

Although the RMSE achieved is considered good as it is below 0.1, there is still a significant difference between the predicted stock prices and the actual stock prices, particularly in the early part of the data. This can be observed in the graphs shown in Figure 11 and Figure 12.

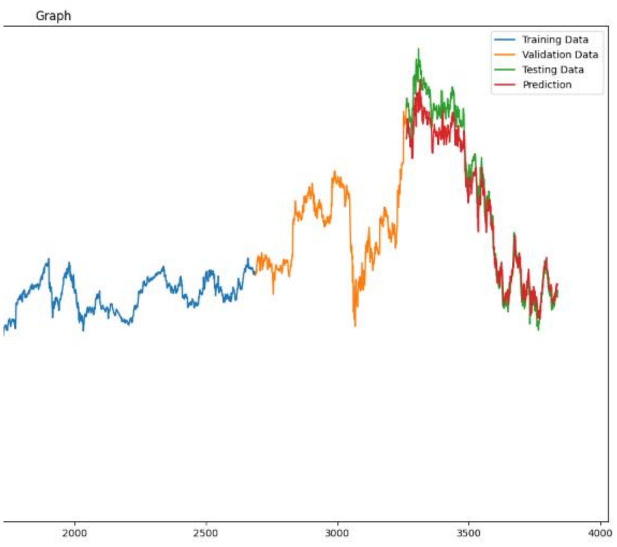
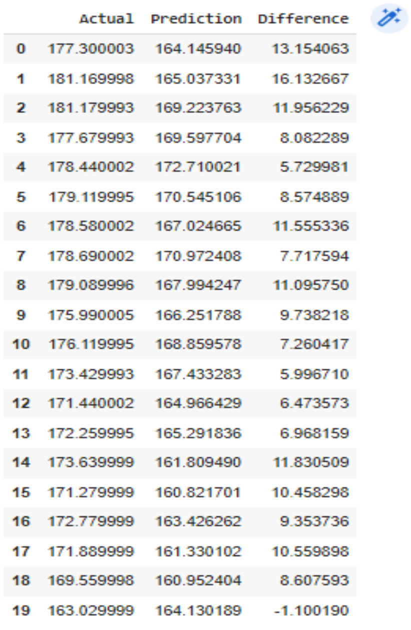


Fig. 11 Graphic of train, validation, testing and prediction



	Actual	Prediction	Difference
0	177.300003	164.145940	13.154063
1	181.169998	165.037331	16.132667
2	181.179993	169.223763	11.956229
3	177.679993	169.597704	8.082289
4	178.440002	172.710021	5.729981
5	179.119995	170.545106	8.574889
6	178.580002	167.024665	11.555336
7	178.690002	170.972408	7.717594
8	179.089996	167.994247	11.095750
9	175.990005	166.251788	9.738218
10	176.119995	168.859578	7.260417
11	173.429993	167.433283	5.996710
12	171.440002	164.966429	6.473573
13	172.259995	165.291836	6.968159
14	173.639999	161.809490	11.830509
15	171.279999	160.821701	10.458298
16	172.779999	163.426262	9.353736
17	171.889999	161.330102	10.559898
18	169.559998	160.952404	8.607593
19	163.029999	164.130189	-1.100190

Fig. 12 Comparison between actual price and prediction

6. CONCLUSION AND SUGGESTIONS

6.1 Conlusion

From the conducted research, it can be concluded that predicting stock prices using a multi-layer perceptron neural network with backpropagation has performed well. This is evident from the small RMSE value of 0.042649862994352014. Another supporting piece of evidence is the price difference between the predicted and actual prices. However, there is still an issue observed in the early part of the testing, where there is a significant difference between the predicted and actual stock prices

6.2 Suggestion

Based on the issue mentioned in the conclusion, which is the significant difference between the predicted and actual stock prices in the early part of the testing, a suggestion for future research is to investigate the underlying reasons for this occurrence. Another recommendation is to explore parameter adjustments in future experiments, such as changing the optimizer, epoch, batch size, learning rate, and other relevant parameters, to potentially improve the model's performance

7. REFERENCES

[1] F. Provost and T. Fawcett, “Data Science and its Relationship to Big Data and Data-Driven Decision Making,” Big Data, vol. 1, no. 1, pp. 51–59, Mar. 2020, doi: https://doi.org/10.1089/big.2013.1508.

[2] A. Subasi, Practical Machine Learning for Data Analysis Using Python. Academic Press, 2020.

[3] I. Akil and I. Chaidir, “Prediksi Harga Saham Twitter Dengan Long Short-Term Memory Recurrent Neural Network,” INTI Nusa Mandiri, vol. 17, no. 1, pp. 1–7, Aug. 2022, doi: https://doi.org/10.33480/inti.v17i1.3277.

[4] N. Afrianto, D. H. Fudholi, and S. Rani, “Prediksi Harga Saham Menggunakan BiLSTM dengan Faktor Sentimen Publik,” Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 6, no. 1, pp. 41–46, Feb. 2022, doi: https://doi.org/10.29207/resti.v6i1.3676

[5] R. Nopianti, A. T. Panudju, and A. Permana, “Prediksi Harga Saham Indonesia pada Masa Covid-19 Menggunakan Regresi Pohon Keputusan,” Jurnal Ecodemica Jurnal Ekonomi Manajemen dan Bisnis, vol. 6, no. 1, pp. 68–76, Apr. 2022, doi: https://doi.org/10.31294/eco.v6i1.11365.

[6] J. S. Putra, R. D. Ramadhani, and A. Burhanuddin, “Prediksi Harga Saham Bank Bri Menggunakan Algoritma Linear Regresion Sebagai Strategi Jual Beli Saham,” Journal of Dinda : Data Science, Information Technology, and Data Analytics, vol. 2, no. 1, pp. 1–10, Feb. 2022, doi: https://doi.org/10.20895/dinda.v2i1.273.

[7] A. Fitriadini, T. Pramiyati, and A. B. Pangaribuan, “PENERAPAN BACKPROPAGATION NEURAL NETWORK DALAM PREDIKSI HARGA SAHAM,” in Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA), Jakarta, Aug. 2020, pp. 1–13. Available: https://conference.upnvj.ac.id/index.php/senamika/article/view/627/464