Contents

1 Introduction

In this project we are asked to test different algorithm against the time series data and were asked to describe our analysis on different algorithm. Description of the project contain the information where would we collect our data from, which is just a stock price data throughout the time, description of the project also suggests us to use an Algorithms which we will implement in the following project. The three algorithms that I have chosen in this project are LSTM (Long Short Term Memory), RNN (Recurrent Neural Network) and ARIMA (AutoRegressive Integrated Moving Average).

1.1 Time series data

A time series is a sequence of data points that occur in successive order over some period of time a time series data can be taken in any variable that changes over time. A perfect example of time series data would be investing, it is common to use a time series to track the price of a stock over time. This can be tracked over the short term or long term like what is the value of certain stock an hour ago what is the value now, to what is the value it holds 5 years ago (investopedia, 2022).

1.2 RNN

A RNN (Recurrent Neural Network) is a special type of artificial neural network adapted to work for time series data or data that involves sequences. It is trained to process and convert a sequential data input into a specific sequential data output (Research Graph, 2024). Little more about RNN and how does it works, they are distinguished by their memory as they take information from prior inputs to influence the current input and output (IBM, 2024).

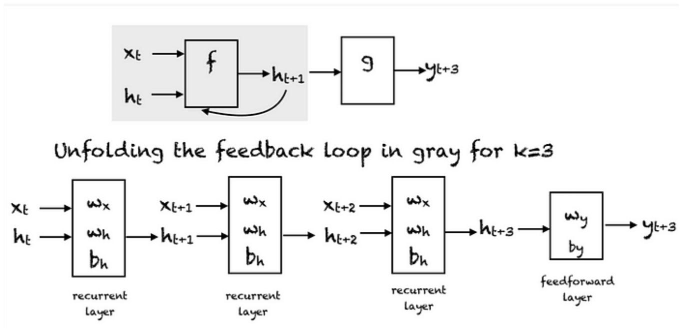


Figure 1 RRN architecture (Research Graph, 2024)

1.3 LSTM

Long Short Term Memory networks usually called LSTMs, which are explicitly designed to avoid the long term dependency problem. Remembering information for long periods of time is practically their default behavior not something they would struggle to learn (Olah, 2015). Unlike standard feedforward neural network, LSTMs have feedback connections allowing them to exploit temporal dependencies across sequences of data. LSTM networks introduce memory cells which have the ability to retain information over long sequences. It is design to handle the issue of vanishing or exploding gradients, which can occur when training traditional RNN on sequences of data (Hamad, 2023).

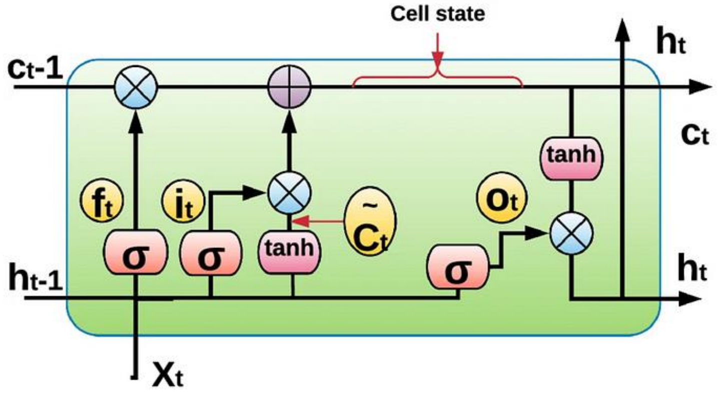


Figure 2 LSTM Network (Hamad, 2023)

1.4 Gradient Boosting machine (GBM)

Gradient boosting is a technique attracting attention for its prediction speed and accuracy especially with large and complex data. It relies on the intuition that the best possible next model, when combined with previous model, and minimizes the overall prediction error. The key idea is to set the target model in order to minimize the error. If the change in the prediction for a case causes a large drop in error then next target outcome of the case is high value (Hoare, 2024).

2 Comparing the algorithm

Let’s start we our first algorithm RNN, we used tensorflow and keras to build this neural network. In this architecture we used there are total 7 layer, 3 RNN hidden layers, 3 Dropout layers which are used for regularization, and 1 dense layer as output. Each layer has 50 neurons. Since RNN are recurrent, so each layers has both an input from previous layer and feedback connection its own. The dense layer final output layer which will hold value between 0 and 1 due to the sigmoid activation function. There a dropout layers between each layer which randomly drops 20% of neurons in the corresponding RNN layer during training to prevent overfitting.

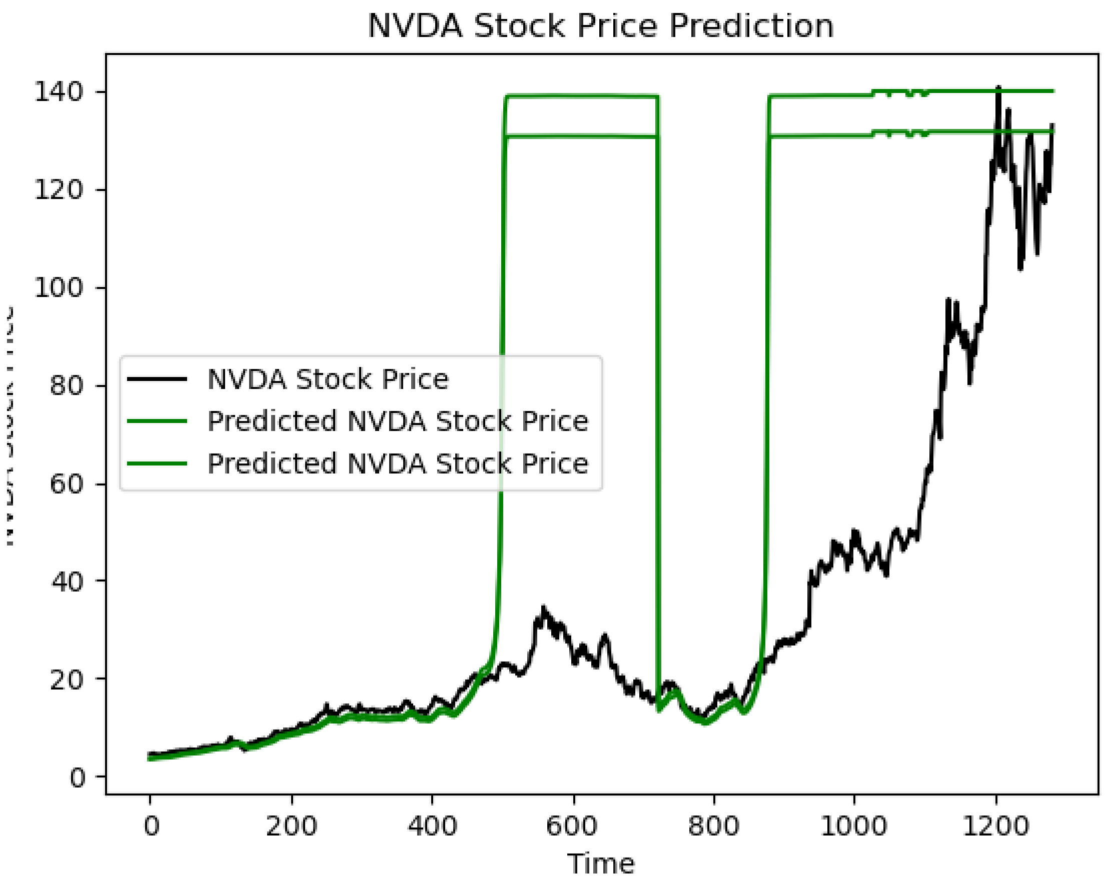
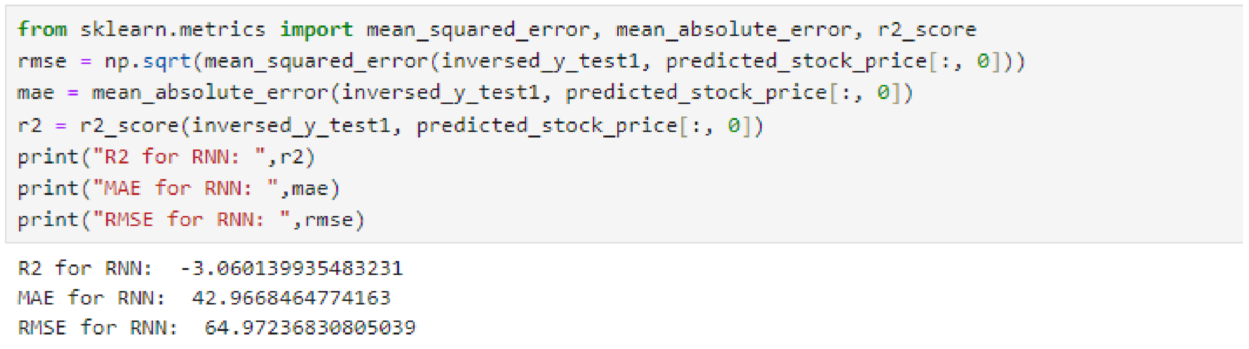


Figure 3 RNN



```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
rmse = np.sqrt(mean_squared_error(inversed_y_test1, predicted_stock_price[:, 0]))
mae = mean_absolute_error(inversed_y_test1, predicted_stock_price[:, 0])
r2 = r2_score(inversed_y_test1, predicted_stock_price[:, 0])
print("R2 for RNN: ",r2)
print("MAE for RNN: ",mae)
print("RMSE for RNN: ",rmse)
```

```
R2 for RNN:  -3.060139935483231
MAE for RNN:  42.9668464774163
RMSE for RNN:  64.97236830805039
```

Figure 4 Loss Metric for RNN

LSTM architecture consist of 9 layers 4 LSTM hidden layer, 4 dropout layer, 1 Dense output layer, each layer output a sequence for each time step, like RNN LSTM layers are recurrent as well it has the same connections from the previous layers as well as feedback connection from its own previous time step, like our previous network I tried to keep the configuration same as possible so each layer here also have same number of neurons. The dropout layer between each layer randomly drops 20% of the neurons during training to prevent overfitting. Below is the graph showing predicting accuracy of the model.

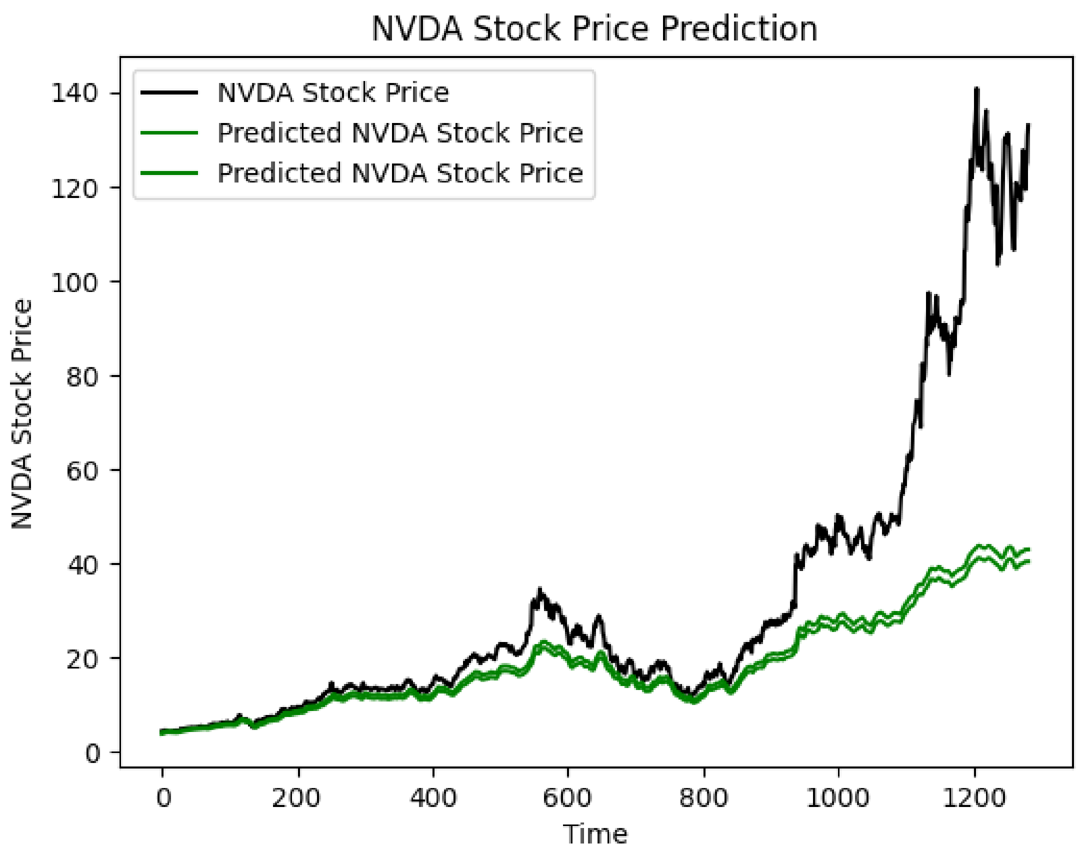
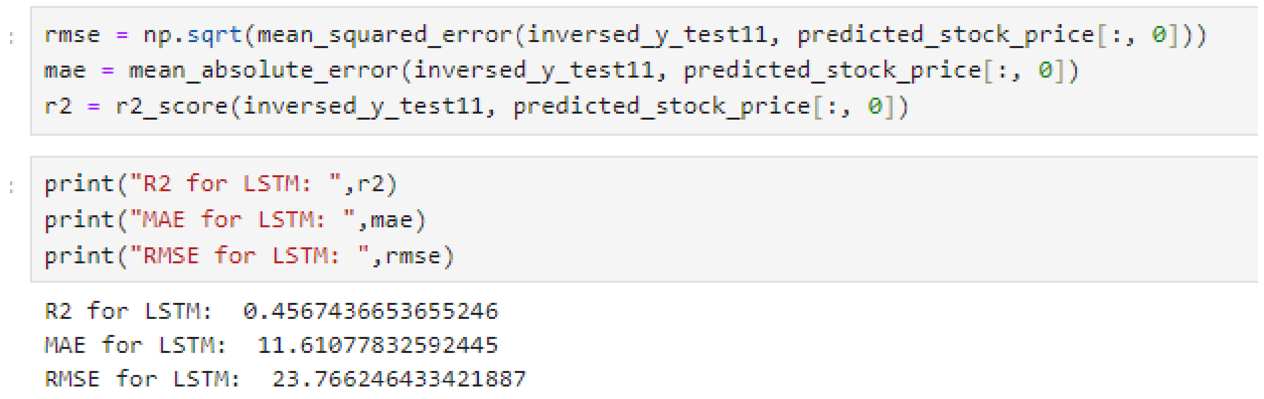


Figure 5 LSTM



```
rmse = np.sqrt(mean_squared_error(inversed_y_test11, predicted_stock_price[:, 0]))
mae = mean_absolute_error(inversed_y_test11, predicted_stock_price[:, 0])
r2 = r2_score(inversed_y_test11, predicted_stock_price[:, 0])
```

```
print("R2 for LSTM: ",r2)
print("MAE for LSTM: ",mae)
print("RMSE for LSTM: ",rmse)
```

```
R2 for LSTM:  0.4567436653655246
MAE for LSTM:  11.61077832592445
RMSE for LSTM:  23.766246433421887
```

Figure 6 Loss Metric for LSTM

The implementation of this algorithm is little different from other algorithm, we have to implement lag feature or shifted features to handle time series data. It is created by shifting a time series backward by specific number of time step, lag feature can help capture how past value of series affect or influence the future values (hopsworks, 2024).

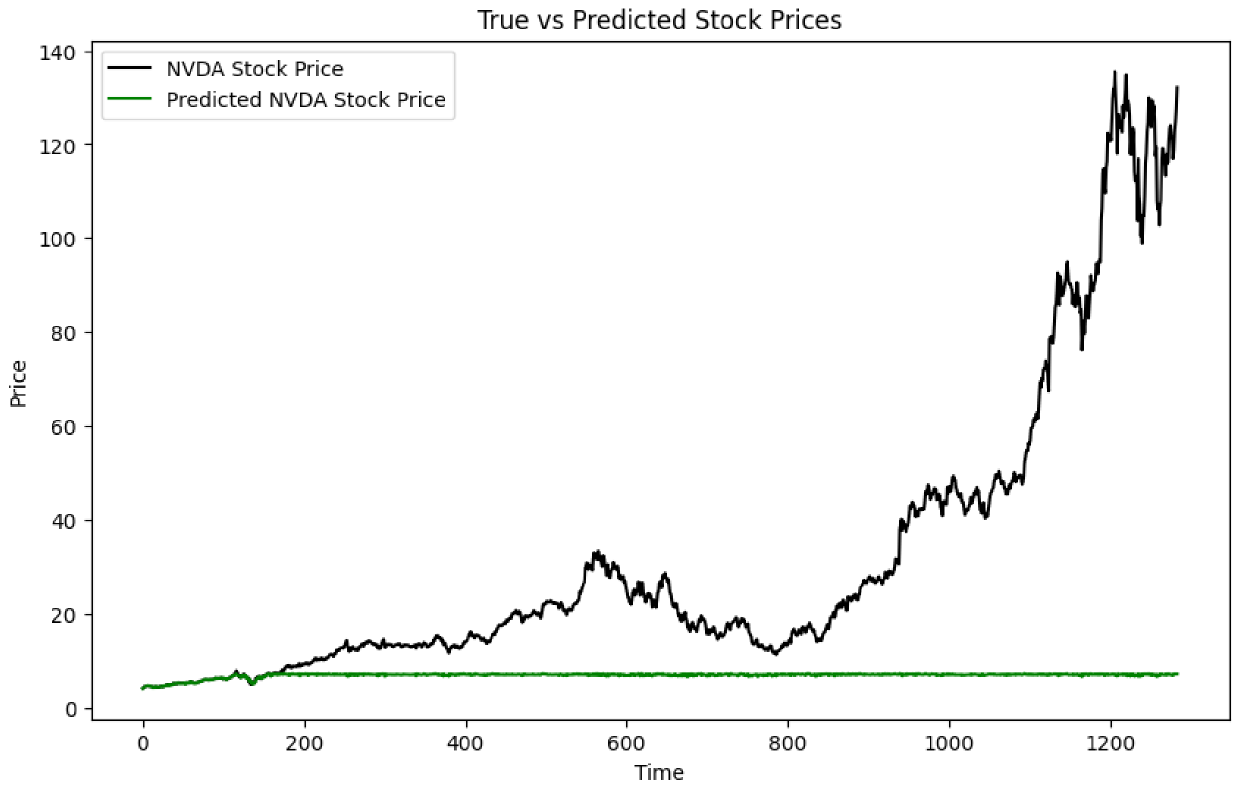
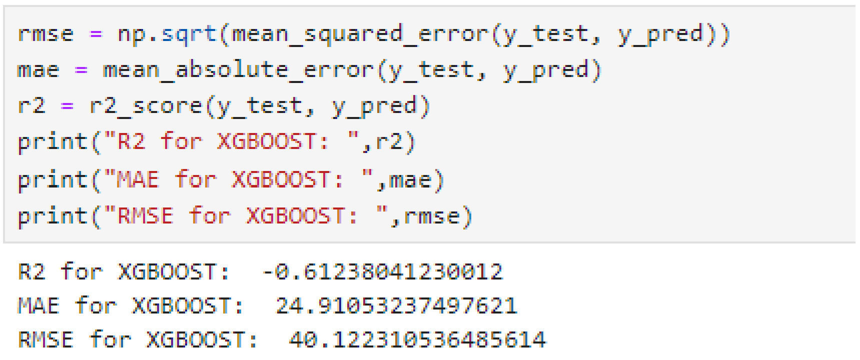


Figure 7 Gradient Boosting Machine



```
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("R2 for XGBOOST: ",r2)
print("MAE for XGBOOST: ",mae)
print("RMSE for XGBOOST: ",rmse)
```

```
R2 for XGBOOST:  -0.61238041230012
MAE for XGBOOST:  24.91053237497621
RMSE for XGBOOST:  40.122310536485614
```

Figure 8 Loss Metric for XGBOOST

Comparing this three algorithms LSTM gives us the best performing output and we will try to recreate the particular algorithm.

3 Own Algorithm

This program was possible because Dr. Fridolin (Fridolin, 2024), this particular youtube channel help me a lot in understanding the concept and implementing the mathematics how the program works program contains of 2 directory one for scrapping data from yahoo finance and one my own implementation of LSTM. Just running main file you can access the algorithm and it prints the loss.

4 References

Brownlee, J. (2023, 11 18). *machinelearningmastery*. Retrieved from How to Create an ARIMA Model for Time Series Forecasting in Python: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

Fridolin, D. (2024, 04 11). *Youtube*. Retrieved from https://www.youtube.com/@DrFridolin

Hamad, R. (2023, 12 02). *Medium*. Retrieved from What is LSTM? Introduction to Long Short-Term Memory: https://medium.com/@rebeen.jaff/what-is-lstm-introduction-to-long-short-term-memory-66bd3855b9ce

Hoare, J. (2024, 10 1). *Displayr*. Retrieved from Gradient Boosting Explained – The Coolest Kid on The Machine Learning Block: https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block/

hopsworks. (2024, 10 1). Retrieved from Lagged features: https://www.hopsworks.ai/dictionary/lagged-features

IBM. (2024, 10 4). *IBM*. Retrieved from What is a recurrent neural network (RNN)?: https://www.ibm.com/topics/recurrent-neural-networks

investopedia. (2022, 06 12). *Blog: Investopedia*. Retrieved from What Is a Time Series and How Is It Used to Analyze Data?: https://www.investopedia.com/terms/t/timeseries.asp

Olah, C. (2015, 08 27). *pages: github*. Retrieved from Understanding LSTM Networks: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Research Graph. (2024, 03 26). *Medium*. Retrieved from An Introduction to Recurrent Neural Networks (RNNs): https://medium.com/@researchgraph/an-introduction-to-recurrent-neural-networks-rnns-802fcfee3098