Assignment 2 - Algorithmic trading

Minghao Zhong(Section 1,2,3) and Napat Viseshsin(Section 4,5)

1.Introduction and previous works.

Algorithmic Trading(AT) is a rapidly growing market regarded as lucrative and potentially massive improvements to be made.It is said that 'The Algorithmic Trading Market is expected to witness a CAGR of 10.5% over the forecast period (2022-2027).'by the report https://www.reportlinker.com/p06246232/Algorithmic-Trading-Market-Growth-Trends-COVID-19-Impact-and-Forecasts.html?utm_source=GNW.

Deep learning now is widely applied to many areas such as medical diagnosis, automatic driving, facial recognition, and some other analytic tasks. RNN (Recurrent Neural Network) is one kind of neural network particularly good at dealing with sequential data, of which LSTM (Long Short-Term Memory) is a variant. LSTM is used to tackle tasks involving long sequential data, which would expose RNN to the problem of gradient vanishing, commonly seen in a feed forward neural network.

In this report, a preliminary effort is made on applying LSTM to FTS (Financial Times Series) in order to predict the future rates and realize the Algorithmic Trading technique. The proposed method is then evolved by an evolutionary approach and compared against a traditional method in Finance dealing with FTS.

Some related previous works are listed as following:

IEEE Xplore Full-Text PDF: Time-weighted LSTM Model with Redefined Labeling for stock price prediction

https://doi.org/10.1016/j.neucom.2018.09.082 Time series forecasting of petroleum production using deep LSTM recurrent networks

https://doi.org/10.3390/su10103765 Genetic Algorithm-Optimized Long Short-Term Memory Network for Stock Market Prediction

https://doi.org/10.1109/ACCESS.2020.3047109 Stock Prediction Based on Genetic Algorithm Feature Selection and Long Short-Term Memory Neural Network

https://doi.org/10.1016/j.neucom.2012.10.043 Evolving RBF neural networks for rainfall prediction using hybrid particle swarm optimization and genetic algorithm

https://doi.org/10.1111/j.1365-2478.2012.01080.x Reservoir permeability prediction by neural networks combined with hybrid genetic algorithm and particle swarm optimization

https://doi.org/10.1016/j.artmed.2011.06.008 Hybrid genetic algorithm-neural network: Feature extraction for unpreprocessed microarray data

https://doi.org/10.1007/s13042-010-0004-x Genetic Algorithm-Neural Network (GANN): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection

https://doi.org/10.1016/j.eswa.2014.07.039 A proposed iteration optimization approach integrating backpropagation neural network with genetic algorithm

https://arxiv.org/abs/1803.01271v2 An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling

https://doi.org/10.1007/978-3-319-93351-1_32 Stock Price Forecast Based on LSTM Neural Network

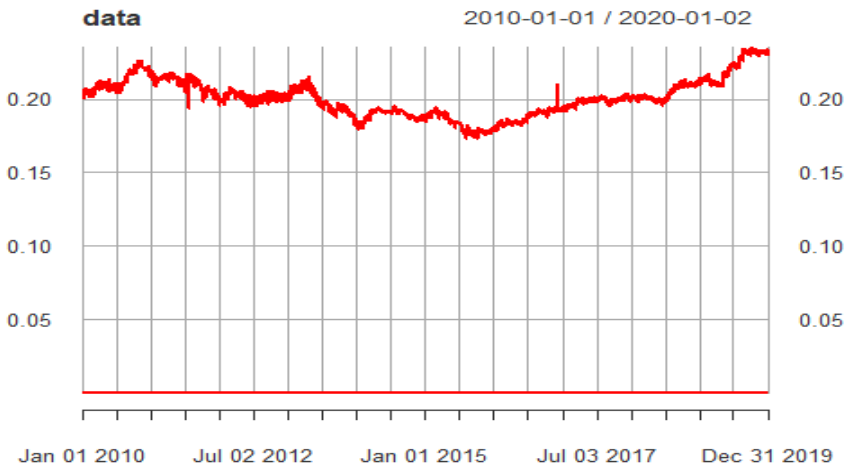
https://doi.org/10.1007/s00521-019-04504-2 Stock closing price prediction based on sentiment analysis and LSTM

https://doi.org/10.1109/ACCESS.2020.3004284 Forecasting Stock Prices Using a Hybrid Deep Learning Model Integrating Attention Mechanism, Multi-Layer Perceptron, and Bidirectional Long-Short Term Memory Neural Network

https://doi.org/10.1016/j.eswa.2018.03.002 Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models

2.LSTM model and its predictions.

The data we are working with is THB/CNY from 2010 to 2020. We have deliberately selected this time period for our purpose as later we will show that in most cases in FTS, the prices or rates are very hard to predict accurately due to their natures. The stock market is highly sensitive to sudden eruptions of major events such as financial crisis, war, public health crisis (e.g., COVID19), etc. By choosing this specific time span, we have avoided the problem of encountering some famous major events like 2008 Subprime Mortgage crisis and 2020 COVID19 outbreak.



We use the time period of 2010 to 2018 as training set, 2018 to 2019 as validation set, 2019 to 2020 as test set.

Scaling is necessary as we are going to pass the data through a neural network, which is supposed to receive scaled data in order to perform well.

To create a data frame of time series data, we also need to get lagged data for a certain span of days, which in our report it is taken as 10, in order to be able to predict the 11th day with the previous 10 days. This data frame is then properly truncated to get rid of the NA values produced by the lagging operations and split into values and targets.

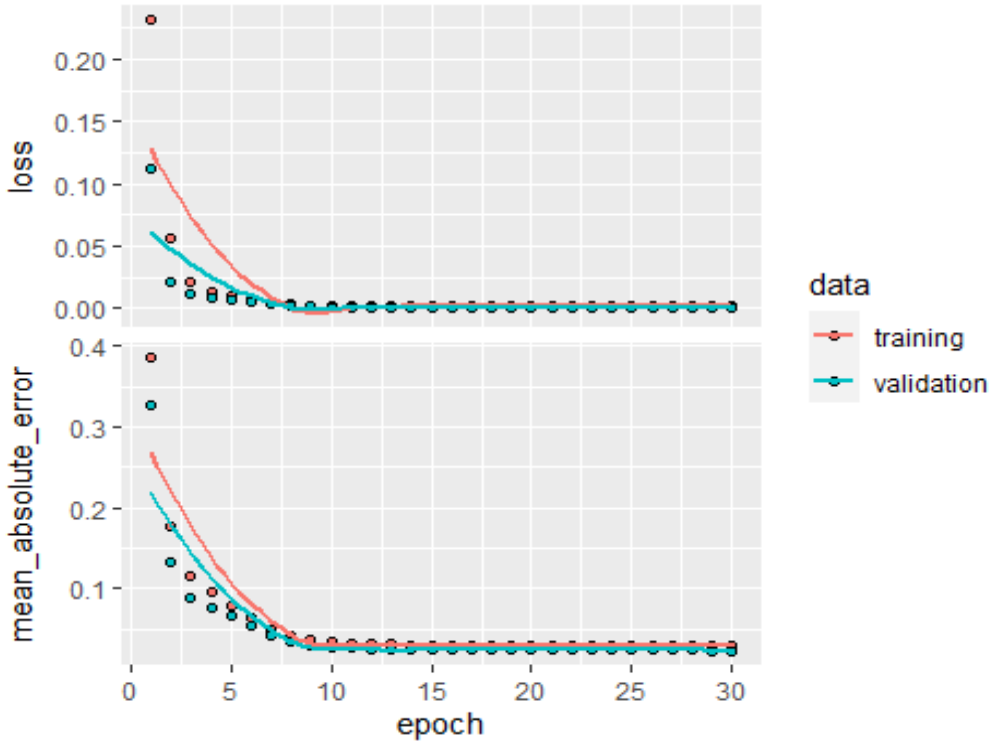
Now we are ready to build the LSTM model for prediction. As we have seen in the reference papers, a common approach is to use evolutionary methods such as Genetic Algorithm to evolve the neural network. However, we are not adopting this method in our report due to the limitation of computational resources at hand. Instead, we select the hyper parameters for LSTM based on our experiences and intuitive of tuning neural networks.

The reason of choosing 8 as hidden units and 30 epochs to train is that we found larger numbers of hidden units like 12 or 16 or even further tend to overfit the training data and the model performs badly on testing data. The number of epochs has a similar effect as well. So, we have chosen rather moderate numbers for these 2 hyper parameters. We don't stack multiple LSTM layers or add dropout layers in between simply because by experiments, we found those kinds of structure don't perform well on this data set.

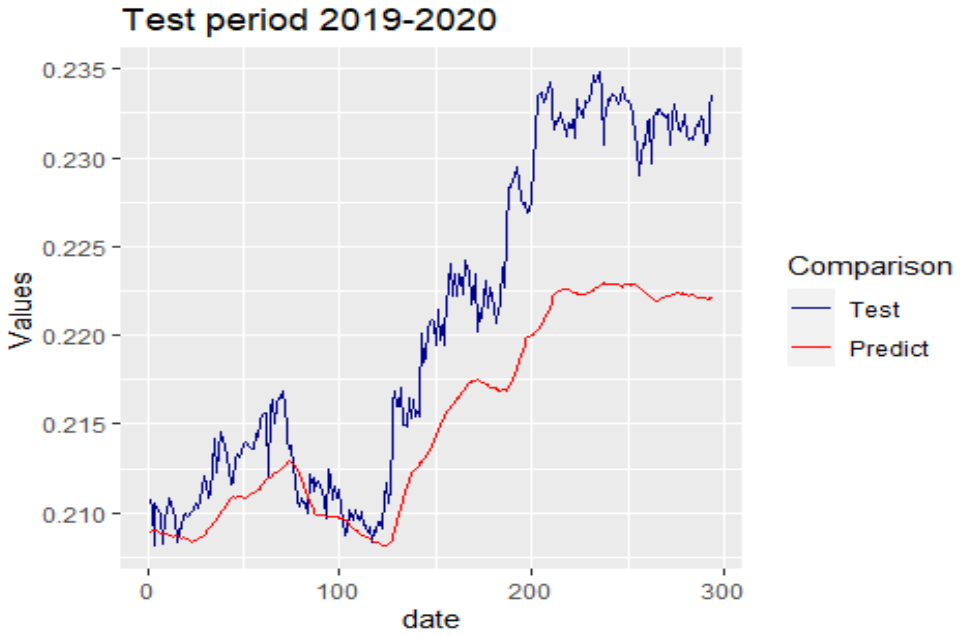
After the training is done, we could have a look at the structure of the LSTM and the training and validation loss curve.

```
summary(model)

## Model: "sequential"
## _________________________________________________________________
_____________
##  Layer (type)                 Output Shape              Param #
## =================================================================
============
##  lstm (LSTM)                  (5, 8)                    320
##
##  dense (Dense)                (5, 1)                    9
##
## =================================================================
============
## Total params: 329
## Trainable params: 329
```

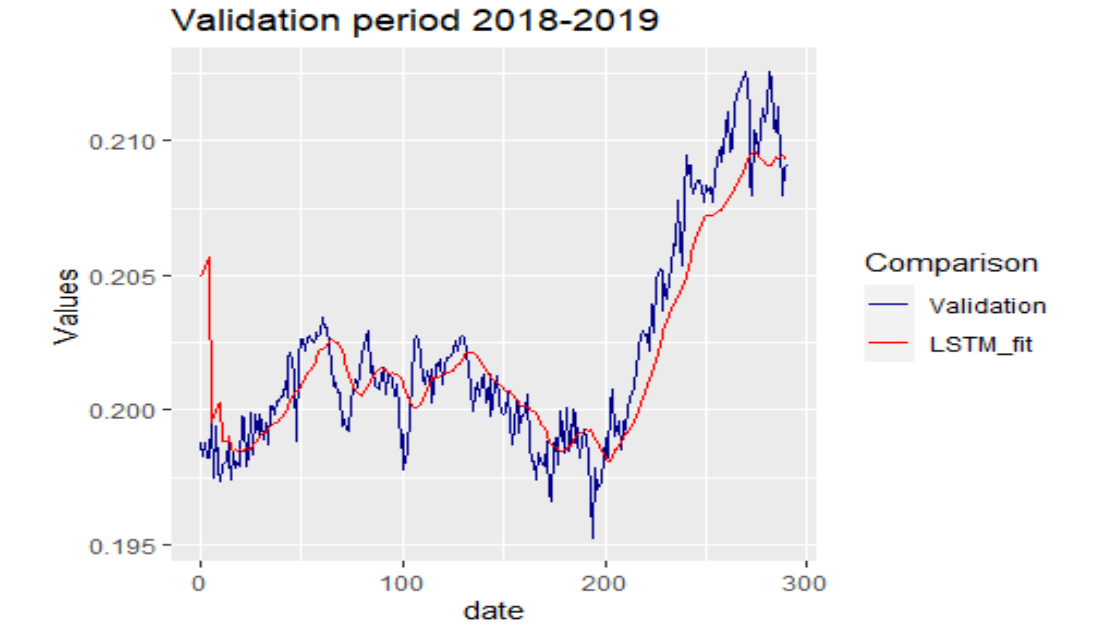
```
## Non-trainable params: 0
## _____________________________________________________________________________

plot(history)
```

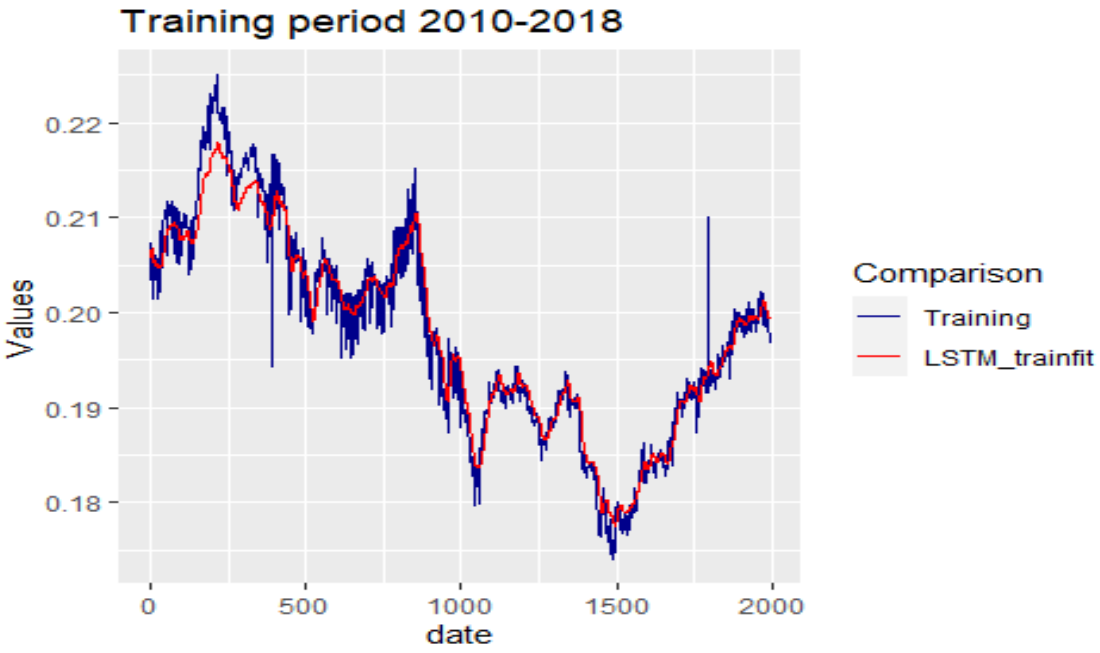


Now we have trained the LSTM, predictions on the test set can be made and plotted.



From the plot we can see the LSTM does not fit well with the test set. There is a wide gap between the ground truth and the predictions. Yet the predictions somehow is able to capture the trends. We will verify this statement by looking at both the numerical and graphical results.

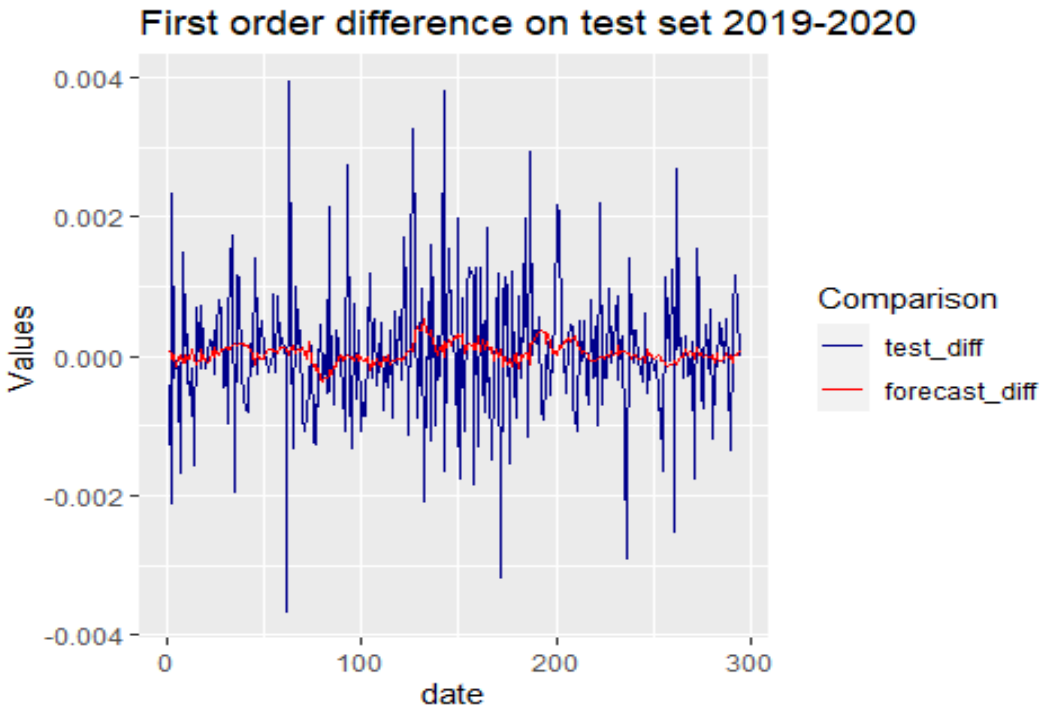


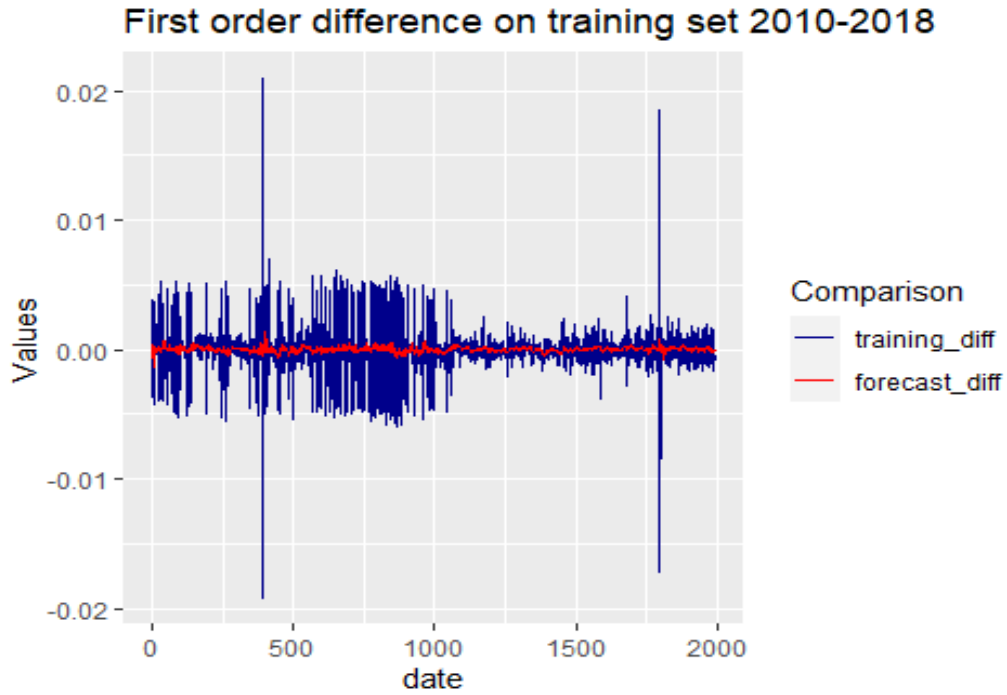


Taking the differences of predictions and that of true values, then we multiply these 2 vectors and get the results, in which greater than 0 stands for the moving direction is the same for prediction and true value, less than 0 for opposite direction, 0 means either prediction or true value is 0.

On training set, the number of correct predictions is 987, that of wrong predictions is: 956. Number of one of the values is: 46.

On test set, the number of correct predictions is: 152, that of wrong predictions is: 135. Number of one of the values is: 7.



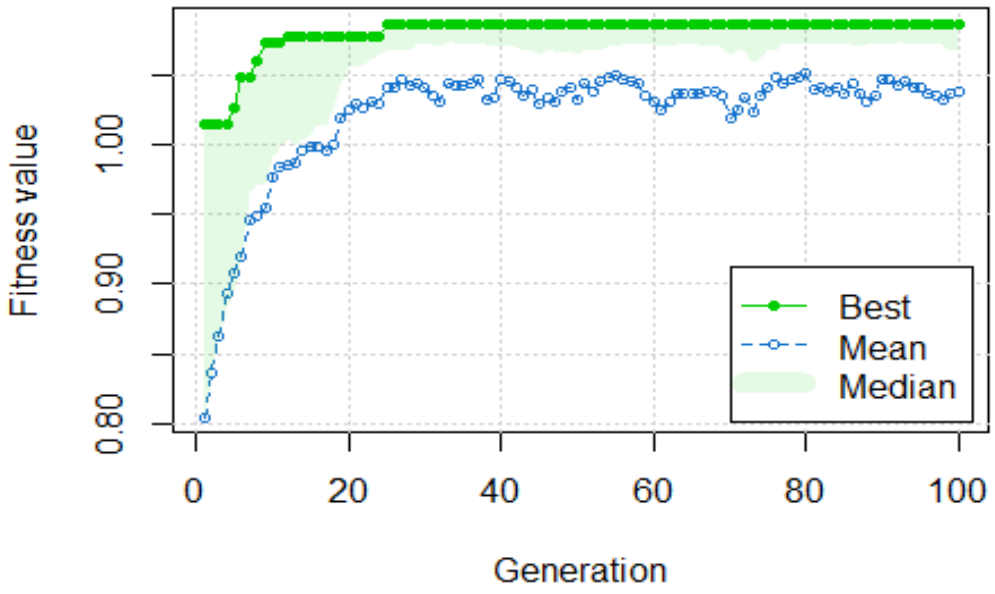


As we mentioned earlier, the graphs illustrate that in real-world trading, the fluctuations tend to be sharper whereas the predictions do not. This potentially showcases there are some other factors outside the stock market that impact the prices massively, such as those major events we made example of earlier.

Although the prediction could not fully capture every bit of the true value, we are likely able to generate some profits on this LSTM model because of the slight differences of right trend predictions and wrong ones.

Next, we apply GA to find the thresholds in the trading rules, named as buy and sell. Our trading rules is quite simple: if the prediction implies tomorrow's price would increase by a certain percentage greater than buy%, we decide to take long position, otherwise if it is less than sell%, we take short position.

3. Utilising GA to evolve trading rules thresholds.




```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                = real-valued
## Population size     = 200
## Number of generations = 100
## Elitism             = 10
```

```
## Crossover probability =  0.6
## Mutation probability  =  0.1
## Search domain =
##       x1   x2
## lower 0.0 -0.1
## upper 0.1  0.0
##
## GA results:
## Iterations           = 100
## Fitness function value = 1.086519
## Solutions =
##             x1          x2
##  [1,] 0.03153749 -0.0008611381
##  [2,] 0.03150523 -0.0008994043
##  [3,] 0.03151900 -0.0008950233
##  [4,] 0.03151640 -0.0008980334
##  [5,] 0.03150345 -0.0009640157
##  [6,] 0.03150587 -0.0010458827
##  [7,] 0.03150678 -0.0010788143
##  [8,] 0.03153792 -0.0008322001
##  [9,] 0.03152204 -0.0007261485
## [10,] 0.03150450 -0.0008340925
## [11,] 0.03152211 -0.0008468926
```

The trading strategy we apply is quite straightforward: we define 2 parameters, namely, buy and sell, as thresholds to indicate whether to buy or sell. If the prediction of next day would be higher to today's closed price than the defined parameter, then we decide to buy. If the prediction of next day would be lower, then we decide to sell. These 2 thresholds are optimized on training data by GA to enable us to get the highest profit.

The fitness function of GA will aim to maximize the profit which is calculated by applying those 2 thresholds parameters on the overall training data. We then apply the thresholds to calculate the profit on test data and we believe the thresholds should work consistently on test data as they are evolved from a quite long period of training data.

The configuration of GA is chosen based on our previous experiences of applying GA. The crossover rate is set relatively low to 0.6 to ensure the population does not change to fast. Mutation rate is set to 0.1, slightly higher than usual as we wish to escape the sub-optimal solution. The search domain is set to a very limited range as we found that by narrowing this range, the search of GA is more efficient at looking for the optimal thresholds in that limited range. Since we are dealing with currency, the fluctuations aren't so fierce as what we would see in stock markets.

The profit evolved from GA is 1.0838186, which verifies our assumption and is very consistent with the profit calculated from training data.

4. Comparison against traditional financial approach.

In this section we introduce a forecasting approach called “Holt-Winters forecasting”. It is used to model the trend, seasonal and irregular components of a time series. Holt-Winters decomposes a time series into 3 mentioned components: that is, estimating these three components.

‘’If you have a time series that can be described using an additive model with increasing or decreasing trend and seasonality, you can use Holt-Winters exponential smoothing to make short-term forecasts.’’ --- A Little Book of R For Time Series, Avril Coghlan (P41)

Holt-Winters uses exponential smoothing to encode past values and use them to predict “typical” values for the present and future.

‘’ Holt-Winters exponential smoothing estimates the level, slope, and seasonal component at the current time point. Smoothing is controlled by three parameters: alpha, beta, and gamma, for the estimates of the level, slope b of the trend component, and the seasonal component, respectively, at the current time point.’’ ---

A Little Book of R For Time Series, Avril Coghlan (P41)

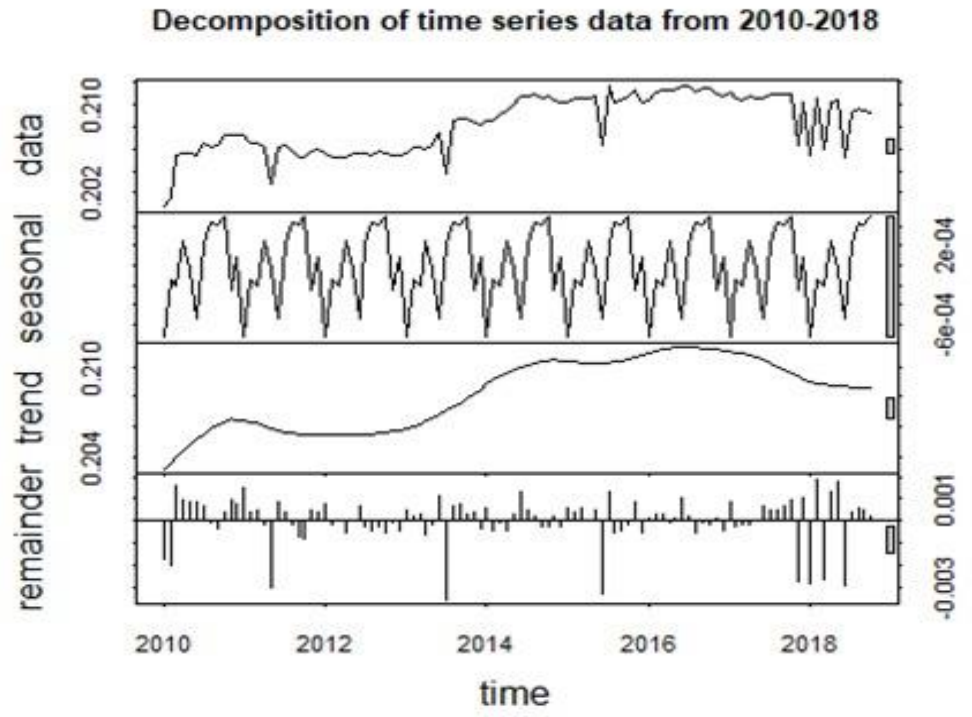
Holt Winter forecast method is contained in R library(forecast), which is very handy to use for this report.

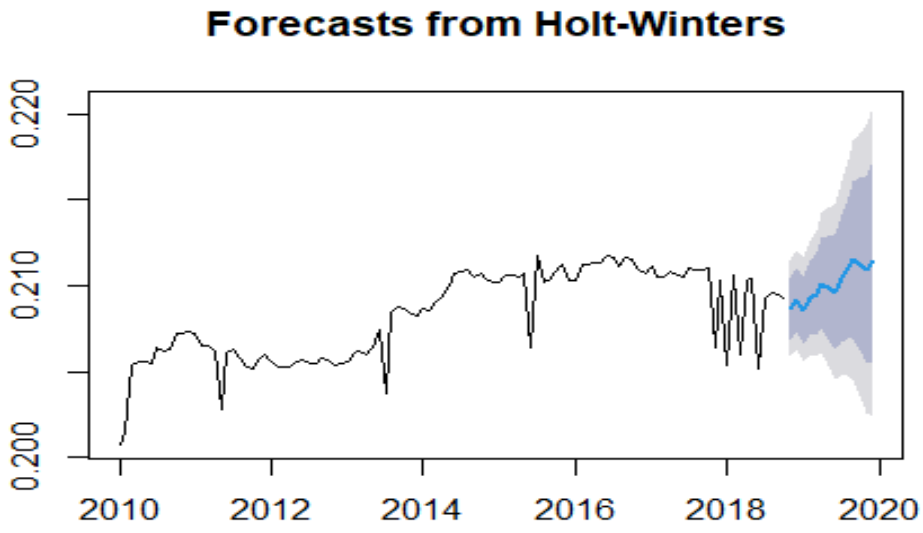
Jan 2010 to Oct 2018 is used for Holt Winter decomposition and predictions are made from Nov 2018 to Dec 2019.

The details of Holt Winter decomposition and forecasting are listed below:

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = ts_data, model = "AAA")
##
##   Smoothing parameters:
##     alpha = 0.1718
##     beta  = 0.1041
##     gamma = 1e-04
##     phi   = 0.9727
##
##   Initial states:
##     l = 0.2023
##     b = 0.001
##     s = 1e-04 -2e-04 3e-04 7e-04 4e-04 0
```

```
##             -6e-04 -1e-04 3e-04 -1e-04 -2e-04 -7e-04
##
##   sigma:  0.0014
##
##       AIC      AICc       BIC
## -879.9242 -872.0621 -831.9823
```





The predicted values form the blue line which shows a general positive trend. The accompanying shadow areas are 80% and 95% prediction intervals.

For the optimizing purpose, we implement Differential Evolution to evolve the trading rules for the Holt-Winter decomposition.

Since Holt Winter is based on monthly return rather than daily, we calculate the profit on both monthly and daily basis in order to be safe and comprehensive.

The profit calculated from Holt Winter prediction on test period's monthly mean closed prices is 1.1350479.

The profit calculated from Holt Winter prediction on test period's daily closed prices is 1.0521621.

5. Discussion about two approaches and reflections on further improvements.

As part of the objectives of this study is to compare the prediction results in daily basis between LSTM model and Holt-Winters approach in terms of the profit calculation. We can obviously see that the profit calculated from LSTM model is 8.38% on test period while the traditional approach gives us profit of 5.21% which is significantly larger than Holt-Winter model but when we saw especially on the monthly basis with Holt-Winter model, it also provides a larger profit of 13.5%, which is greater than that of LSTM. Thus, we can make a conclusion based on the observation that LSTM is a useful technique to predict the future price of stocks and currencies.

However, our comparison is based on different test time divides for Holt-Winter model. The Holt Winter model also fits all the daily training closed prices into monthly prices so it is not very proper to compare Holt Winter with LSTM for these reasons.

Concerning the improvements on LSTM model, some potential threads are:

1. Apply wide-deep model, which enables us to mix low-order features to later generated features. This is the mainstream method to improve the performance of Neural Network.
2. Add embedding layer before LSTM model is a common practice when using RNN to deal with text and image data. Whether this is applicable to time series data remains to be explored.
3. Introduce attention mechanism to the LSTM model, and potentially stack multiple attention layers.
4. Replacing LSTM with TCNN and Dense layer with Transformer.
5. As mentioned earlier, we can optimize LSTM with GA, which is not done in our report due to the limitation of computation resources.