Assignment 4 - Neural Net prediction based trading for Zoom Video Stocks

28/03/2021

Introduction

Given the wide changes in the financial sector due to the 2020 crisis we decided to create a model to predict a stocks closing value for the next day. These predictions would then be fed into a trading function that decides if the trader should buy, sell or hold their stock with the aim to achieve the highest possible profit.

For this task, we present the use of a Neural network to predict closing price of the financial asset of Zoom Video Communications. This asset received a huge upsurge in value due to the 2020 crisis and is thus of interest to look at. The neural net, with 3 hidden layers, will be trained on a variety of financial indicators from the 2020-2021 trading period and tested on the 2021 data.

A genetic algorithm (GA) will be used to tune the hyperparamters for the neural net, namely the number of nodes to implement per layer. Design of a neural net's architecture is a non-trivial task so evolution based approaches could aid in finding an optimal configuration. We only applied evolutionary approaches for determining the number of nodes that yields the best Root Mean Square Error (RMSE) in the test data, although it could be expanded to examine other parameters such as epochs and how many layers are required.

Having created a series of predictions for the 2021 trading days, a genetic algorithm will also be used to generate the trading rules for when to buy or sell the Zoom asset. Using historic data to inform future decisions is key and we find that coupling these techniques leads to a net profit. These GA tuned trading rules will be compared to a simple 'mean' model to compare how each fair.

Related work

For some inspiration on our predictive model, we found "Forecasting stock movements with Artificial Neural Networks in R" (Paul Rivera,28/05/2018) to be helpful as a guide.

In this forum, the author trains a simple 2 layer neural network to predict the movement of six stocks traded in the NYSE and NASDAQ using some technical indicators. Some issues encountered in this example were the random change of hyperparameters and numbers of layers and neurons. We believed that evolutionary approaches could expand on the technique they demonstrated, and so chose GA to help refine on the work they presented.

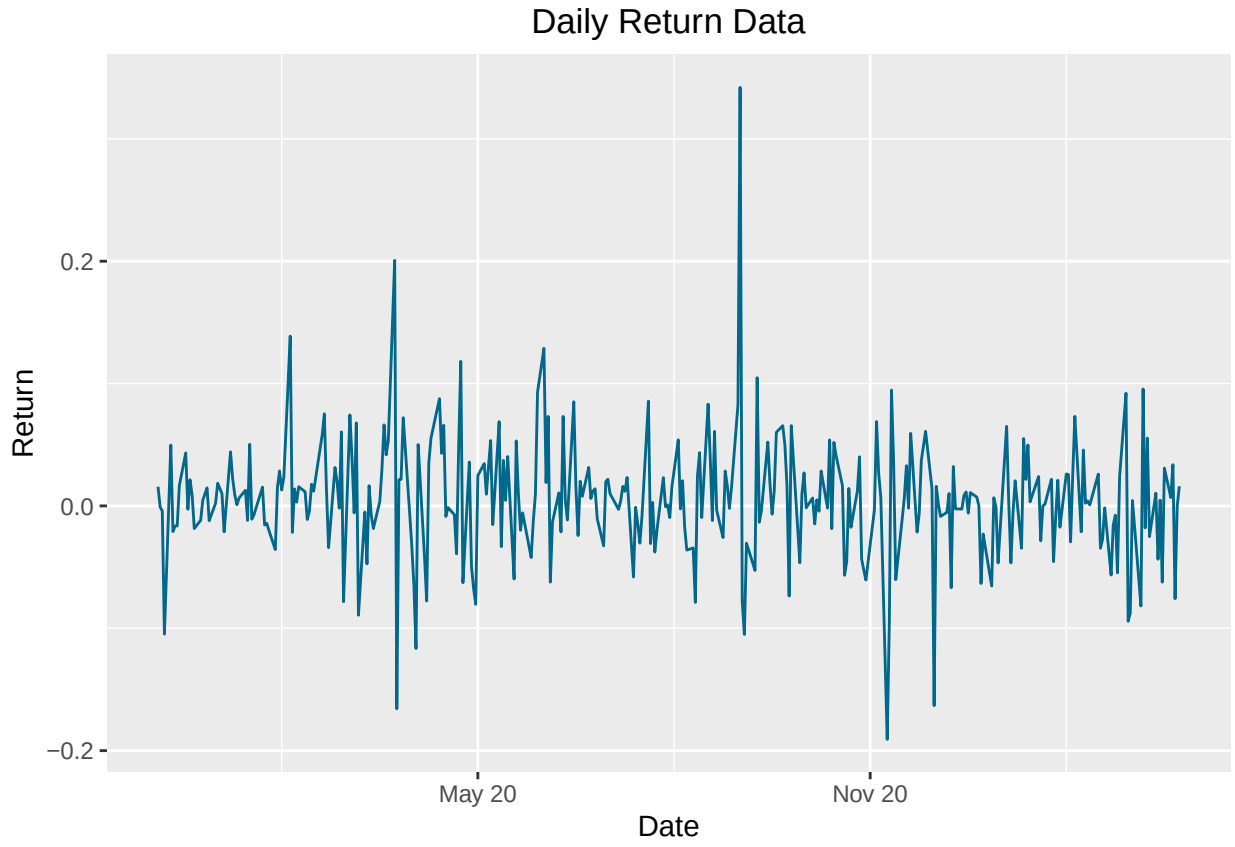
Additionally we sought to predict the closing price, rather than just the direction of the stock, and use that as an indicator that a trader ought to buy/sell/hold.

Overview of the data

Examining the closing price time series data, we can see many short term fluctuations accompanied by long term tends. As inputs for our neural net we want to include both the short and long term patterns for it to make its estimations. Several different attributes of the data have been examined and included for the model, such as the Daily returns, RSI, and EMA.

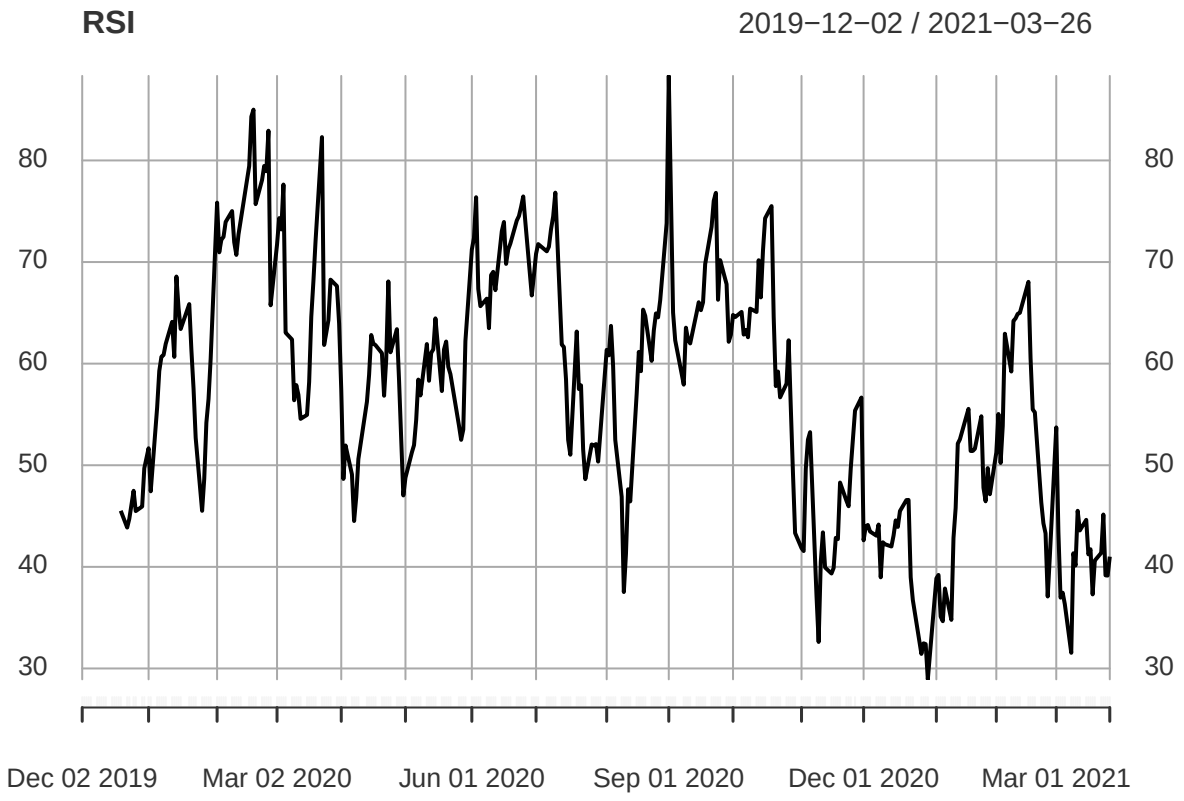
Daily Returns

Daily returns calculate the division between today's closing price and yesterday's, this results in a percentage of close price variance. Used as an input to the neural net it is a short term indicator of performance, capturing short term patterns for the neural net to learn from.



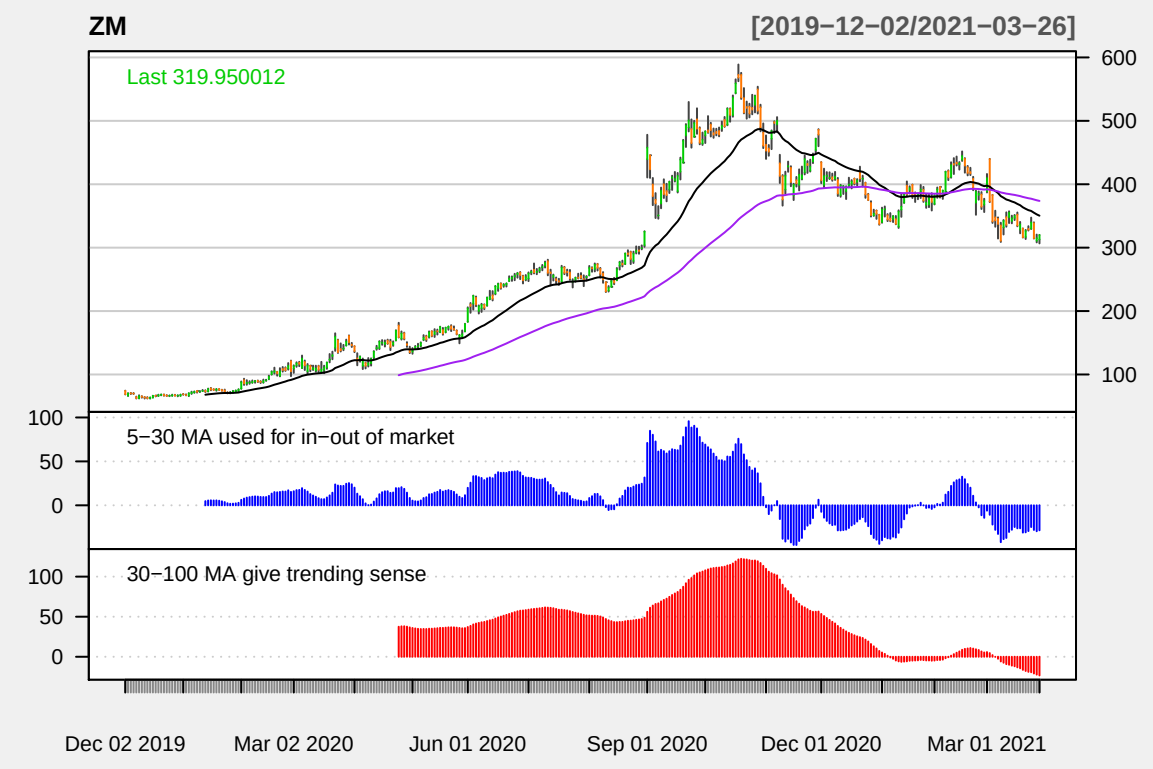
RSI

Relative Strength Index (RSI) is used to calculate the momentum rate in the change of prices, this would be insightful data inputs for our predictor as a leading indicator for identifying a trend reversal.



EMA

Exponential Moving Average (EMA) can be used to smooth out short term fluctuations and highlight longer-term trends, we used EMA with windows of 7 and 14 days to train our Network. Looking at different windows we realized that the narrower the window, the more reactive the results. The graph below shows the EMA with different lags, the black curve is using a window of lag 30 days and the purple curve is using one of 100 days. Below this, the blue graphs represent a 'slow difference', in blue, and a 'fast difference', in purple, done by subtracting a 5 day lagged window frame to the 30 day window, and by subtracting the 30 window from the 100 window, respectively. The purpose of this was to understand how EMA reacts to different parameters in window sizes.



Zoom video only started it's journey in trading on the 18th of April of 2019, therefore there is not a huge amount of historical trading data for this asset, using lags of up to 100 days would require a trade off for amount of data (which we then also have to split between training and testing sets), so we decided to include in our model an EMA of 7 days and another of 14 days to loose as little data as possible but having insight of more than one parameter variance.

Neural Net Predictive-Model

To get the RSI and EMA values for all of 2020, we imported the data from 2019 and then sliced this off to train the model on 2020 data. We then tested our model on the available data from 2021.

```
NN_data<- data.frame(as.xts(merge(RSI[,1],
                                  ema14[,1],
                                  ema7[,1],
                                  Lag(myRetData[,1],1),
                                  Lag(ZM[,'ZM.Open'],1),
                                  Lag(ZM[,'ZM.Open'],8),
                                  Lag(ZM[,'ZM.Open'],15),
                                  Lag(ZM[,'ZM.Close'],8),
                                  Lag(ZM[,'ZM.Close'],15),
                                  ZM[,'ZM.Close']
                                  )))

NN_inputs<-c('RSI','EMA14','EMA7','DailyRet','Open','Open7',
             'Open14','Close7','Close14','TMClose')

colnames(NN_data)<-NN_inputs
NN_normal<-normalize(NN_data,range=c(-1,1))
NN_scaled<- data.frame(NN_normal[,1:9],
                       Close=Lag(NN_normal['TMClose'],1),
                       NN_normal[,10])

colnames(NN_scaled)<-c('RSI','EMA14','EMA7','DailyRet','Open','Open7',
                       'Open14','Close7','Close14','Close','TMClose')

NN_scaled<-NN_scaled[22:332,]

NN_train<-NN_scaled[1:253,] #From 2020 to 2021
NN_test<-NN_scaled[,1:10][254:311,] #2021  test data
NN_expected<-NN_scaled[,11][254:311] #2021 closing prices scaled
```

Optimisation of a Neural Net using GA

The architecture of a neural net is one of the key parameters to look at when trying to improve performance. With the use of too few nodes, it can't capture the patterns from the input. However, with too many nodes or epochs, it can begin to overfit the training set, which would result in an increase in performance while training but the results would suffer in the testing set.

Here we are using the GA program to search for the optimal number of nodes in a dense, 3 hidden layer neural net trying to predict tomorrow's closing price.

The fitness function takes in the 'solution' from the GA, in the format of a list of real numbers. We were unable to find how to make this solution an integer within the GA so it was rounded to the nearest integers. These three integers were used to create a neural network, which is trained on the features from 2020 (RSI, EMA7, etc), and then tested on the 2021 data. The evaluation of that neural net is the Root Mean Square Error (RMSE) of the predictions for 2021 close price and the actual values of the closing price.

The GA tries to maximize the fitness function, so -RMSE was used, for which the 'best' result would be 0 error and the predictions matching the expected.

```
fitness_NN<- function(nn) {
  nn<-round(nn)
  zoom.nn <- neuralnet(TMClose ~ RSI+
                         EMA14+
                         EMA7+
                         DailyRet+
                         Open+
                         Open7+
                         Open14+
                         Close+
                         Close7+
                         Close14,
                       NN_train,
                       hidden=(c(nn)),
                       stepmax=1e6,
                       linear.output = TRUE,
                       rep=5)

  zoom.output = compute(zoom.nn, NN_test)
  fit<--rmse(NN_expected,zoom.output$net.result)
  return(fit)
}
```

There are some constraints on how extensively we could build and test Neural Networks via GA program. The number of epochs was set to be 5, the GA popsize to 25, and the max iterations to be 10. This was because the time taken to run through the epochs, population, and iterations, was significant and must be run over night to gain a result.

The search space for integers of 3 digits from 1-10 is only 1000 but the GA allowed us to search through that space effectively without manually building and making each one.

```
GA <- ga(type = "real-valued",
         fitness = fitness_NN,
         lower=c(1,1,1),
         upper=c(10,10,10),
         popSize = 25,
         maxiter=10,
         monitor=FALSE)

plot(GA)
summary(GA)
```

GA Results

```
-- Genetic Algorithm -------------------

GA settings:
Type                  =  real-valued
Population size       =  25
Number of generations =  10
Elitism               =  1
Crossover probability =  0.8
```

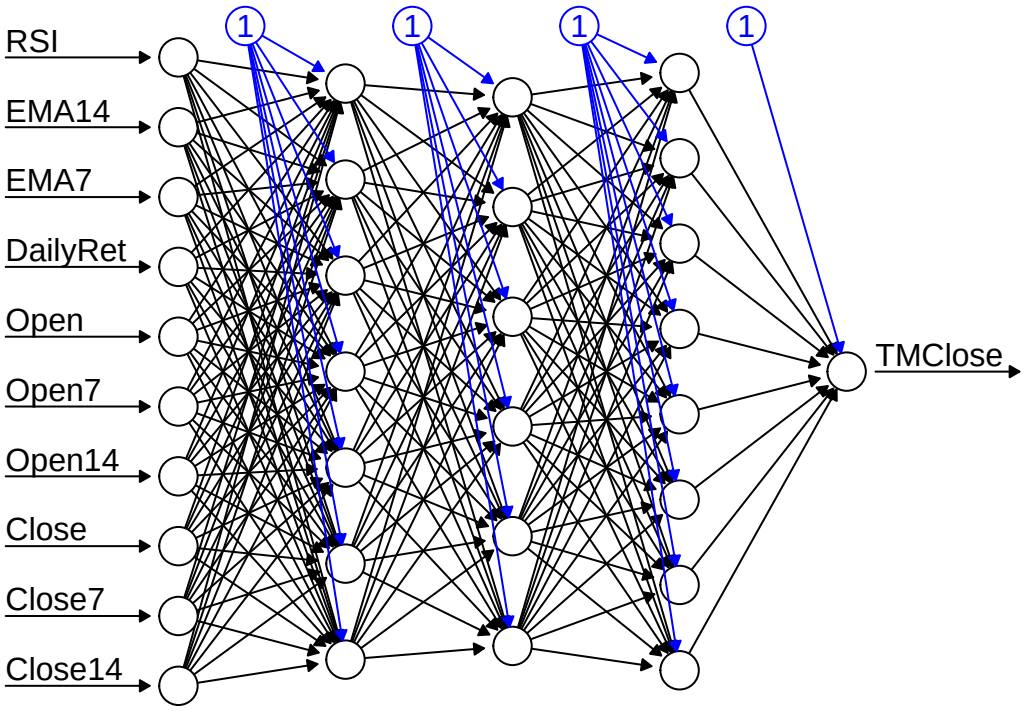
```
Mutation probability  =  0.1
Search domain =
      x1 x2 x3
lower  1  1  1
upper 10 10 10

GA results:
Iterations             = 10
Fitness function value = -0.0734826
Solution =
           x1       x2       x3
[1,] 7.422438 6.356126 8.315569
```

So rounding the GA found the our neural network should consist of 3 layers of 7,6,and 8 nodes. The fitness function for this over the test data shows an RMSE of 0.0734 which is acceptable given the range of the scaled data.

Building the Neural Net

Having run the GA to get the architecture of our neural net we can build and visualize it. We can look at the error over the training data for the 5th epoch.



It has an error rate of ~0.08 which is in line with the RMSE obtained from our fitness function. This would seem to indicate that the model isn't overfitted to the training data but we can compare this directly.

```
zoom.output = compute(zoom.nn, NN_test)

errors = zoom.output$net.result - NN_expected
```

```
results = data.frame(zoom.output$net.result, errors)
outputs <- cbind(NN_expected, results)
colnames(outputs) <- c("Expected O/P", "Neural Net O/P", "Errors")
print(outputs[1:20,])
```

```
##            Expected O/P Neural Net O/P        Errors
## 2021-01-04    0.6339310      0.8857476  2.518167e-01
## 2021-01-05    0.6398638      0.8450247  2.051609e-01
## 2021-01-06    0.5252521      0.6215305  9.627839e-02
## 2021-01-07    0.5119203      0.6219969  1.100766e-01
## 2021-01-08    0.5615481      0.5928902  3.134215e-02
## 2021-01-11    0.4784859      0.5177245  3.923861e-02
## 2021-01-12    0.6118042      0.7491346  1.373303e-01
## 2021-01-13    0.6663880      0.7515119  8.512384e-02
## 2021-01-14    0.7962860      1.0181015  2.218156e-01
## 2021-01-15    0.8052903      0.9361662  1.308759e-01
## 2021-01-19    0.8704836      0.8704082 -7.534394e-05
## 2021-01-20    0.7932148      0.6717417 -1.214730e-01
## 2021-01-21    0.7932846      0.6636484 -1.296362e-01
## 2021-01-22    0.7974028      0.6291742 -1.682286e-01
## 2021-01-25    0.8562443      0.7195547 -1.366895e-01
## 2021-01-26    0.7345130      0.6309681 -1.035450e-01
## 2021-01-27    0.7090358      0.7263289  1.729315e-02
## 2021-01-28    0.7638289      0.7951881  3.135911e-02
## 2021-01-29    0.7183193      0.7359604  1.764109e-02
## 2021-02-01    0.7871422      0.8146900  2.754785e-02
```

```
print(c('RMSE for the Neural Net predictions with the true values:',rmse(NN_expected,zoom.output$net.res
```

```
## [1] "RMSE for the Neural Net predictions with the true values:"
## [2] "0.155606471254235"
```

The RMSE of the neural net is about double what the fitness function predicted, possibly due to difference in seed/ random number generation at the neural net creation. Looking at the outputs, some of the predicted values are significantly different from the expected. Given that uncertainty we will need some robust trading rules to try and capitalise on this potential information on the future.

Trading rules via GA

Now we have made some predictions we need to see if we can make them work for us and make some profit from our investment.

We need to generate trading rules to decide to buy when the predicted price is a certain amount higher than 'todays' closing price, and to sell when it's a certain amount lower. This is another problem that the GA program lends itself towards and we can search for a value that maximises profit. The fitness function will simply be the sum of the costs of buying and the rewards of selling over the test period.

Firstly we need to make a table of the true closing prices for 2021 and another containing the close price and predicted price.

```
close_price<-ZM[,'ZM.Close'][274:330]

Trading_dec<-data.frame(NN_test['Close'],zoom.output$net.result)
T_names<-c('Close','Predicted')
colnames(Trading_dec)<-T_names
Trading_dec[1:20,]
```

```
##                Close Predicted
## 2021-01-04 0.4757638 0.8857476
## 2021-01-05 0.6339310 0.8450247
## 2021-01-06 0.6398638 0.6215305
## 2021-01-07 0.5252521 0.6219969
## 2021-01-08 0.5119203 0.5928902
## 2021-01-11 0.5615481 0.5177245
## 2021-01-12 0.4784859 0.7491346
## 2021-01-13 0.6118042 0.7515119
## 2021-01-14 0.6663880 1.0181015
## 2021-01-15 0.7962860 0.9361662
## 2021-01-19 0.8052903 0.8704082
## 2021-01-20 0.8704836 0.6717417
## 2021-01-21 0.7932148 0.6636484
## 2021-01-22 0.7932846 0.6291742
## 2021-01-25 0.7974028 0.7195547
## 2021-01-26 0.8562443 0.6309681
## 2021-01-27 0.7345130 0.7263289
## 2021-01-28 0.7090358 0.7951881
## 2021-01-29 0.7638289 0.7359604
## 2021-02-01 0.7183193 0.8146900
```

We define a function which takes in a list of 2 numbers; the bounds on which the decision to buy or sell will be made. The ratio of the predicted close /close will be what the decision is made on and will be the values that the GA will be used to try and optimise.

```
T_Descision<- function(list){
  Trade<-c()
  counter<-0 #We can't sell an asset we haven't bought so a counter to keep track is needed.
  for (i in 1:57) {
    close<-Trading_dec[i,1]
    expected<-Trading_dec[i,2]
    if (expected/close>list[1]) {
      Trade<-c(Trade,1)
      counter<-counter+1 #increase stock counter for buying.
    } else if (expected/close<list[2]) {
      if (counter>0) {
        Trade<-c(Trade,-1)
        counter<-counter-1 #if we have stock, sell and decrease.
      } else {
        Trade<-c(Trade,0) #if we don't have a stock, hold.
      }
    } else {
      Trade<-c(Trade,0) #if neither condition is satisfied, hold.
    }
  }
```

```
  return(Trade)
}
```

The function to check how much profit we make is simple, roster through the trade list from our decision function and if it's a 1 then buy or -1 then sell. A buy reduces our capital by the close price of that day and a sell increases it by that amount. It returns the simple sum taken over that period, making it a clear option for a fitness value to try and maximise.

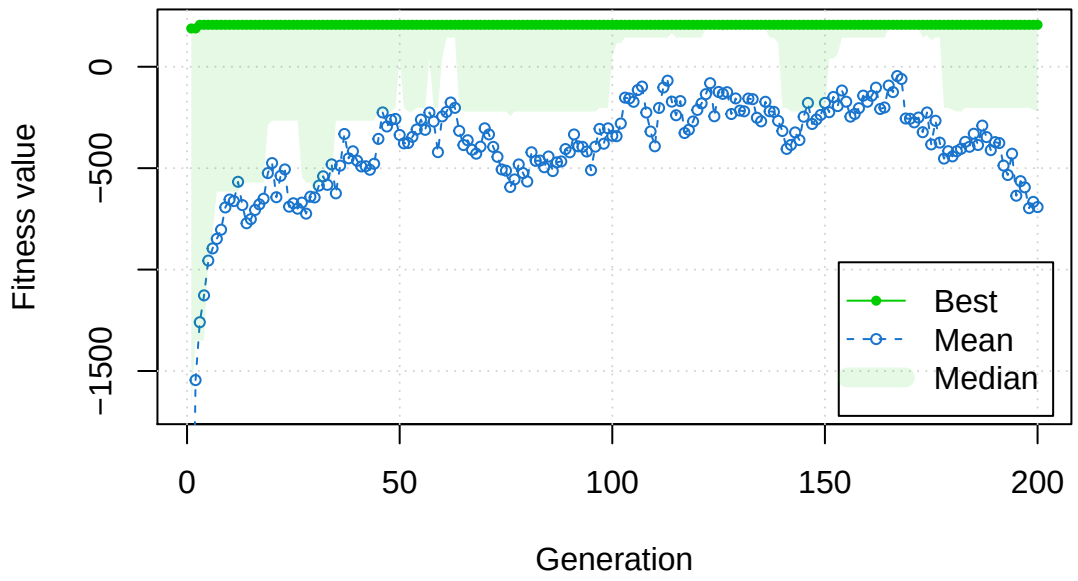
```
profits<-function(list){
  profit<-c()
  for (i in 1:57) {
    if (list[i]==1) {
      profit<-c(profit,-close_price[i])
    } else if (list[i]==-1) {
        profit<-c(profit,close_price[i])
    } else {
        profit<-c(profit,0)
    }
  }
  return(sum(profit))
}
```

Building the GA function we can be a little more rigorous than we were for the neural net as it can run significantly faster. The popsize and iterations were increased significantly to search for a solution that would yield profit for the trading rules.

```
Fitness_Trade<-function(bounds) {
  return(profits(T_Descision(bounds)))
}

Trade_GA <- ga(type = "real-valued",
         fitness = Fitness_Trade,
         lower=c(1,0),
         upper=c(2,1),
         popSize = 200,
         maxiter=200,
         monitor=FALSE)

plot(Trade_GA)
```




```
summary(Trade_GA)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  real-valued
## Population size       =  200
## Number of generations =  200
## Elitism               =  10
## Crossover probability =  0.8
## Mutation probability  =  0.1
## Search domain =
##       x1 x2
## lower  1  0
## upper  2  1
##
## GA results:
## Iterations             = 200
## Fitness function value = 206.82
## Solutions =
##             x1        x2
## [1,]  1.512124 0.8786377
## [2,]  1.519693 0.8777821
## [3,]  1.516726 0.8830207
## [4,]  1.517846 0.8780962
## [5,]  1.504932 0.8786782
## [6,]  1.518976 0.8787177
## [7,]  1.515196 0.8801936
```

```
## [8,]  1.517408 0.8813594
## [9,]  1.468057 0.8899637
## [10,] 1.522671 0.8775230
##  ...
## [30,] 1.506281 0.8772570
## [31,] 1.512667 0.8785679
```

We can interpret these solutions on how much confidence we ought to have in our predicted values.

Evaluating one of the solutions:-

```
## [1] "Profits"    "206.820009"
```

```
## [1] "Stock remaining:" "0"
```

It can be seen that this generates a profit! It can already be thought of as a success from a materialistic perspective. Looking at the sum of the trade decision we can see that there are also currently no stocks held in hand. Now we ought to compare it to another model to see how well it's performing.

Simple Model

To compare our neural network to a very simple model, we created a predictor that uses the 2 previous days closing price and averages them, using this as the estimation for today's closing price. Then, if the estimation is higher than the real closing price for that date, this would mean the price is lowering, so would advice to sell. However, if the estimation is lower than the real price for "today", this would mean the price is increasing, and then would advice to buy.

```
##            Closing Lag 2 Closing Lag 1 Estimation Real price
## 2019-12-04         68.93         70.02     69.475      69.96
## 2019-12-05         70.02         69.96     69.990      69.67
## 2019-12-06         69.96         69.67     69.815      62.74
## 2019-12-09         69.67         62.74     66.205      65.94
## 2019-12-10         62.74         65.94     64.340      64.57
## 2019-12-11         65.94         64.57     65.255      63.52
```

To calculate the profitability of this model, a similar system than the one for the neural net was used. Using the closing prices from our testing data, we would calculate the profits from buying and sharing according to this decision maker.

```
##            Closing Lag 2 Closing Lag 1 Estimation Real price Trade
## 2019-12-04         68.93         70.02     69.475      69.96     1
## 2019-12-05         70.02         69.96     69.990      69.67    -1
## 2019-12-06         69.96         69.67     69.815      62.74    -1
## 2019-12-09         69.67         62.74     66.205      65.94    -1
## 2019-12-10         62.74         65.94     64.340      64.57     1
## 2019-12-11         65.94         64.57     65.255      63.52    -1
```

```
#Calculation of profit - 251 is the last value of 2020
money <- 0
shares <- 0
for (i in 251:309){
```

```
  if (df[,"Trade"][i]==1){ #If you "buy"
    money <- sum(money, -(df[,"Real price"][i]))
    shares <- sum(shares,1)}
  else if (df[,"Trade"][i]== -1){
    if (shares >= 1){
    money <- sum(money, (df[,"Real price"][i]))
    shares <- sum(shares, -1)}
    }
}

#Total profit will be the money from prior trades plus
#the shares currently held (according to today's closing price)

todays_closing <- c(df[,'Real price'][309])
money_in_shares <- todays_closing * shares
profit <- sum(money_in_shares, money)

print(c('The estimated profit from a simple average model:',profit))
```

```
## [1] "The estimated profit from a simple average model:"
## [2] "-573.159823"
```

It can be seen that the profits from a trader using simple rules along with only averaging data isn't able to turn a profit and is operating at a significant loss.

Conclusions

The built neural network had some excellent results with a very low RMSE. Some more tuning of other hyperparameters, like number of epochs and layers, could have resulted in an even lower RMSE. However, the computing time of this algorithm is currently very high, therefore this would be a matter of trading off faster computing for higher accuracy.

As can be observed above, the simple comparison model would have resulted in a loss of around 573 USD. In comparison to our neural network, which outputted a profit of around 152 USD.

In conclusion, this neural network model can accurately predict, with an RMSE of around 0.1-0.2, Zoom's closing price of the next day. Combining this and our trading rule, the profit output for this model is very satisfactory. Of course, the use of GA to optimize the number of neurons, upper bound and lowers bound for the estimator, have played a key role in the accuracy and profitability results of this algorithm.