# **Introduction**

<https://github.com/Shadman191411903/UTS_ML2019_ID13000029>

I have always been curious about stock markets and the way they behave. A lot of studies and research has been undertaken to accurately predict whether the market would go up or down. According to the Semi-Strong market efficiency; stock prices includes all public information. This definition is true for most of the top stock markets we see in today’s world with serious implications for individuals who has tried to do insider trading and have more knowledge than what the public already knows. This leaves us; small time investors, speculators as well as traders to invest in this market having the same information as everyone else.

With this wealth of public information of stock around us, people usually invest in these markets by analysing either the key financial ratios of the company(fundamental analysis); or looking at historical data and market statistics(technical analysis). More recently, sentiment analysis such as looking at company’s activities through news or social media has gained popularity(market sentiment analysis). While each way of analysing has their own merits I chose the technical analysis as my method for forecasting and analysing particularly because throughout the day a stock price follows simple demand and supply rules. if I can identify and learn how the stock prices behaved in the past according to market patterns and buy and sell indicators I can form and accurately predict of future price trajectories to some extent.

The way I chose to build a forecasting model to predict stock market prices is to use genetic algorithm and evolution strategies. The rest of the report is organized as follows; Section 2 describes the intuition and the development of the model. Section 3 describes the algorithm implementation. Section 4 describes the results achieved using the model and looks at some comparative results with other algorithms. Section 5 proposes a back testing strategy to maximise our returns provided the model is used in the future. Section 6 discusses some challenges encountered and how those were overcame. Section 7 looks at an ethical aspect of this model. Finally the report concludes itself by looking into improvements in the model and suggestions for the future.

# **EXPLORATION**

After deciding that I want to develop a model that predicts stock market prices it was clear to me I needed to find a machine algorithm that can minimise the difference between the prices on consecutive days based on technical indicator features. I came across genetic algorithm while learning about various optimization algorithms.

Genetic algorithms generally are classified as adaptive methods that are used to efficiently solve search and optimization problems. Since I was always fascinated how nature selects the ‘most fittest individual’ I believed from the start that applying a genetic algorithm would mean I can select the best set of technical features that would allow me to predict the price of the stock next day.

Given I am trying to perform technical analysis based on technical indicators for example Relative Strength index, Simple moving averages, Momentum, etc. I needed to find a function that would help me minimise these indicators(fitness function). Similarly like the genetic process these indicators would be the genes and the combination of these indicators would be a chromosome. Taking a further step forward, the set of technical features would be the genotype and finished product would be the phenotype. Since the phenotype is measure of the individual’s fitness the chromosomes make up the phenotype we can apply a fitness function on the chromosomes that minimises the difference between actual and predicted price. The fitness function should be unique and would be the one that tries to minimised the price today, and the next day based on these technical features.

I researched various frameworks that in python that would help me build a genetic algorithm model. I chose the DEAP model which stands for “Distributed evolutionary algorithms in python(DEAP)”. The reason for choosing the DEAP model was because I failed numerous times to replicate the genetic framework in the context of stock prices with other libraries. With DEAP:

* I could define my custom types and was not limited to predefined types.
* DEAP does the various initializations that requires me to replicate the genetic process, for example the number of individuals, the mutation rate, the crossover etc.
* It gave me more control as I need to focus only on the fitness and evaluation functions and data structures transparent.

Now that I have a framework and know how to implement it I just needed to find helper functions that would calculate the technical features for me. The module I chose was Quant software tool kit (QSTK) which is now deprecated and no longer is used and also cannot be installed. Getting these helper functions meant that I longer needed to worry about the accuracy of the result of calculating these technical features.

Finally the stock market data was taken from yahoo finance. The data is public information and by all known sources deemed to be accurate.

# **Methodology**

## **Base Model**

The first step is to get data into our model. I had tried this model with stocks listed in the Australian stock exchange. I took the training data set from the January 2014 to 12 December 2018. The main reason for selecting this training period is that before 2011 the world was experiencing financial crisis and post 2011 the stock markets recovered. Hence, I believed the price today would be a better reflection if I trained during the aforementioned period. The technical features selected in our base model were Relative Strength Index, Simple Moving Average and Momentum (for more information about these technical features kindly go this site <https://www.investopedia.com/>). Relative Strength index is chosen as an indicator because it measures the magnitude of recent prices changes to evaluation to overbought and oversold conditions. Since price of a stock is determined by demand and supply and markets automatically adjusts themselves this magnitude can be good indicator if the stock is being overbought (higher RSI values) and is likely to go down and vice versa. Momentum indicator is a kind of predictor by itself. It measures the current price in relation to what its price was in the past. It gives us the idea that if prices were higher in that period did it further increase the price or decreased it. Lastly the simple moving average is a buy or sell trigger. Generally if the price is above the simple moving average we buy as we expect the stock price to decrease in the future and vice versa.

Given our inputs and the features the base model forecasts the price of the next (n) days as following.

**[Step 1]** Enter the name of the stock you want to predict this can be any stock in the Australian stock exchange or anywhere in the world. This goes in the stocks array in the BaseModel.ipynb in the A2 folder.

**[Step 2]** Go to yahoo finance, and enter the stock name you put in Step 1 there. You can specify the time period. As mentioned above I took the training period from 1st Jan 2014 to 12 December 2018. Download this as a csv file and stored it to your desired folder. Rename the csv file to your StockName\_training.csv. For example if it is ASX.AX.csv it should be ASX.AX\_training.csv. You might need to change the date in the training set to YYYY-MM-DD format.

**[Step 3]** Go to yahoo finance again and take at least the recent thirty days of the stock you entered in step 1 above (Since some indicators has loopback period of 30 days). Save this data as a csv file and stored it to your desired folder. Rename the csv file to your StockName\_current.csv. For example if it is ASX.AX.csv it should be ASX.AX\_current.csv. You might need to change the date in the training set to YYYY-MM-DD format.

**[Step4] (Optional)** In step 3 you can remove the last five days and put it in another file called StockName-Test.csv; for example if remove the last five dates from the ASX.AX\_current.csv and make a new csv file called ASX-Test.csv and paste the last five dates you can compare how the algorithm predicted over the real prices on those given dates. However this step is purely from testing the accuracy of the model and is not needed. You might need to change the date in the training set to YYYY-MM-DD format.

**[Step5]** That’s it the model does all the computations and predicts the price over the next five days and stores it in a csv file called stockname.baseModel.csv. Note if doing reruns you must click the restart the kernel and previous values will persist and give an inaccurate result.

### **Underlying architecture of the base model-Training**

1. The closing prices are taken, which is the last price that stock was traded before the market closed on that day.
2. The differences in RSI, Momentum and Simple moving average over consecutive days are taken.
3. For the base model stopping criteria which in the DEAP model is the number of generations is taken to be five; which means the most fittest individual will be taken from five iterations.
4. The weightage for the fitness function is set to -1 since we want to minimise the fitness function.
5. The fitness function is defined like this.

I used <https://math.typeit.org/> to format the equation nicely.

A screenshot of a cell phone

Description automatically generated

1. The goal is to find these coefficients. After getting these coefficients the fitness function is added to the closing prices.
2. Error= Price the next day(day2) – (the price calculated today + fitness function values).

The reason being technical indicators with their values are likely to affect how the stock was traded today and if the same set of features were applied what would the price be tomorrow. The difference in prices tomorrow i.e actual and predicted is something the model will try to minimise. Since we would not know the price tomorrow, but we know the prices today as well as the technical features over a certain period, the model can try its best to learn what values of features accurately predict the price tomorrow.

### **Underlying architecture of the base model-Testing and evaluating**

1. DEAP models calculate the fittest individual for us; in our case it the fittest individual is the best set of technical features that gave the lowest error described above. Predicted Price is calculated as follows.

A screenshot of a cell phone

Description automatically generated

## **Extended Model V1**

After the successful development of the base model it was time to further extend our base model by now trying out different parameters of the DEAP library. As mentioned above our goal remains the same that is to minimise the error of price difference between [day2 – (day 1+features]. The following parameters were experimented with.

1. Population size- A larger population size means a greater exploration of the solution set. A small population size might mean that the algorithm might converge too quickly and the converse is also true.
2. Cross over probability-Random probability of individuals passing over their genes to the next generation.
3. Mutation-Increases search space and brings new changes to genes i.e. increase or decrease our technical indicators slightly.
4. Generation- The number of generation i.e. the iterations the algorithm would run and try to find the fittest individual. If there are too many generations specified it can lead to overfitting and the converse is also true.

## **Extended Model V2**

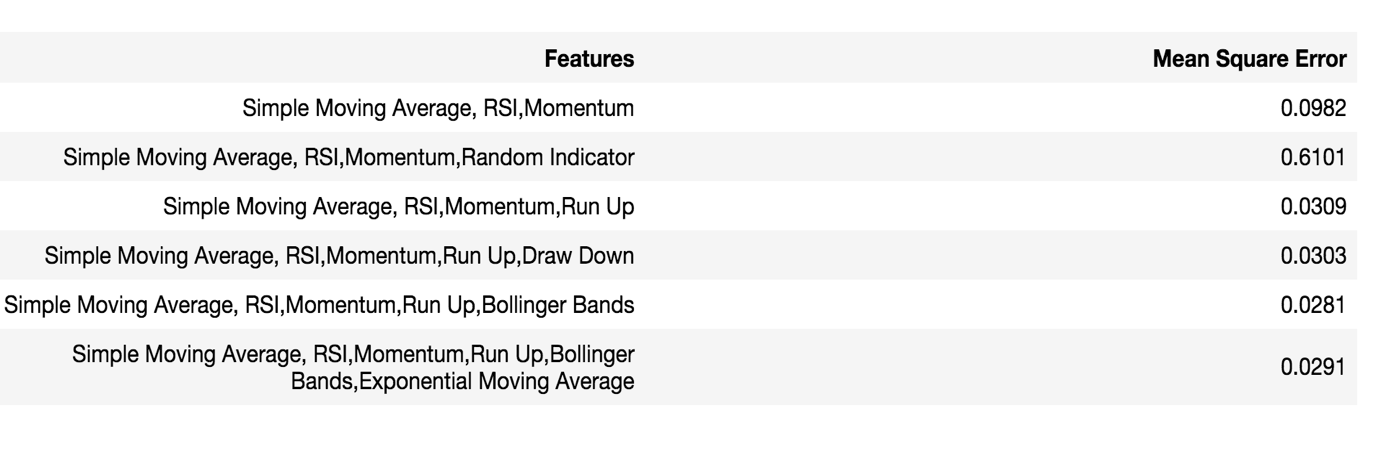
So far the model has three technical indicators and the best set of parameters that gives the lowest error. Now more technical features were added in addition to the three features to see whether the error could be further reduced.

The following technical features were added. For more reference:- <https://www.investopedia.com/>).

1. Exponential Moving Average
2. Draw Down
3. Run Up
4. Random Feature
5. Bollinger Bands

If any of the above features or a combination of the them gave a lower error, this was added to the set of features. If not they were discarded.

The following table shows the errors when additional features were added to the base model.



# **4. RESULTS**

Two stocks the ASX.AX and BHP.AX were taken for the initial base model and the extended models. To ensure accuracy of the results the algorithm was run five times. For testing and demonstration purposes I took the last date as 16th September so I can calculate the price for 17th, 18th and 19th of September which are already available. I wanted to test how the model predicted the prices for data it had not seen before. For the extended model V2 when the best features and the additional features were incorporated both the stocks showed accurate price movements of UP. For the base model only ASX.AX showed accurate results on the first five tries. BHP.AX failed to show accurate results two times out of five. This showed the importance of tuning the model parameters and adding additional features. Overall the error for both the models was less than 0.02.

The following two graphs shows the result of BHP.AX stock and the ASX.AX after running the extend model V2.

A close up of a street

Description automatically generated

Figure 1 BHP STOCK

The picture above shows the results of actual and predicted prices of BHP stock. Although the actual stock market price did go down by a greater margin, the model accurately predicted whether the price would go down.

A screenshot of a cell phone

Description automatically generated

Figure 2 ASX.AX

The picture above shows the results of actual and predicted prices of ASX stock. Although the actual stock market price did go up by a greater margin, the model accurately predicted whether the price would go up.

Five other stocks were also tested. For complete list of csv files please check the GitHub link.

Their results are shown below.

|  |  |  |
| --- | --- | --- |
| STOCK NAME | PREDICTED RESULT | ACTUAL RESULT |
| COH.AX | UP | UP |
| ALU.AX | UP | UP |
| SAR.AX | UP | UP-then went DOWN |
| IAG.AX | DOWN | DOWN-then went UP |
| CBA.AX | UP | DOWN-then went UP |

# **5. Backtesting Strategy**

***This code is implemented in ExtentedModelV2.ipynb***.

Back testing according to Investopedia.com is how well a strategy/model would have performed based on historical data. Usually strategies such as simple moving average can be a good indicator as to when to buy or sell. Moreover it allows traders and investors to simulate results without actually investing any real capital.

I implemented a back-testing strategy using the pyalgotrade model. Pyalgotrade is an event driven python algorithmic trading library. I took the previous year historical prices of ASX.AX. Instead of the actual prices I run my model and put my get the predicted prices over the last year. After computing my forecasted prices I simulate a simple moving average strategy; that is buy when the closing prices are above the simple moving and vice versa. I simulate paper money ($10000) to see how much I would have earned over the period using my forecasted prices.

The following pictures shows the exact time, and the position I should have taken if I wanted my portfolio to increase whilst holding the ASX.AX stock over the last year.

A close up of text on a white background

Description automatically generated

A screenshot of a map

Description automatically generated

Given I use my model to predict I would have made a profit of 71 dollars over the last year on ASX.AX stock using the simple moving average strategy.

# **LIMITATIONS AND CHALLENGES**

## **Limitations**

The model although forecasts quite accurately however I believe it has two main limitations.

1. **Automation**- The model lacks automation. Since yahoo has restricted its API access preventing get requests to download csv files this means CSV files have to manually downloaded, changed into training and current, before it can be used for prediction. Furthermore, if someone wants to get the best parameters first they have to run ExtentedModelV1.ipynb which would give the best parameters. Then they would have to run ExtentedModelV2.ipynb to see which additional features give the lowest mean square error and then finally run the forecast for just one stock!. However I believe this issue is not grave concern and it can be fixed through additional data structures, loops and functions.
2. **Does only linear prediction**:- Because of the way the genetic algorithm in this model is designed. The following illustrates this

t here stands for today.

=>price(t+1)=closingPrice(t)+(best feature)

=> price(t+2)=price(t+1)+(best feature)

=> price(t+3)=price(t+2)+(best feature)

.

.

.

So on

Since the best feature is a constant it will always give output in one direction. For instance the graph for SAR.AX illustrates this issue

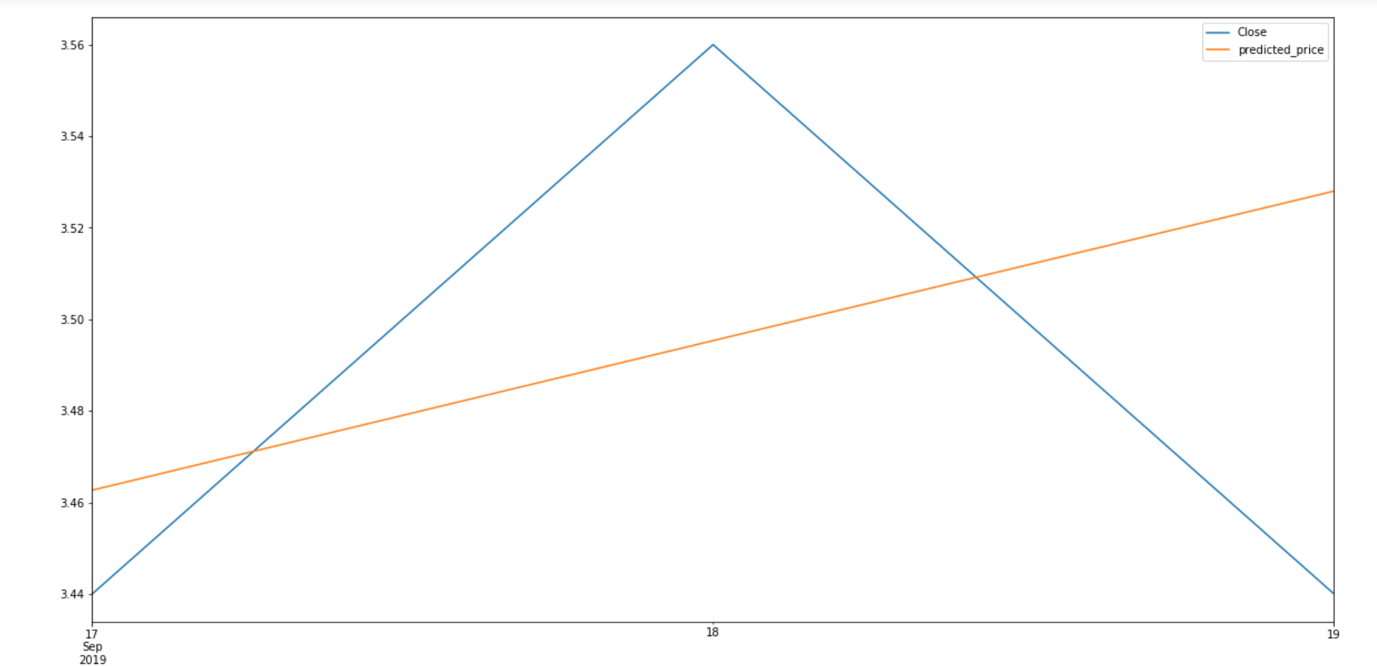


Figure 3 SAR.AX

This model accurately predicted the price the next day, however it will only keep on increasing, instead of decreasing. The quick solution would be to add the new predicted price to the dataset, run the fitness function again, get the value of the new fitness function and calculate the price for the next day. Moreover, I read on Investopedia how recently genetic algorithms are being embedded in artificial neural networks and that is also a direction to definitely look into.

## **Challenges**

The biggest challenge was the amount of support available online there is for the DEAP library and pyalgotrade online. Genetic algorithm by itself is very difficult to understand and implement particularly in the context of stock market modelling. I overcame this by doing going over python libraries on genetic algorithms and reading their documentation and trying and testing on the stock data that I had downloaded from Yahoo. However the following sources helped me understand the basics of genetic algorithms

1. The Applications of Genetic Algorithms in Stock Market Data Mining Optimisation (Lin et al., 2004)

<https://medium.com/datadriveninvestor/an-insight-to-genetic-algorithms-part-i-a7f5a5d6d214>(Chathurangi Shyalika, 2019)

3.DEAP library (F\’elix-Antoine Fortin et al., 2012)

Pyalgotrade despite its amazing capabilities and features is also extremely difficult to understand and implement and would not have been possible without extensive testing and reading of its documentation.

For feature extraction I used the library functions from the QSTK library. However, as mentioned above the library had long been deprecated and no longer suitable for python 3 or 2 imports. I had to go over their documentation on GitHub and modify the data I read from the csv file according to what the QSTK module would expect.

Furthermore, it was through trial and error, a lot of persistence and solo white-board sessions that helped me develop this model.