Practical Project of Time Series Analysis on the Stock Market

Cameron Robson-Leigh  
210764598Dr Emmanouil Benetos  
Big Data Science

*Abstract*—With the constant rise of online investments into areas such as cryptocurrency and the stock exchange, there is an increase and likening into the use of machine learning models to attempt to predict the future of what the value of these assets may hold. However, there has been little evidence to suggest that these have much effect. On top of the confusion around different time series analysis methods and predicting the market, people are also unsure on whether this will actually benefit them. I have created a platform for users to invest fraudulent money after performing a machine learning model of their choice; ARIMA time-series analysis or an LSTM Neural network.

Keywords—ARIMA, LSTM, Neural network, Time series analysis.

# Introduction

With the addition of new ways to invest online, new currencies such as BTC, Digital assets such as NFTs and generally just becoming more aware of the power of investment, I decided to look into ways of creating a platform so that people can understand how machine learning models perform on them. These intelligent methods can potentially work to make people profit. However, there is a lot of negative reviews and the impact they have on people’s financial status. Throughout this report we will be delving into the different models applied within the code, breaking them down and reviewing the results on how they perform.

One model I have implemented is ARIMA which is an acronym that stands for Auto Regressive Integrated Moving Average; which is also a combination of both Auto Regression and Moving average. The model is used to understand historical data and to predict future data in a time series. It can be used when a metric is recorded in regular intervals, from fractions of a second to daily, weekly or monthly periods.

The other model I implemented within the platform is a Long Short-Term Memory network, which are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.

These models were implemented using a variety of packages and research articles sourced from the Internet. They are all centralized onto a local platform that takes advantage of the powers of a web framework named Django, we take advantage of the frameworks authentication methods and databases to store users and their corresponding investments.

The project aims to develop an understanding for users to invest their money cautiously and responsibly when using these machine learning technologies. The models are using datasets from Yahoo Finance where we use the past 5 years of historical data for each asset to feed into the respective models.

# RELATED WORK

Generally speaking, we can split this project up into three different sections. Throughout the paper I will be discussing the optimal parameters, research that have used similar techniques and why I decided to use these techniques.

* Explore the implementation of the ARIMA Time series analysis model.
* Explore the implementation of the LSTM Neural network.
* Explore the web framework and showcase the stages of the site.

# Exploring the ARIMA Time Series Model

The ARIMA model is used to forecast future time series values based on previous and past error term values. The ARIMA model can predict future values of a time series based on the last values of the time series, past values of the error terms, and past values of the differenced time series. [1]

ARIMA models can be broken down in regards to its variables through (p, d, q).

P refers to periods to lag for, if P = 5 then this will mean we use 5 previous periods of our time series in the autoregressive portion of the calculation. P will aid to fit the line in regards to the forecasting.

D stands for the number of differencing transformations required by the time series to get stationary. In an ARIMA model we transform a series into a stationary one using differencing; this refers to a series without trend or seasonality. Stationary time series is when the mean and variance are constant over time. It is easier to predict when the series is stationary. Here we can see just by this definitions the problems with Time series analysis using ARIMA – as financial stock data is extremely volatile.

When implementing the differencing, we need to find the difference between the current time period and the previous time period. If these values fail to revolve around a constant mean and variance then we find the second differencing using the values of the first differencing – this is then repeated until we get a stationary time series.

In regards to how I implemented the ARIMA model in python I took advantage of a particular package named auto ARIMA – this allows us to optimize the ARIMA model to get the best results for our data.

Text

Description automatically generated

Here we can break down our parameters to understand what is going on.

Start\_p – this is the starting value of p, meaning that we start with 1 previous period.

Start-q – the starting value of q, this is the order of the moving-average portion of the model.

Max-p/min-p – the maximum and minimum orders of p, respectively.

M – the period for seasonal differencing, the number of periods in each season, e.g. 4 for quarterly data.

D – The order of the seasonal differencing. If none.

Seasonal – Boolean variable to determine if data is seasonal or not.

Stepwise – a Boolean variable to determine whether to use the stepwise algorithm outline in Hyndman and Khandakar (2008) to identify the optimal parameters. Using this stepwise algorithm can be significantly faster than fitting all hyper-parameter combinations and is less likely to overfit the model.

A screenshot of a computer

Description automatically generated with medium confidence

If we analyse the summary fit with the stock BP for the past 1 year data we can see that the ARIMA model’s best fit is (0,1,0) – a random walk.

A random walk reveals that we have random, uncorrelated, noise, meaning that the data is not allowing us to predict well. This is not relating to just the stock but in general many time series receive random walks as output, referencing the fact that stock markets are so volatile that they are extremely difficult to predict correctly. In fact, surprisingly the stock data model that fit a non-random walk was Netflix, that fit (1,1,1), meaning the model has one AR term, a first difference, and one MA term.

Text

Description automatically generated

Let us dissect Microsoft (1 year data)

A screenshot of a computer

Description automatically generated with medium confidence

According to the model summary, the model meets the condition of independence in the residuals (no correlation) because the p-value of the Ljung-Box test (Prob(Q)) is greater than 0.05.

Forecast on graph:

Chart

Description automatically generated

Text

Description automatically generated

The above screenshot demonstrates when adding more data (5 years of historical) to the model for Microsoft. Let’s break this down further.

A picture containing text

Description automatically generated

The Augmented Dickey Fuller test is a common statistical test used to test whether a given Time series is stationary or not. The results of this test are a negative number, if the negative is strong, we reject the hypothesis that there is a unit root at some level of confidence. [4] What we mean by a unit root is a stochastic trend in a time series, sometimes called a random walk with drift. If a time series has a unit root, it shows a systematic pattern that is unpredictable. Technically speaking, a unit root is said to exist in a time series of the value of alpha = 1. [5]

A picture containing clock

Description automatically generated

Where Yt is the value of the time series at time ‘t’ and Xe is an exogeneous variable –this could be variables such as a pandemic that would affect the series, this is very applicable to our datasets and will provide more reason for higher ADF statistics.

Before we fully understand what the ADF test does, we need to understand the Dickey-Fuller test.

A picture containing text, antenna

Description automatically generated

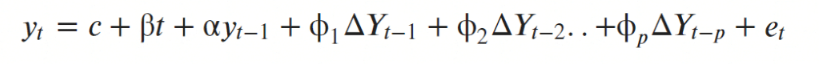
A Dickey-Fuller test is a unit root test that tests the null hypothesis that α=1 in the following model equation. alpha is the coefficient of the first lag on Y.

Text, letter

Description automatically generated

It has a similar null hypothesis as the unit root test – meaning that the coefficient of Y(t-1) is 1, declaring presence of a unit root, if this is not rejected the series is non-stationary.

The ADF tests is an expansion on the former, to include high order regressive process in the model.



The equation above is very similar, the only difference is that we are adding more differencing terms – this enables removing dependence on time to allow more thoroughness on the test.

Our null hypothesis assumes the presence of unit root, since we have a p-value of greater than 0.05, we do not reject the null hypothesis, and the series in fact not stationary. This is a trend throughout all of our stocks.

Chart, histogram

Description automatically generated

Chart, line chart

Description automatically generated

We can see from the plots that the correlogram does not show any correlation in the residuals. [6] The residuals being what is left over after fitting the model, they are equal to the difference between the observations and the corresponding fitted values, denoted as:

A picture containing text, clock

Description automatically generated

Residuals are beneficial in checking whether a model has captured what it needs too. In ARIMA, we can assume that the residuals have following properties:

1. They are uncorrelated, if there are correlations between them, then there is information left in the residuals which should be used in computing forecasts.
2. The residuals have zero mean.

The Standardized residual plot shows us there is a lot of variance in the latter of the plot.

We can see from the QQ-plot that the residuals do not follow a normal distribution – a requirement to validate the model.

The above tests have been on large datasets (1 year and 5 year respectively), however I have only demonstrated 1 day predictions, we have not yet described the difference in long-term and short-term forecasting. I will be forecasting 100 days into the future.

Text

Description automatically generated

Chart, line chart, scatter chart

Description automatically generated

As we can see from the prediction for 100 days, we have a Heteroskedasticity test of 6.98, along with 9.44 for 1 day forecast. This means that the model explains the variance much better and outperforms in terms of dependability. This gives us the conclusion that ARIMA Time series forecasting is much more effective short term than it is long term. I will be discussing the results for variety of stocks further on in this report.

# Exploring the LSTM Neural Network Implementation

Long Short-Term Memory (LSTM) networks have been used for years for a variety of different tasks; such as machine translation, speech recognition and development in the health care sector. However, what a lot of people do not know is that they can prove to be quite effective in time series analysis – this is because they are capable of learning order dependence and sequence prediction problems. [7] The LSTM cell adds long-term memory in an even more performant way because it allows even more parameters to be learned. This makes it the most powerful [Recurrent Neural Network] to do forecasting, especially when you have a longer-term trend in your data. LSTMs are one of the state-of-the-art models for forecasting at the moment. [8]

Throughout the project I built a sequential Neural network using Keras based on bidirectional LSTM layers to capture the patterns in univariate sequences that we will input to the model.

1. *Data Pre-processing*

The data I received from Yahoo Finance API which provides us with a dataframe and we are looking at the previous 1 year’s data with a 80/20 training/test split.

Text

Description automatically generated

Training a neural network within a machine learning model needs to have a format of {<features>, <target>} format. Similarly, we need to convert the given data into this format – let’s talk about the concept of a look back.

1. *Look Back*

A look back is nothing but the number of previous days’ data to use, to predict the value for the next day. For example, let us say look back is 2; so in order to predict the stock price for tomorrow, we need the stock price of today and yesterday.

We can break this down with an example of how the data needs to be in a particular format. If we have the dataset [2,3,4,5,6,7,8,9] and we have (n=3) (n being look back), then the data format would look like:

* [2,3,4] -> [5]
* [3,4,5] -> [4]
* [4,5,4] -> [6]

There is a function in Keras that does this for us.

1. Implementation of the Neural Network

Text

Description automatically generated

For my implementation I kept things quite simple, from a user experience point of view (not waiting long periods for the model to train when navigating the platform).

Chart, diagram, box and whisker chart

Description automatically generated

[10] The picture above depicts four neural network layers in yellow boxes, point wise operators in green circles, input in yellow circles and cell state in blue circles. An LSTM module has a cell state and three gates which provides them with the power to selectively learn, unlearn or retrain information from each of the units. The cell state in LSTM helps the information to flow through the units without being altered by allowing only a few linear interactions. Each unit has an input, output and a forget gate which can add or remove the information to the cell state. The forget gate decides which information from the previous cell state should be forgotten for which it uses a sigmoid function. The input gate controls the information flow to the current cell state using a point-wise multiplication operation of ‘sigmoid’ and ‘tanh’ respectively. Finally, the output gates decides which information should be passed on to the next hidden gate.

These are the default parameters for the Keras package, however we used a ReLU activation function.

1. Activation function

An activation is used to generate or define a particular output for a given node based on the input it is getting provided.[11]. They are critical for any type of neural network, as they provide assistance within learning the complex patterns in the data.

The main 3 activation functions in neural networks are Sigmoid, ReLU and Tanh.

The sigmoid function, also known as the logistic function which helps to normalize the output of any input in the range between 0 to 1, providing data consistency, efficiency and accuracy. It is denoted as y  = 1/(1+e(-x) ).

There are drawbacks to the sigmoid activation function, which is that it creates a vanishing gradient problem.

The vanishing gradient problem is encountered during the backpropagation of the network. This happens during each iteration of training each of the neural network’s weights receives an update proportional to the partial derivative of the error function with respect to the current weight. This becomes an issue as the gradient will become small, and prevents the weight from changing its value. [11]

Tanh activation function is very similar to the sigmoid function, except it has a higher range (-1 to 1 instead of 0 to 1). This superiority over Tanh means it can adapt a greater impression on datasets as negative data is also considered. However, it still faces the same issues of the vanishing gradient problem.

ReLU was the activation function that I went with, it is the most optimal activation function as its range is from 0 to infinity, meaning that the vanishing gradient problem becomes no more.

I attempted all three activation functions within my solution, however ReLU was the only candidate with sensible predictions, indicating that I was suffering with vanishing gradient problems.

1. LSTM Units and Input shape

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. [9]

Diagram

Description automatically generated

Each unit (in my case 10) has an input, output and a forget gate which can add or remove the information to the cell state. The units are essentially neurons, the image below depicts the structure of the network used to forecast our data. [12]

Diagram

Description automatically generated

The input shape is the shape of the input data provided whilst training, the rest of the model however, cannot know the shape of the training data. The shape of the other layers is computed automatically through the Keras package. We used dense layers for this LSTM, which requires input of (batch\_size, input\_size). If we reference what I discussed earlier, the batch\_size is our look back (15 I used) and the input size being 1.

1. Optimizer and loss function

Adam optimizer is a replacement optimization algorithm for stochastic gradient descent for training deep learning models – it updates network weights iteratively based on the training data.

Adam combines the benefits of Adaptive Gradient Algorithm (AdaGrad) and Room Mean Square Propagation (RMSProp). AdaGrad is an algorithm that maintains a per-parameter learning rate that improves performance on problems with sparse gradients. RMSProp also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm oes well on non-stationary problems, this is why Adam is best used for time series.

The below references displays the effectiveness of Adam compared with other optimizers.

Chart, histogram

Description automatically generated

Mean squared error is used in regression settings where your expected and your predicted outcomes are real number values, hence why this provides fit for the purpose of the project. The formula for the loss is the squared difference between the expected value and the predicted value, denoted as: [14]

Text

Description automatically generated

To conclude my LSTM network parameters, I went with 25 epochs – meaning that we pass through the training dataset 25 times. I felt as though this is sufficient for complexity of the datasets used.

# Exploring the Web Application

The Django platform provided extremely well for this project, as its native language is python and allowed for native database storage and user authentication. The improvement here would be to store everything on the cloud, from a financial perspective this was not possible.

Graphical user interface, application

Description automatically generated

As I have provided the code in an alternative location, I will simply outline the areas of the website briefly with its functionality.

The Main page allows for a user to select a stock from the market (only included 10 for the concept of the website).

Once a user has decided the stock they want to make a prediction from, they can then decide if they want to model the data to an ARIMA or a LSTM Neural network.

After deciding the stock they want, they can decide how many days forward they would like to predict, which also has the ability to predict x amount of days. This will then generate a graph displaying the forecast along with printing the actual value onto the web page. Users will also be able to put money down on a stock if they feel as though the prediction will work in their favour.

# Results

For predicting the stock market for the day after the user has placed a transaction, I have created a front end UI to see how their predictions performed for the following day.

A screenshot of a computer

Description automatically generated with low confidence

As we can see from the above, generally speaking the next day prediction is rarely accurate. Although, ARIMA short term forecasting provides much more effective.

# Conclusion

In this work, we developed a centralised system for users to predict the future of stock markets to get a better understanding of the power of these machine learning models. What we learned is that ARIMA stock market prediction is much more effective for next day predictions in comparison with neural networks. We learned as a general idea that machine learning models should not be used to benefit financially and that the market is an extremely volatile place. We witnessed the power of data first hand by accessing Yahoo finance API where we can at least attempt to predict the future through their access to vast datasets.

# Future Work

In regards to improving this platform there are a lot of features that can be added, one that sticks out to me is allowing the user to amend machine learning parameters to their liking so that over time they can get a real understanding of how the machine learning models perform through parameter tuning.

Creating an API that retrieves every stock from the market, and also allow the user to select how far they want to go back in the data.

Although a specification for this project, timing got the best of me, creating a page that scrapes relevant news articles for the stock selected.

I believe with more time and investment this application could be very beneficial for users who appreciate and want to learn more about the power of machine learning models.

##### References

1. Alam, B. (2022). *Using the ARIMA model and Python for Time Series forecasting*. [online] Hands-On-Cloud. Available at: https://hands-on.cloud/using-the-arima-model-and-python-for-time-series-forecasting/#h-understanding-the-terms-used-in-the-arima-model.
2. Sangarshanan (2019). *Time series Forecasting — ARIMA models*. [online] Medium. Available at: https://towardsdatascience.com/time-series-forecasting-arima-models-7f221e9eee06.
3. DOR (n.d.). *Forecasting Time Series with Auto-Arima – Data Science Portfolio*. [online] Available at: https://www.alldatascience.com/time-series/forecasting-time-series-with-auto-arima/.
4. Stephanie (2016). *Unit Root: Simple Definition, Unit Root Tests*. [online] Statistics How To. Available at: https://www.statisticshowto.com/unit-root/.
5. Prabhakaran, S. (2019). *Augmented Dickey Fuller Test (ADF Test) – Must Read Guide*. [online] Machine Learning Plus. Available at: https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/.
6. 3.3 Residual diagnostics | Forecasting: Principles and Practice (2nd ed). (n.d.). [online] *otexts.com*. Available at: https://otexts.com/fpp2/residuals.html.
7. https://www.facebook.com/jason.brownlee.39 (2017). *A Gentle Introduction to Long Short-Term Memory Networks by the Experts*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/>.
8. ‌https://en.wikipedia.org/wiki/Long\_short-term\_memory#:~:text=A%20common%20LSTM%20unit%20is,and%20out%20of%20the%20cell.
9. Goswami, D. (2020). *Comparison of Sigmoid, Tanh and ReLU Activation Functions*. [online] AITUDE. Available at: https://www.aitude.com/comparison-of-sigmoid-tanh-and-relu-activation-functions/#:~:text=ReLu%20is%20the%20best%20and.
10. Wikipedia Contributors (2019). *Vanishing gradient problem*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Vanishing\_gradient\_problem.
11. Thakur, A. (2020). *Keras Layer Input Explanation With Code Samples*. [online] W&B. Available at: https://wandb.ai/ayush-thakur/dl-question-bank/reports/Keras-Layer-Input-Explanation-With-Code-Samples--VmlldzoyMDIzMDU#:~:text=Input%20Shape%20In%20A%20Keras.
12. Jason Brownlee (2017). *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.
13. Seb (n.d.). *An Introduction to Neural Network Loss Functions*. [online] Programmathically. Available at: https://programmathically.com/an-introduction-to-neural-network-loss-functions/.

**MSc Project - Reflective Essay**

|  |  |
| --- | --- |
| **Project Title:** | Practical Project of Time Series Analysis on the Stock Market |
| **Student Name:** | Cameron Robson-Leigh |
| **Student Number:** | 210764598 |
| **Supervisor Name:** | Dr Emmanouil Benetos |
| **Programme of Study:** | Big Data Science |

1. Strengths / Weaknesses
   1. Strengths

* Creating a working web application that has achieved the main scope of the requirements.
* Implementing working databases in relation to working CRUD operations
* Working with a popular web framework such as Django with User Authentication
* Making use of external API’s
* Implementation of ARIMA Time Series forecasting
* Implementation of LSTM Neural network for Time series forecasting
  1. Weaknesses
* No implementation of Web scraping news articles
* Would want better analysis of the performance of the models
* Allow the users to parameter tune themselves

1. Presentation of possibilities for further work

* Allow for much larger datasets, allow for different splits of the data to interpret the results
* Some kind of dashboard where users can see performances based on their parameters
* Web scraping news articles with sentiment analysis techniques applied
* Allow option for using real money
* Get real time stock data as opposed to daily
* Potentially adding more machine learning models for more comparisons

1. Critical Analysis of the relationship between practical and theory work

As fond as I am with the final outcome of the project, the relationship between the practical and theory in terms of time was not enough. Understanding the models in greater depth as well as creating a functional web application applying these techniques was a lot longer than I initially anticipated.

This is why referencing the weaknesses I would create a dashboard where analysis could be much more effective for the user, and myself. The packages provided across the internet are a beautiful thing but I believe in terms of understanding things in a great detail requires much more than just reading and implementing packages, to understand the true power of these models takes years of research and dedication.

Although that is no critique to myself, I am happy with the platform and will definitely be continuing with it after my Masters has concluded.

1. Awareness of Ethical, Social Ethical Issues and Sustainability

Placing transactions on the stock market is essentially gambling, implementing the machine learning models that I have can lead to people thinking they will win the system; however this is not the case as we know. In regards to going forward with the project, especially when real money becomes involved for the user, disclaimers will need to present.

Verification of users will also need to be present as when real money will eventually become involved as there can obviously be many cases of fraudulent that will evolve from this.

The main purpose of the application was to educate users, especially when dealing with younger users there needs to be documentation in place about the negative implications of investing within the stock market and that these models are not a given that you will reap the rewards.

Of course when we hold all this data about users the privacy is a big factor, funds and personal data must be held securely.