7CCS3PRJ Final Year Stock Trading

Final Project Report

Author: Maten Rehimi

Supervisor: Dr Dino Oglic

Student ID: 1638595

April 17, 2020

Abstract

Stocks for any company can be brought and sold for profit if timed correctly. This paper will go through multiple ways of predicting stock prices using data in various markets such as real estate, stocks, bonds, natural resources and cryptocurrencies. This data will be used to train our models which will all be neural networks. The hyperparameters of the neural networks will be changed to find the optimal parameters to produce an ideal model which has a small error rate when compared to the actual value. The predicted stock values for each model will be compared to find the optimal accuracy and hyper parameters. Trends can be identified from the predicted stock values. These trends will be observed to determine whether deep learning approaches can provide an edge on the market by predicting trends allowing one to trade stocks while being more profitable than common methods used to track trends in algorithm trading.

Originality Avowal

Maten Rehimi

April 17, 2020

Acknowledgements

I would like to thank my supervisor Dr Dino Oglic for proposing this project and guiding me throughout the project. He has provided me with great insight about the problem space and showed various applications to further my understanding of this interesting topic.

Contents

Chapter 1

Introduction

This paper will identify strategies to find trends using time-series forecasting which is a technique used to predict future events by analysing the past trends and assuming these trends will continue in future. These trends will be compared to common methods used to track trends in algorithm trading. The key problem in this process is to produce accurate future trends from the time-series forecasting predictions therefore the main problem is predicting accurate stock prices using machine learning.

A large amount of research is done to explore ways of analysing stock price data to identify when to buy and sell to make profit. When analysing data, past data is being used to determine the action to take at that particular time. An example can be to only sell a stock when the 30 day moving average is lower than 200 day moving average. Many other techniques can be used including human judgement to decide on the action after observing a 30 day moving average which could potentially increase profit returns. This process could be extended if the future stock prices could be predicted which is known as a time-series forecast. This could help develop both the prior strategies mentioned as the future trends could alter when to buy and sell. However, the future trends must be accurate otherwise there could be a loss. Despite this, the benefits would be advantageous for both experienced stock traders and unexperienced people. The reason for this is that experienced traders would be able to observe both the past and future trends to determine when to buy and sell. However, unexperienced people could use an autonomous agent to determine when to buy and sell for them using the predicted trends therefore the human factor is reduced which increases the likelihood of profit. The primary task is to obtain a good model using machine learning. In addition, research in machine learning methods can help gain a better understanding of how they work and could potentially benefits other industries which use machine learning. An example of this is the Healthcare industry which uses machine learning methods to assess patients health in real time and flag anomalies to Doctors so they can be treated. In this paper, deep learning will be used to predict stock prices using data from various fields to attempt to find new trends.

Stocks can be predicted using various deep learning methods such as Long Short Term Memory (LSTM), Multilayer Perceptron (MLP) and Convolutional neural networks (CNN). The problem of predicting stocks could be categorised as a supervised learning task where there are two classes for the stock going up and down. Another representation of the problem is to predict the direct value of the stock. In this paper, the value of the stock will be predicted to help show the exact amount of increase or decrease from the actual value. The level of deficit is necessary to track as it helps identify trends which would be unclear if the output was simply increase or decrease. A general problem with modelling is overfitting which occurs when the model predicts well for training data but does not generalise well when introduced to new data. Neural networks have high flexibility as you can have different types of layers and these layers can be modified to reduce the chances of overfitting.

Supervised learning tasks have an input which is known as a feature vector and an output associated with the corresponding input. The model is a function which predicts the output when given a feature vector. It uses input-output pairs to train the model so that it can predict with high accuracy. For example, the input feature vector could contain the stock price for Amazon stocks in the last 10 days. The corresponding output would be the Amazon stock price on the 11th day. This would represent one input-output pair and the input can also be represented as (time step, feature), where in this case the values would be (10,1). In addition to this, there would be many samples of these to help the model train. Therefore, overall we have (sample, time step, feature) as input.

The data used in the feature vector was broken down into three main types: Index funds (section 4.1), cryptocurrencies (section 4.2) and natural resources (section 4.3). The index funds collected are VTSAX (Vanguard Total Stock Market Index Fund Admiral Shares), VTIAX (Vanguard Total International Stock Index Fund Admiral Shares), VBTLX (Vanguard Total Bond Market Index Fund Admiral Shares) and VNQ (Vanguard Real Estate Index Fund ETF Shares). The natural resources included are prices for gold, silver, platinum and palladium. In addition, the cryptocurrencies included are Bitcoin, Ethereum, Ripple, Litecoin, Tether and Bitcoin Cash. The feature vector was recorded for a single day. All these features were collected as they have an effect on the stock market and hence on individual stock values. The data collected covers different industries such as US stocks, international stocks, bonds, real estate, cryptocurrency and natural resources.

The supervised learning task will be modelled using three different types of neural networks which are MLP (section 5.1), LSTM (section 5.2) and CNN (section 5.3). Each neural network has multiple hyper parameters which will be tuned to produce accurate models which are ideally not overfitted. The neural networks will also have other factors changed such as the number of time steps recorded to predict the next day for the stock. For example, you might record the last 10 days to predict the next or the last 20 days.

There were many different ways to implement neural networks but I chose to use Keras and Tensorflow (section 5.4) in python 3 as it has been used in the past with great success. Many features have been added as there have been so many people using it. Keras is built on top of Tensorflow as Tensorflow is harder to learn than Keras which is a high level API. Many large companies have recommended Keras such as Google, Microsoft, Amazon and Apple. This shows Keras has been approved by experts in industry.

Chapter 2

Motivation

The Stock Market is one of the simplest concepts to make money in theory, buy low and sell high. Anyone could buy and sell stocks from any country with just a single phone. People with any profession, such as white and blue collar jobs, could trade as little as 5 minutes per day and gain a passive income while working their normal job. On Wednesday 16th October 2019, 3,233,694,441 US stocks were traded in a single day which signifies how common trading is [11]. The stocks could be held for months without checking and still potentially generate a profit when sold which has an extra benefit as money is susceptible to inflation. However, blindly trading random stocks is merely gambling and precautions need to be taken to reduce the risk of the investment.

Generally, long-term stocks that are sold for profit require less tax than short-term stocks which makes it less risky. Long-term stocks are stocks which have been kept for more than one year and short-term stocks have been brought and sold in less than a year. A portfolio with stocks from many companies and many countries reduces the risk of loss as it is extremely unlikely that all the markets in each country crashes simultaneously. In addition, if one stock drops significantly it does not effect you as much because you have so many other stocks. An example would be the S&P 500 (Standard and Poor) which is an index fund for the top 500 US companies.

Index funds are created by top investors who have a vast experience in finance and the economy and it can potentially contain stocks, bonds and real estate in the same or different markets. Warren Buffet, considered by many to be the most successful investor advised "My regular recommendation has been a low-cost S&P 500 index fund" [10]. An example of an international bonds only index fund is VTIAX (Vanguard Total International Stock Index Fund

Admiral Shares). A common technique practised is to have 3 index funds in your portfolio: US stock index, international stock index and a bond index. These index funds all have different markets so it is less risky if one of the indexes drops significantly like the VTIAX between 2015 and 2016 (as shown in figure 2.1 below).

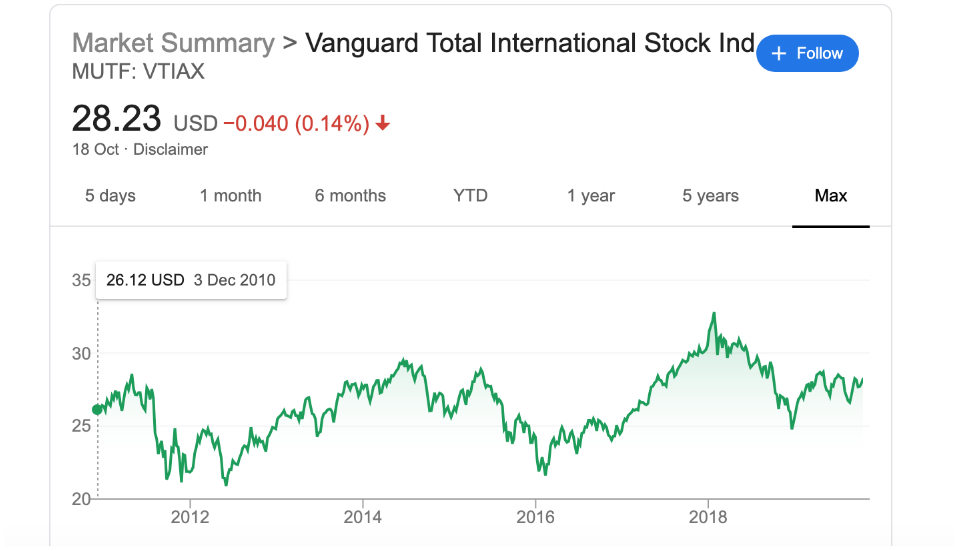


Figure 2.1: Value of VTIAX
[4]

Although Index funds are a less risky alternative than pure stocks, they have their own risks associated with them as the prices are volatile. The prices of stocks are dependent on many features such as other stock prices and bond prices. In addition, the features could be politically related such as a speech about Brexit could effect stock prices and potentially currencies too. Some of these links between features are obtainable by the human perspective however some more complex links can only be obtained by a machine. Neural Networks are a machine learning technique used to find patterns between correlated variables. They have been effectively used in image processing, medical diagnosis, speech and handwriting recognition. If the neural networks in this paper are successful and have a high accuracy rate, there will be a large economic value in such a model. In addition, the model could be observed and analysed to see what factors were used in high accuracy predictions of stocks. Therefore extended work could be done to include more factors to potentially increase the accuracy rate further to generate larger profits from the stock markets.

Chapter 3

Background Research

To produce accurate trends in the data, we first need to have accurate stock predictions. There are several ways to predict stocks however one of the most popular methods is using neural networks. Neural networks have been successful in many fields such as Character Recognition, Image Compression, Travelling Salesman problems and most importantly Stock Market predictions. This versatility in producing good predictions is due to the flexibility of creating different types of layers in the neural network. There are many different types of neural networks that can be applied to predicting stocks such as MLP, CNN and LSTM.

3.1 MLP

A study was produced to predict stock prices using a Multi-Layer Perceptron by Barack Wamkaya Wanjawa [3]. The data used to predict contained the closing stock price of around 60 companies traded at the NSE (Nairobi Stock Exchange). Not all the stocks could be predicted as there were certain conditions that were required to be considered, such as constantly being traded. After the considerations, only 6 stocks remained which were Kakuzi, Standard Bank, Kenya Airways, Bamburi, Kengen and BAT. After using the MLP model, the MAPE (mean absolute percentage error) was measured for each stock. The average MAPE was 1.583% [3]. The figure below shows an indication of how the MLP model captures trends before they appear. Especially between 20-Jan-12 and 24-Jan-12 where the prediction of the stock is greater than the actual value increase on the 25-Jan-12. This could show that the algorithm predicted the price would increase again which proved to be correct.

The paper used 60 traded companies to predict the stock price however in this paper, data

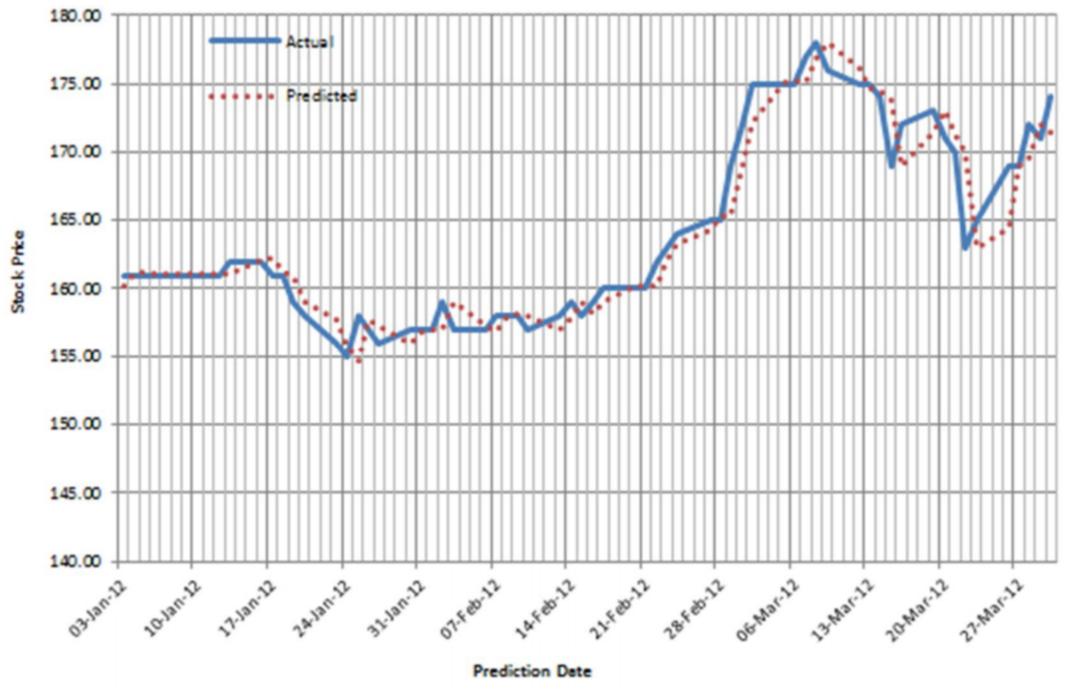
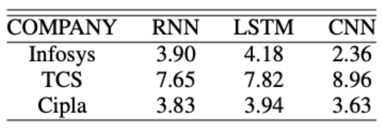


Figure 3.1: A prediction of Standard Bank stock using MLP
[3]

from various markets such as bonds, stocks, natural resources and cryptocurrencies will be collected to potentially observe new trends that were not captured in this MLP model. The advantage of a MLP over CNN and LSTM is that they provide non-linearity and are simple building blocks therefore the they are easier and hence quicker to train.

3.2 CNN and LSTM

In 2017, a paper was publish exploring different ways to predict stocks using RNN, CNN and LSTM [9]. The stocks being predicted were stocks from three companies: Infosys, Cipla and TCS (Tata Consultancy Services). The figure below shows the error when using the three models mentioned.



COMPANY	RNN	LSTM	CNN
Infosys	3.90	4.18	2.36
TCS	7.65	7.82	8.96
Cipla	3.83	3.94	3.63

Figure 3.2: A prediction of company stocks using different models [9]

As seen from above, the CNN performed best in both Infosys and Cipla. This could be due to the fact that convolutions are good at capturing patterns that differentiate between time-series. However, LSTMs are recurrent neural networks which capture the sequential information between time steps. In this scenario, LSTMs performed the worst for two of the three companies which could be due to the fact that the data collected was too small so the benefit of long-term relationships was nullified. Specifically, Cipla had 400 to 700 records which would explain why the accuracy was the lowest of the three. This is understandable however as LSTMs generally take longer to train due to trying to capture longer term dependencies.

Chapter 4

Data Collection

The data was collected from Perthmint and the Yahoo Finance API, and stored in csv (comma separated values) files. Each csv file had a column labelled Date which contained the day, month and year for each row. The csv files were combined using the same date and ensuring no other columns were empty. If the other columns were empty, some features would not be included in the input of the model which would have effected the output. In addition, our model could have been trained on partially complete data which could potentially effect all subsequent predictions negatively. The fully complete feature vector contains the date and data for different markets such as the American stock market, international stock market, bond market, real estate market and finally, the cryptocurrency market. All of these features have an effect on the stock market. Therefore, it must be considered when producing the output of the model. The American stock market, international stock market, bond market and real estate market has data in the form of index funds.

4.1 Index Funds

An index fund is a calculated value that summaries the performance of assets in a market. For example, we have the S&P 500 which represents the value of the top 500 largest capitalisation weighted US companies. The S&P 500 is a capitalisation weighted index which means larger companies effect the index fund more greatly than smaller companies. If there were two companies in an index fund which were worth £100 million and £200 million, then a 5% increase in the £200 million company would increase the index fund twice as much as a 5% increase in the £100 million company. In addition, the top 500 largest companies may change so the stocks are sold and brought to bring it as close to the S&P 500 index as possible. The figure below shows the valuation of each company to the S&P 500 index.

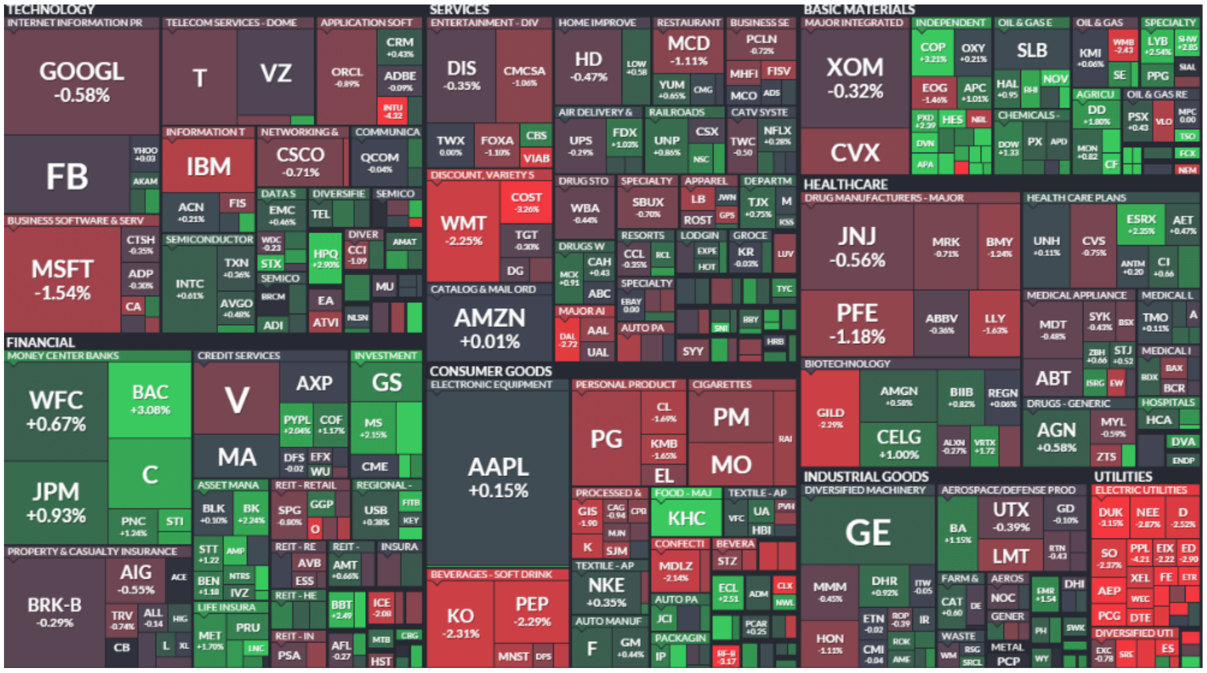


Figure 4.1: S&P 500 Visualization of Companies by Market Capitalisation [2]

4.1.1 Vanguard Total Stock Market Index Fund Admiral Shares (VTSAX)

The VTSAX is an index fund which contains assets in the US stock market. It covers all the top companies such as Apple, Microsoft, Amazon, Facebook and Johnson&Johnson. This index fund is important to consider when trying to predict a single stock because it represents the whole US stock market which is the most lucrative and contains the largest stock exchange, called the New York Stock Exchange (NYSE). If VTSAX has been increasing in price, it is likely that your stock will go up if it is a large company as it has a larger effect on the index fund. There is no guarantee that it will go up as there is always an element of risk when predicting stocks. Generally, people invest in multiple index funds in different markets to diversify their portfolio because when a market goes down in price and lowers one index fund, there is a chance that another market you have invested has gained the loss from the first market.

4.1.2 Vanguard Total International Stock Index Fund Admiral Shares (VTIAX)

VTIAX is another index fund which is similar to VTSAX but contains assets in the international stock market. When predicting a stock, stocks of other countries could have an effect because a rival stock could be represented by another country. For example, VTIAX contains the Toyota Motor corporation stocks therefore if you were predicting another car company stock like Vauxhall stocks then you must consider other car companies. There could be many different relationships between the two stocks. Toyota could have a new model which attracts potential Vauxhall car owners to switch car company then there would be an inverse relationship between the two stocks. These relationships should be considered as it will be beneficial to the model to make these links hence producing a more accurate output.

4.1.3 Vanguard Total Bond Market Index Fund Admiral Shares (VBTLX)

The Vanguard group has another index fund called VBTLX which represents the bond market in America. An example of the assets it contains is US treasury stocks, agency-backed mortgages and corporate bonds [5]. Bonds are a type of debt usually given out by banks, governments or corporations. For example, a bank could sell 3 year bonds to people. If a person buys a 3 year bond then they must payback the bank by giving a certain amount of money over the duration of the bond which is 3 years in this case. It is similar to a reverse loan. For a currency to have value, people must think the currency has value which is an extremely important concept of currency. People who lose faith in one type of currency generally exchange it for another type of currency which they deem as more valuable. For example, if the economy is declining, people will start investing more into bonds and less into stocks which will increase the bond values and decrease the stock values due to the change in demand. These relationships are important to track therefore bond prices are necessary features that should be part of our feature vector.

4.1.4 Vanguard Real Estate Index Fund ETF Shares (VNQ)

The VNQ index fund summarises the real estate industry which can have an effect on the stock prices predicted. An example of this can be seen in the 2008 recession. The Lehmen Brother investment bank were lending out money to people without checking the background information of the clients. This lead to the housing market to increase in price as more people were taking out mortgages causing the market to bubble. Lehmen brothers brought lots of housing assets as the retail estate industry was becoming more lucrative and many people took out higher-risk loans who could not pay them back because the price of housing kept increasing. The banks had a lot of debt which they could not pay off and generally when this happens, banks sell off their debt to other banks temporarily who have more money. Once the Lehmen Brother bank had enough money, they would buy their debt back with interest from the bank it just sold it to. However to do this, they were required to produce more money quickly so they kept giving out loans to people who could not pay them back, mainly in mortgages. This lead to a cycle where the Lehmen Brothers bank would take mortgages and collect debt they sold off to other banks for buyback interest, just to get more mortgages to get into more debt. This spiralled out of control which led to the bank being in under more than $639 billion in debt to other banks [6].

The Lehmen brother's stopped giving out high-risk mortgages, causing the bubble to pop in the retail estate industry and the housing prices started dropping. When people could not pay the mortgages already given out, the bank would take the house and this common occurrence lead to house prices dropping drastically. Rich people who had houses decided to sell them because they feared that their house price would drop and poor people were losing their houses to the bank as they could not pay off their mortgages. There was an oversupply of houses so the house prices dropped and hence the index fund for house prices also decreased. The bank that collapsed during the 2008 recession was Lehmen Brothers which was the fourth largest US investment bank during the time of its collapse with over 25,000 employees worldwide [6]. People lost faith in the system when house prices were dropping and as a result there was a large supply of houses but many rented until the house price stopped dropping. This is one factor which lead to the recession among others but was a large contributor. Once a recession was triggered, the economy started to do even more poorly and as a result the prices of stocks dropped. From this example, it shows that the retail estate industry can effect the stock market.

4.2 Cryptocurrencies

Cryptocurrencies are a form of digital currency which can not be duplicated. The only way to obtain it is by mining it or receiving it from someone else who has previously mined it. In my feature vector, it contains the price for Bitcoin, Ethereum, Ripple, Litecoin, Tether and Bitcoin Cash. Cryptocurrencies are decentralised from the government because they can not produce more and it can not be removed. This makes it appealing for consumers to buy. Any popular forms of currency has an effect on stock prices because if the cryptocurrency price goes up then money from another market could be reduced, potentially a stock from a stock market. Therefore, popular cryptocurrencies prices should be observed when predicting the price of our stock.

4.3 Natural Resources

In our feature vector, we include prices of gold, silver, platinum and palladium. Natural Resources have been known to be a safe investment for most people because they have real world value. For example, gold can be used to make jewellery therefore it will always have value as oppose to currency such as pounds and dollars. People invest in natural resources because its price does not fluctuate greatly. A common approach is to transfer wealth from pounds to a natural resource when you feel that the value will drop then you can sell the natural resources back to pounds to make a profit. Similarly, the same approach can be taken in stocks to gain a profit and inflation is avoided as natural resources are being invested in. An increase in natural resource price could indicate that people value it over stocks. These relationships need to be considered so natural resource prices will be in our feature vector to help the model extract these complex relationships.

4.4 Implementation

Firstly, I collected the companies in the S&P 500 from a list in wikipedia which is updated regularly. The tickers are collected from the table and stored in a list which is used to obtain stock data for the 505 stocks from the top 500 companies in America with the highest market capitalisation. The stock market is open from 9:30AM to 4PM. For each stock, the features obtained were high, low, open, close, volume and adjusted close for every weekday using the Yahoo Finance API. The high and low features represent the highest and lowest sold stock during the day. Open and close features represent the first brought value of the stock when the market first opens and the close feature represents the last brought value before the stock market closes. The volume is the number of stocks sold in the day and the adjusted value is the value after corporate actions take place such as stock splitting. Stock splitting is used by companies to reduce the value of their stock to affordable amounts by increasing the volume while decreasing the price. For each stock, a csv (comma separated values) file was created to store the data. An inner join was carried out for all the csv files representing the S&P 500 using the Date attribute therefore a new csv file was made to store all S&P 500 stock data for every day. The stocks observed are introduced in the market at different times therefore the join had to include only dates where the columns were not empty. This lead to missing lots of days at the start of the data due to stocks which were created recently. In addition, some stocks had missing values in between dates therefore the rows were dropped when the join occurred. The stocks are only recorded if there is a sufficient number of values between set years.

Secondly, I obtained the index fund for stocks, international stocks, bonds and real estate. The prices of cryptocurrencies and natural resources were also obtained. The index fund data was collected from the Yahoo Finance API and I collected all the data for every year to try to maximise the samples used to train the model. In addition, the prices of cryptocurrency was collected using the same API and the cryptocurrencies that were obtained were the top 6 most popular: Bitcoin, Ethereum, Ripple, Litecoin, Tether and Bitcoin Cash. The most popular cryptocurrencies were collected as they have the biggest influence on the cryptocurrency market and in turn are a good way to show the relationship between the cryptocurrency market and the stock market. The features collected for natural resources were Gold AM, Gold PM, Silver, Platinum AM, Platinum PM, Palladium AM and Palladium PM. The AM prices were obtained at 11AM and the PM prices were produced at 3PM. The data came from London Gold Fix which was set by the 5 leading gold trading banks. Representatives from these market makers would meet twice a day at 10:30AM and 3PM to agree a fix based on prices which they sold and brought at. In 2014, a trader had been shown manipulating the gold fix prices. The system was then changed to prevent further manipulation of the prices. From March 2015, the price is fixed by a greater number of traders from more companies. Some additions are Goldman Sachs, HSBC banks, JPMorgan Chase, Morgan Stanley and UBS among several others. I have used the same join to merge my data as explained previously using the Date to join features. All my data comes from the price after the system changed in 2015 therefore it has a better representation of the market as it has decreased the likelihood of price fixing.

Finally, there were some problematic cases when producing the code. For example, the tickers scraped from the table from wikipedia had to have the full stop replaced with a dash so the appropriate company could be found from the Yahoo Finance API. In addition, the names of the individual stocks had to be changed so the merge of the join for dates would not be conflicted. Therefore I added the individual stock name then the feature so I could determine the features for each stock when the data was combined. The volume for some stocks were also not shown as it was continuously 0 when the stock had been traded when observing different websites. So I removed the volume column when combining the data in the join. The final feature vector is the index funds, cryptocurrencies and natural resources and the target is a stock price from the S&P 500 collected.

Chapter 5

Model

Neural Networks are a supervised learning or unsupervised learning technique inspired by cells in our brain. For example, when we read the letter 'a', naturally we associate the drawing to the letter. The letter we observe will not always be drawn the same however we can still distinguish what it represents. An example of the use of Neural Networks is classification of writing. The Neural Network would have an input of writing and an output of the predicted word. Neural Networks generally have many layers and an example of a layer could be 26 nodes. Each node from the 26 would contain information about a feature so for the example of classifying letters, each node could represent a letter in the alphabet.

These models can have a different number of inputs and outputs therefore they are widely used in many different areas such as image and object recognition, voice recognition, video game performance, paraphrase detection and spell checking. For example, Neural Networks can be one to one, one to many, many to one or many to many. In this paper, I will carry out a multivariate time series forecasting to predict stock values. This involves using many features to predict a single stock price. I chose multivariate time series forecasting over univariate time series forecasting as many different aspects effect stock prices. Also, if I used one feature to predict the stock price, there could be a correlation between the two but no causation. For example, if I had average milk price and was predicting divorce rates, for a certain period of time they could have both increased. This may have lead me to erroneously believe that an increase in milk prices causes divorce rates to increase. If many variables are used to predict a single feature, it decreases the likelihood that there is an erroneous causation when analysing the results. There are many different types of Neural Networks such as MLP, LSTM and CNN.

There are two main types of ways a Neural Network learns which makes it either a Recurrent Neural Networks or a Feedforward Neural Networks. Feedforward Neural Networks have no cycles or loops in the architecture and move in one direction whereas Recurrent Neural Networks do contain at least one loop or cycle. Each node in a Recurrent Neural Network has the output of the previous time-step as input for the next time step. This shows that Recurrent Neural Networks can make links between different time-steps which the Feedforward Neural Network cannot do. Therefore, this makes Recurrent Neural Networks ideal for learning temporal information such as predicting stock prices every day. Some problems are not immediately apparent to be temporal information such as natural language translation. Recurrent Neural Networks take in the words to be translated as input and outputs the translated word in the desired language. By using Recurrent Neural Networks, the context of the translation can be compared to the past contexts of the translation to help produce a better model. To summarise, Recurrent Neural Networks work well with predicting what happens in the future whereas feed forward Neural Networks are better at predicting what something is. Examples of a Feedforward Neural Network is MLP (Multilayer Perceptron), CNN (convolutional neural network) and an example of a Recurrent Neural Network is LSTM (Long Short Term Memory) Neural Networks.

5.1 MLP

Perceptrons are a single node model which has inputs that are each weighted individually. The weights are randomly initialised at the start of the algorithm. Once the weights have been applied, the node adds up all the weighted inputs and compares it to a threshold. If the value is below the threshold then a value of 0 is produced otherwise 1 is the output. The weights from a Perceptron learn through a method known as gradient descent. An update rule is used to produce new weights if the input is misclassified. The update rule is w=$p(y_{target} - y_{output})x$ where y_{target} is the actual output value and y_{output} is the predicted value. The p is the learning rate which is used to specify how far to move from the previous value. The main purpose of gradient descent is to minimise the cost function by moving the weights. Ideally, you want to move the weights to produce the lowest value for the cost function which is known as the global optimum. If the learning rate p is too large, then you could potentially skip a global optimum which will produce less accurate predictions. Conversely, if you use a learning rate which is too small then you reach your global or local optimum very slowly which requires more computational power and time. It is difficult to produce the learning rate as each problem has a different optimum learning rate. Generally, the main approach is to have a large learning rate and to decrease it slowly as you progress through the algorithm to ensure you get closer to the optimum and do not bounce to either side of it. As shown in figure 5.1 below, there is a minimisation problem which highlights the global and local optimum solutions. It is difficult to obtain the global solution in some scenarios as the data might plateau therefore your algorithm is not making progress on finding an optimum as it is moving randomly.

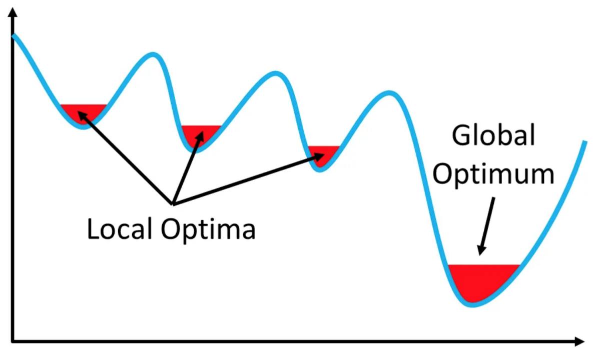


Figure 5.1: Local and Global Optimum
[8]

The problem with a Perceptron model is that it can only model linear problems. An example of this can be shown when trying to predict the output of XOR. Let us have a function XOR with two inputs x1 and x2. Both x1 and x2 can either be 1 or 0. The XOR function returns 1 if the two inputs are different and returns 0 if the two inputs are the same. We want to predict the output of the function given the two inputs. Now let us map the solutions to all inputs onto a graph to determine if a single Perceptron can solve the problem. Since the solution to the problem is non-linear, we can see from the graph in figure 5.2 below that a Perceptron can not create a straight line which splits the two classes.

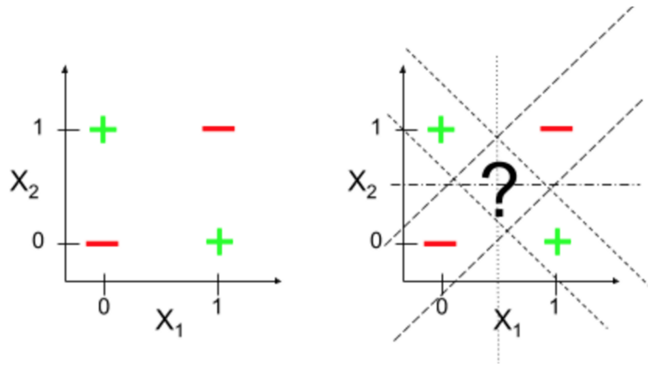


Figure 5.2: The XOR problem
[8]

The Perceptron was proposed during the period of 1957 to 1962 by Psychologist Frank

Rosenblatt [7]. Computing problems using perceptrons would not be possible due to a lack of computational resources as there were no GPUs during those times plus the vanishing gradient and exploding gradient problems were common. The vanishing gradient problem causes the weights to not be updated whereas the exploding gradient would cause the weight to be updated to positive or negative infinity. Minsky and Papert paper in 1959 showed that the Perceptron was unable to solve non-linearly separated problems specifically stating XOR [7]. To solve the problem of non-linearity, the feature vector could be augmented to ensure that the data is linearly separable however it comes at the cost of increasing the dimensionality by 1. The reason why having a high dimensionality of feature vector is that it increases the dimensionality of the feature space. If the dimensions of a feature space is too high, it can cause a well-known problem called "curse of dimensionality". The volume of the space increases so fast that the data points become sparse. Therefore, to obtain an accurate model, the number of data points will need to be increased exponentially. A way of preventing the increase of dimensions by 1 is to use MLP which support non-linear hypothesis like the XOR problem. MLPs started to gain popularity during the 1980s due to its work in speech and image recognition.

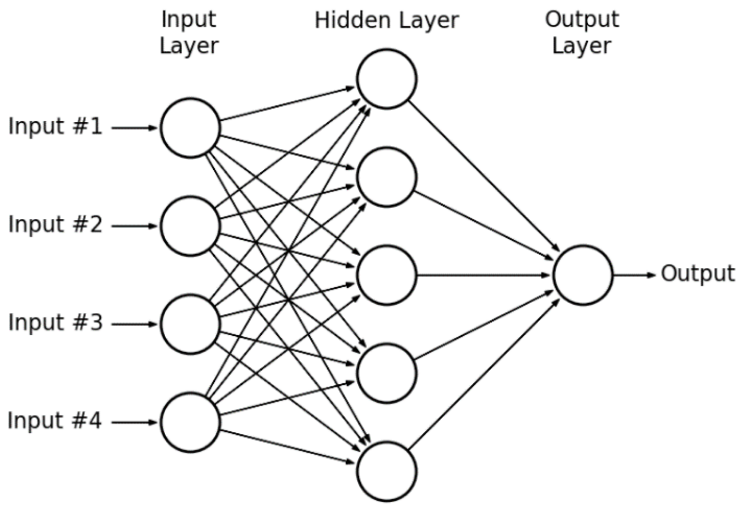


Figure 5.3: Multilayer Perceptron
[12]

As shown above in figure 5.3, MLP have three layers the input layer, hidden layer and an output layer. Some layers can have a bias which is another node that generally has a value of 1. Bias nodes can be used for many reasons. For example, you may have an input of 0 so you might add a bias to prevent it. If the input is 0 then it removes a piece of information which could be used to help you predict an outcome. The situation could become even worse if all the inputs to a node is 0 therefore the node becomes useless and would require a bias. In addition, the bias could also cause inputs to become 0 so you must be wary when to use the bias as it is difficult to determine whether to use one or not. The best scenario would be to test both cases and determine how it effects your error rates. The number of layers in the hidden layer can also be changed. There is no set way to determine the optimal number of hidden layers and the number of nodes in each hidden layer. However you can use intuition to help determine the parameters. For example, if you have 10 features in your feature vector you may consider to have 10 nodes in each hidden layer to represent each feature. You could also do a grid search of the parameters to help find a good combination to produce accurate results. This method can be vary time consuming for a large number of parameters so it should be done with two parameters at a time to help you understand the data you are trying to predict.

5.2 LSTM

A common problem with Neural Networks is the vanishing gradient problem. Vanishing gradient occurs during gradient-based learning methods such as backpropogation when the weights are updated during training of the model. The weights are updated using a cost function which is generally computed using the difference between the networks predicted value and the actual value. Then the cost is lowered by making adjustments to weights and biases during training. This adjustment is done through using the gradient which is the rate at which the cost changes with respect to weight. To model complex problems, Neural Networks with a large amount of layers are used to detect more complex features from the detection of simple features from the initial layers. Due to the large amount of layers, the problem of vanishing gradient was amplified which reduced the use of deep Neural Networks until 2006. The opposite of this problem can also occur which is known as the exploding gradient problem. In this problem, the gradient becomes too large where it dominates the weight in the new weight update function. Therefore, the future weight values get larger and larger resulting in an overflow and creating an incomplete model.

We want to maximise the gradient, which is the rate of cost with respect to weight, to lower the cost of the error so the predicted value is closer to the actual value. When the gradient is large the model learns quickly and if the gradient is small then the model learns slowly. During backpropagation, the gradient is calculated from right (output) to left (input) to update the weights. The way gradients are calculated is by multiplying the gradients at prior levels therefore the weights on the left use more gradients then the weights on the right. As a result the learning rate of the initial layers is very slow and it takes a long amount of time to train if there is a vanishing gradient. This has a large effect on the rest of the model as the initial layers will not be able to detect basic features which are the building blocks to help detect more complex features in later layers. Therefore, the output of the model will have a low accuracy due to not detecting many of these features caused by the vanishing gradient problem.

The main problem is that the gradient is below 1 for the last layers during backpropagation. If this can be prevented then the multiplication of gradients would not result in a lower number. The gradient is proportional to the chain multiplication of the derivative of the activation function in the network before a certain layer when moving backwards through the network during backpropagation. This shows that the activation functions derivative has an effect on the gradient and hence must be picked accordingly to prevent the vanishing gradient problem. The activation functions used prior to 2006 was the sigmoid activation function and the hyperbolic tangent activation function. The derivative of the sigmoid activation function has a range of 0 to 0.25. Similarly, the derivative of the hyperbolic tangent activation function has a range of 0 to 1. Due to both derivatives having values less than 1, this makes it almost certain that the gradient descent problem occurs for deep Neural Networks.

In 2006, the relu activation function was used which has a derivative value of 0 or 1 therefore it reduces the chances of the vanishing gradient problem from occurring in deep networks. In this paper, the relu activation function will be primarily used as the Neural Networks will be very deep so they will be more susceptible to the vanishing gradient problem if sigmoid and tanh activation functions are used. The Neural Network needs to have a lot of layers to extract complex features when predicting stocks as it is a difficult problem to solve. Despite this, an experiment could be done to evaluate the accuracy of the model when the activation functions used is tanh or sigmoid to observe the effect.

The vanishing gradient problem is common in Recurrent Neural Networks however LSTMs are much less likely to run into this problem. LSTMs have a forget gate which have a connection to the gradients computation therefore the forget gate can modify the model to prevent the vanishing gradient problem. The forget gate does not always prevent the problem but it makes it much less likely to take place. In addition, the exploding gradient can be solved by gradient clipping which is the process where the gradient is removed if it is above a threshold. Another solution to the exploding gradient problem is gradient scaling which is similar to gradient clipping except the gradient is scaled down.

Normalisation layers can be used to reduce the likelihood of experiencing the vanishing and exploding gradient problems. Normalisation produces data which has a mean of zero and a standard deviation of one so it generally occurs before activation functions. Therefore, the input to the activation function will have the form ux+b where u is the layer's input, w is the weight and b is the bias. Now as the layer's input is normalised, the magnitude of x will be less than or equal to 1 so the magnitude of f'(x) will be larger where f'(x) is the derivative of the activation function. The derived activation functions such as sigmoid and tanh have their largest value at x=0 and they get smaller as the input increases as shown below. This causes the derivative of the activation function's value with normalised input to be greater so the gradient will be greater decreasing the chance of the vanishing gradient problem. However, with relu activation functions and normalisation it prevents the risk of vanishing gradient and exploding gradient completely. If the data is not normalised before being inputted to the relu activation function then it is possible for the output to be 0 for scenarios therefore the vanishing gradient problem could potentially still occur if there are a large amount of relu layers. However, with normalisation it can ensure most or all values are above 0 to prevent this scenario from arising. In addition, normalisation decreases internal covariant shift as the distribution is maintained.

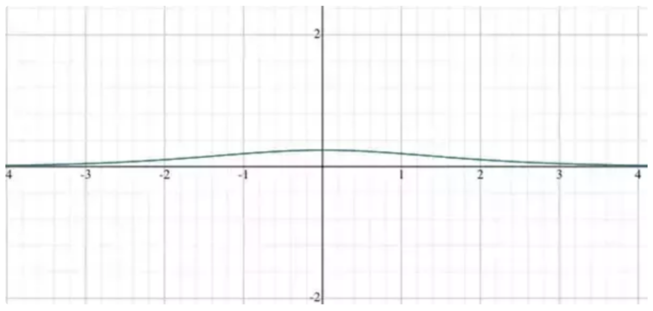


Figure 5.4: The sigmoid derivative

Another strategy to update the weights aside from using gradient descent is to use stochastic gradient descent. Stochastic gradient descent was fundamental for the development of neural networks due to its scalability. Gradient descent updates the parameters after going through every sample whereas stochastic gradient descent updates the parameters after going through only one or a subset sample from the training set. Therefore, stochastic gradient descent converges much faster and keeps oscillating there to create a fixed radius. Decreasing the learning rate, reduces the size of the fixed radius.

One of the benefits of LSTM Neural Networks is that it can connect previous information to the current task. Recurrent Neural Networks can also connect previous information to the current task however the gap can not be too long. As the gap grows, the Recurrent Neural

Network will not be able to connect the two tasks however LSTM Neural Networks do not suffer this gap problem. LSTM can add and remove information to the cell state by using gates. Each cell has three gates: a forget gate, input gate and an output gate. The forget gate is applied first then the input gate and finally the output gate . LSTMs have a cell state which is used to record features for long periods of time to help make a prediction using past data from a long time ago. Firstly, input (x_t) and the previous hidden layer (h_{t-1}) are concatenated and put through a sigmoid function which has a range of 0 to 1. The concatenated output of sigmoid is point-wise multiplied by the previous cell state (c_{t-1}). The range value of 0 to 1 is important as values multiplied by 0 will be completely forgotten and being multiplied by 1 will conserve the value. In addition, any grey area can be produced from a value between 0 and 1.

Secondly, we have the input gate which changes the values of the cell state. It uses another sigmoid function on the concatenated input and hidden layer to decide which values to change. Then it puts the concatenated input and hidden layer through a tanh function which outputs a value between -1 and 1. Sigmoid was not used as the values would only increase therefore it was important for the range of the new function to include negative values. Then the two values outputted by the functions are point-wise multiplied to produce a new value which is added to the cell state (c_{t-1}) after the forget gate was applied. Now the cell state contains values which have been forgotten and updated, the final step is to decide what to output as the new hidden layer.

Lastly, the output layer is applied to the cell state. The forgotten and updated cell state is put through a tanh function and point-wise multiplied by the concatenated hidden layer and the input which is put through a sigmoid function to produce the hidden state (h_t). This same process is applied to the remaining cells of the LSTM.

The LSTM method of 3 gates is the standard model however there have been some little modifications over the years. For example, there is the peephole idea which gives some picked gates the ability to see the cell state before performing a function on it. This process is shown graphically below.

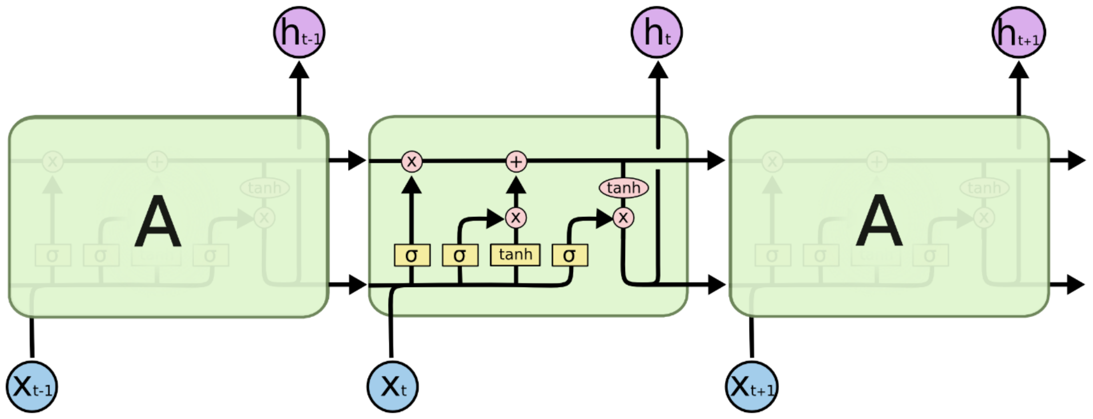


Figure 5.5: LSTM
[8]

5.3 CNN

Convolutional Neural Networks are a type of feedforward network therefore the model consists of no loops or cycles. Convolution can be done in many dimensions such as 1D and 2D. In 2D convolution, there is a filter which goes through the matrix and produces a new smaller matrix as long as the filter is larger than 1 by 1 and multiple filters can be used. The new matrix is produced by multiplying the window with a certain segment in the matrix then it moves on to the next position to generate the new values. A use of 2D convolution can be image recognition. For example, let us get a picture and classify it as either a dog or a cat. The input to the model has dimensions of 32x32x3 where the 3 is the RGB values of each pixel of the screen. Now we apply the convolution by sliding a filter of size 5x5x3 over the input. It starts from the top left and goes right then loops back to the left and goes down by 1 to repeat the process. During every step, the filter is multiplied by the initial image and all these multiplications are added up so the new matrix will be smaller. After all the sliding of the filter, we get a 28x28x1 matrix which is known as a feature map. There are 28x28=784 unique locations that a 5x5x3 filter can be put on a 32x32x3 image. If p filters was used it would create a variable of dimension 28x28xp. The purpose of using multiple filters is to extract different features from the original image. For example, one filter could extract lines or colours of regions of the image. The more convolutional layers you have, the more complex features can be extracted. After 5 convolutions you may have a filter that extract dog ears which is only possible due to the extraction of simple features from the original layers. An example of 2D convolution is shown below.

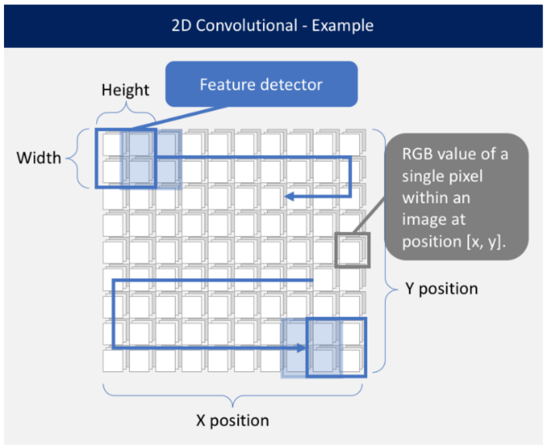

2D Convolutional - Example
Height
Feature detector
Width
RGB value of a single pixel within an image at position [x, y].
Y position
X position

Figure 5.6: Convolution
[1]

A problem with Convolutional Neural Networks is that it can overfit data when predicting stocks. There are ways we can reduce the overfitting in our model. The first way is to use dropout in the Neural Network to cut links so it loses some information and has to infer differently then previously, resulting in a new model being made. Maxpooling is another feature which compresses your data by adding all data in a window. The window slides through the whole matrix resulting in a new smaller matrix which is an abstract form of the original. A benefit of a smaller matrix is that it decreases computation cost for future operations hence making your model simpler and more efficient. Another technique that can be used is to increase the striding value when filtering in the convolution. The striding value forces the filter to skip some values of the matrix hence less information will be taken from the original image therefore your model has to adapt to not having the complete representation which decreases the chances of overfitting. In addition, striding also increases the efficiency of the model hence making it faster to compute.

Convolutional Neural Networks suffer from the vanishing gradient problem however a solution to this problem could be to use smaller learning rates as it will approach 0 or infinity slower in the case of the exploding gradient problem. Unfortunately, this slows down the learning rate of the model therefore it will take a longer time to reach a global optimum. A better solution would be to use rectified linear units (relu) activation functions as the gradient is 0 if the input is negative and 1 if the input is positive. This decreases the chance of vanishing gradient occurring as the derivative is not in between 0 and 1 therefore multiplying by the gradient will have less of an effect than using other activation functions such as sigmoid or hyperbolic tangent (tanh). In addition, having a relu value of 0 for some inputs decreases the chances of overfitting as the values will be cancelled out. This is similar to neurones in people as all neurones do not fire when we do an action, some are firing and some are not. As a result of a sparse network, the computations will be faster as there are more 0 values.

A problem that can arise with relu is the "dying relu". A relu neuron is considered dead when the input is negative and the gradient is 0. Overtime, large parts of your network may end up doing nothing due to these 0 gradient values since multiplying with 0 cancels out values. The problem is made worse when the learning rate is too high or there is a negative bias which pushes the value to be negative. Like previously mentioned, the learning rate decrease could work but has its drawbacks. Fortunately, there are variations of relu that reduce the effect of dying relu. An example is leaky relu which has a gradient of 0.01 when the input is negative which gives the parameter a chance to stablise. Another example would be parametric relu which has a stable gradient when the input is negative which can also be used. In my model, I will use relu and potentially use leaky relu if the model produces a poor accuracy when optimising the hyper parameters. I will use convolutional Neural Networks in my model and try to optimise the hyper parameters to produce high accuracy. Since predicting stocks is a very complex problem, I will also use convolution layers and MLP layers in a single network to try and capture the trends in the data.

5.4 Implementation

Neural Networks can be implemented in many different ways such as Tensorflow, Keras, CNTK, Theano and MXNet. Google use Tensorflow and Keras whereas Amazon use MXNet along with Apple. Keras is a high-level Neural Network API which is capable of running a wrapper on Tensorflow, CNTK or Theano. It must be written in python which is ideal as the data collection and processing also use python. Keras supports Feedforward networks, Recurrent networks as well as a combination of both. In addition, it can run on CPU and GPU like MXNet so if I require faster computation I can switch from CPU to GPU. The Keras library provides a function called summary which shows a clear structure of how your model changes the shape of the data which helps analyse the model. Another benefits of Keras is that you can switch the backend and use the code regardless therefore it is more flexible as it can work with more functions in the backend. In this paper, I chose to use Keras due to its flexible nature which helped me to have lots of options when coding. In addition, Keras is very popular so there were lots of guides of best practices and solutions to multiple errors that I ran into when coding. For the backend I will use Tensorflow since Google keep it updated and well maintained by providing new functions. Also, Tensorflow provide both high level and low level API which is a benefit as Keras is only high level. This allows me to modify the model more if needed during the creation of the model.

Chapter 6

Experiments

There will be experiments in four models which are Multilayer Perceptron, LSTM Neural Networks, Convolutional Neural Networks and both MLP and CNN. However, firstly there will be experiments done on the parameters of a generic Neural Network to observe how changing the parameters may lower the error for both the training and test data. We must insure that our models do not overfit therefore I will calculate the mean squared error between the models output and the actual output to observe train and test error. Ideally, the train and test error should be similar and if test error is much greater than train error it will show that the model is overfitting the data. The data produced through changing individual hyper parameters will allow greater understanding in optimising the hyper parameters in the four models used. The optimal model for each of the four models will be compared to see which performed the best. Ideally, we want to change all hyper parameters to observe the models accuracy for each of the four models however because there are so many hyper parameters it is not possible to do.

6.1 Optimising hyper parameters

There will be experiments that will modify the number of nodes in a dense layer, test with and without batch normalisation, which activation function to use, what value of dropout to have, the number of repeated layers to use, the number of previous days to include and finally the number of epochs to use. The data will first be normalised. This ensure that the standard deviation of all variables have a mean of zero and a standard deviation of 1. A common architecture will be used and kept constant when observing the effect of the changing variable or variables. For each prediction, the 6 stock prices predicted will be high, low, open, close, volume and adjusted close for the ALXN stock. Then the lowest, highest and mean will be calculated.

6.1.1 Dense Layer

The first parameter that will be observed is the number of nodes in the dense layer.

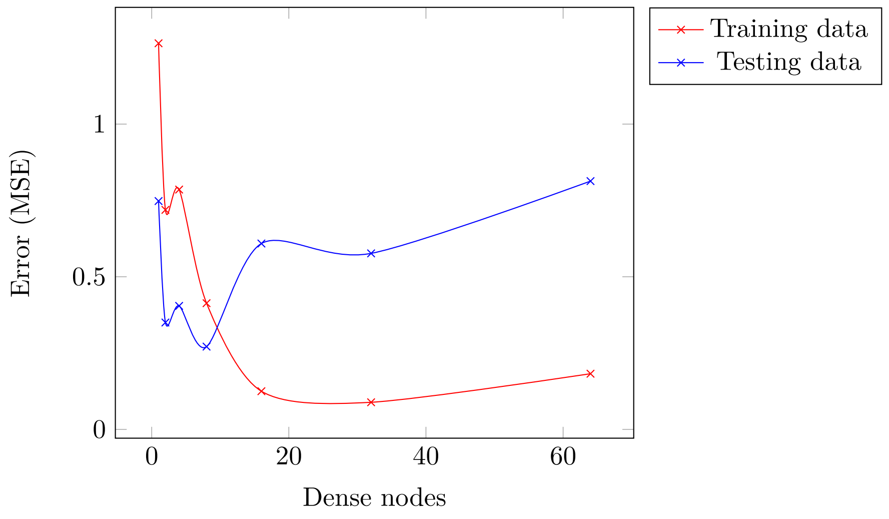


Figure 6.1: The mean error of predicting high, low, open, close, volume and adjusted close for ALXN stocks when changing the number of dense nodes for both training and testing

The training error decreased as the number of nodes increased in general as shown by the decrease in mean. However, the test error mean decreased then increased which could indicate that overfitting was occurring. Using too few nodes caused the mean to be too high as it may not have extracted many features from the data. This experiment has shown me that many nodes should be used however we need to compensate for overfitting if the number of nodes is too large. The range of 8-32 nodes in the dense layer seems like a good starting point when making the optimal architecture.

6.1.2 Batch Normalisation

Batch normalisation works by normalising the output of a previous layer by subtracting the batch mean and dividing it by the batch standard deviation. We will do an experiment with and without batch normalisation to observe how it effects the accuracy of both the training and test components. As shown below, the mean, best and worst errors are all lowered when using Batch Normalisation therefore it will be used.

Type	Batch Normalisation	High	Low	Open	Close	Volume	Adjusted close	Mean	Best	Worst
Train	True	0.08257	0.05912	0.11534	0.08631	0.05242	0.05052	0.07438	0.05052	0.11534
Train	False	0.26552	0.34237	0.30566	0.42638	0.29069	0.24174	0.31206	0.24174	0.42638
Test	True	0.90034	0.53880	0.58215	0.37557	0.40271	0.71551	0.58585	0.37557	0.90034
Test	False	0.11071	0.84527	2.90933	0.35942	2.16620	0.48162	1.14543	0.11071	2.90933

Table 6.1: Error for training and testing for ALXN stock

6.1.3 Activation function

There are different types of activation functions we can use such as rectified linear units (relu), hyperbolic tangent (tanh) and sigmoid. For each activation function, it was tested with and without batch normalisation before the activation function.

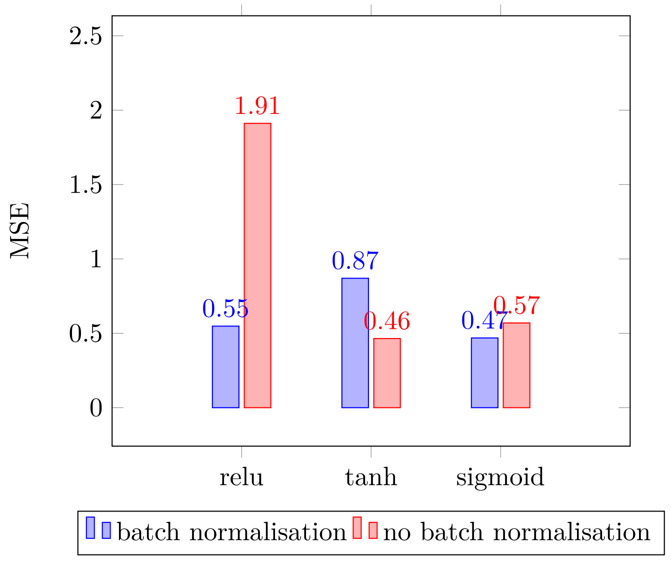


Figure 6.2: Mean error when predicting ALXN stock in the test dataset. Features originally predicted were high, low, open, close, volume, adjusted close when changing batch normalisation and activation functions

After observing the data (section A.2.2), it showed me that they all overfitted as the mean training average was much lower than the mean test average. The sigmoid activation function produced the lowest mean error however it also produced the worst. The rectified linear unit activation function produced the lowest worst error which shows that it is more reliable in producing consistent outputs. The reasons for this could be due to the vanishing gradient problem which is made worst if tanh or sigmoid is used. The gradient of these functions are less than 1 therefore multiplying by the gradient lowers the values hence contributing further towards the vanishing gradient problem. Relu has a gradient of 0 or 1 so it is less likely to suffer the problem but it can still be caused as the weights could be less than 1.

6.1.4 Dropout

Dropout randomly removes connections in the Neural Network. I will experiment with different dropout values to observe how it effects the models accuracy. The maximum value of dropout of Keras is 0.5 therefore all tested values will be less than or equal.

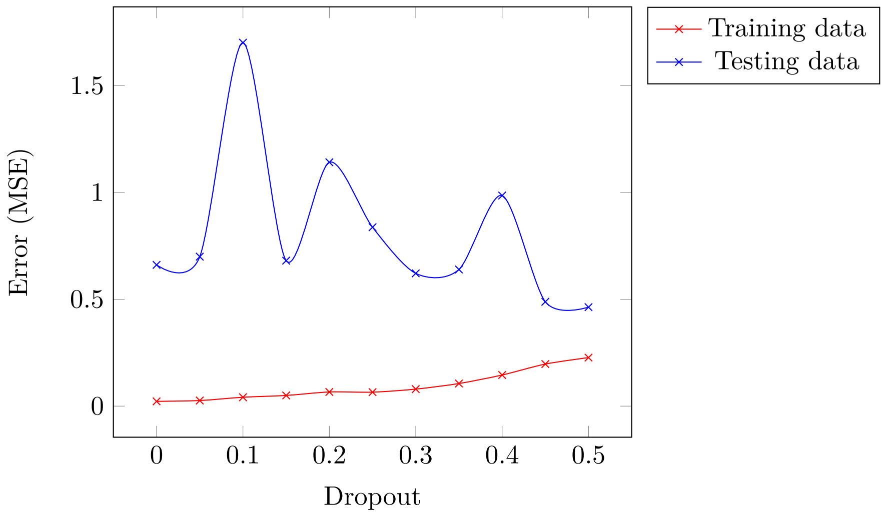


Figure 6.3: The mean error of predicting high, low, open, close, volume and adjusted close for ALXN stocks when changing the dropout factor for both training and testing

As shown by the table (section A.2.3), when increasing dropout the training mean error goes up as more links are cut so it is harder for the model to predict because it loses more information. However, the testing mean error goes down on average because the model overfits less when links are cut due to the information loss therefore it has to infer more when updating the weights. This shows that dropout can be used to prevent overfitting by increase the number of links cut at a certain layer in the network. As a result, the training and testing error will be closer. The ideal would be to have a testing error slightly higher than the training error but for both to be as low as possible.

6.1.5 Number of repeated layers

The repeated layers used will be dense, batch normalisation, relu activation function then dropout.

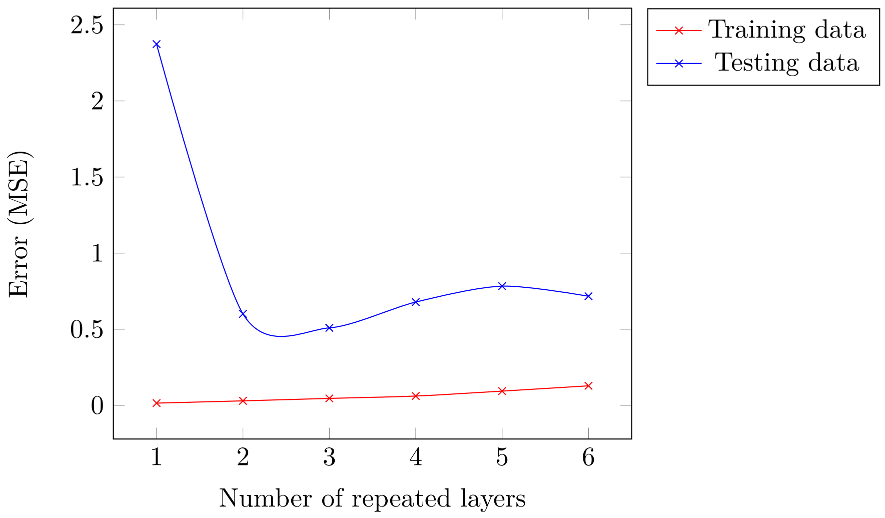


Figure 6.4: The mean error of predicting high, low, open, close, volume and adjusted close for ALXN stocks when changing the number of repeated layers for both training and testing

Initially, I thought that having more layers would decrease both the training and testing errors however this is not the case. The data shows that the optimal mean error is 3 repeated layers as it has the lowest test mean error which shows that increasing the number of layers could just be contributing to overfitting. Having more layers could be compensated by having an increased dropout rate so the model is less likely to overfit. In addition, a good number of repeats should be used to extract more complex features in the dataset.

6.1.6 Number of past days

When predicting the stock we need to have previous k days to predict the k+1 day. The value k can be varied therefore we will do an experiment to observe error as k increases.

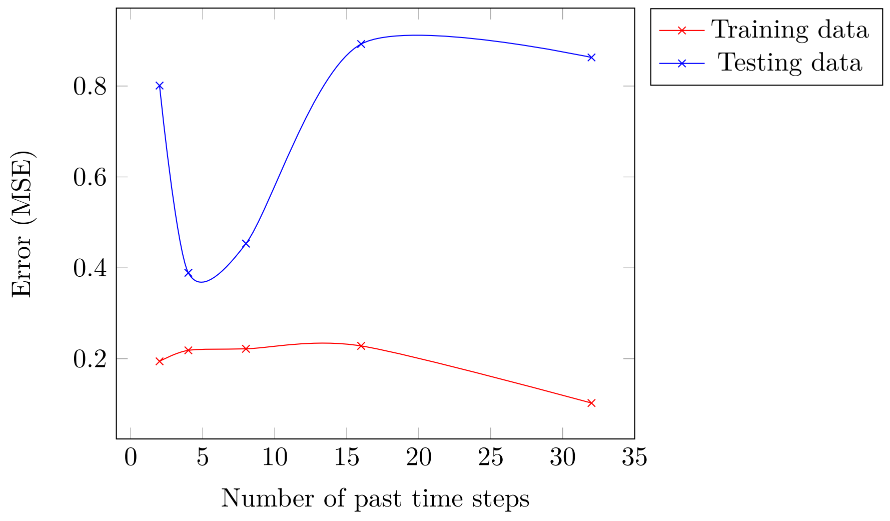


Figure 6.5: The mean error of predicting high, low, open, close, volume and adjusted close for ALXN stocks when changing the number of past time steps for both training and testing

As seen above, if the number of past time steps is too high the model starts to overfit. This is indicated by the decrease in train error and the increase in test error. The optimum test range was around 4 to 8 so to produce the optimal model the range of 3-15 could be the starting point. Despite this however, the best model was produced with 2 past steps.

6.1.7 Number of epochs

Epochs is the number of complete passes through the training dataset which is specified when training the model in Keras. We will modify this value to see how it effects the error and hence will use this information to produce an optimal architecture.

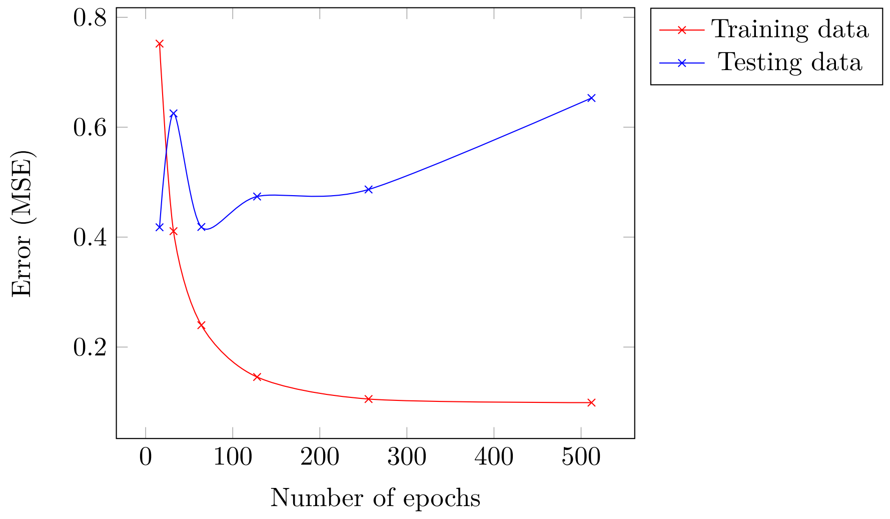


Figure 6.6: The mean error of predicting high, low, open, close, volume and adjusted close for ALXN stocks when changing the number of epochs for both training and testing

The effect of overfitting gets worse as the number of epochs increases however the best test error also increases until 64. This could suggest that a high number of epochs is desired as long as it does not overfit therefore the model extracts complex features from the data.

6.1.8 Number of batches

The batch number is the quantity of samples processed before the model updates the weights of the network. In addition, the batch number must be greater than or equal to 1 and should be less than the number of samples as it would update after processing the whole dataset.

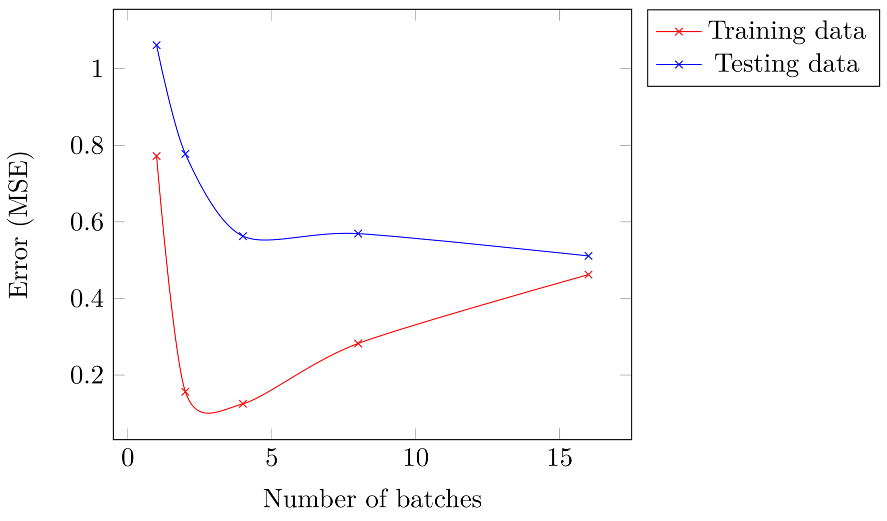


Figure 6.7: The mean error of predicting high, low, open, close, volume and adjusted close for ALXN stocks when changing the number of batches for both training and testing

The data shows us that increasing batch size prevents overfitting as the training error and testing error difference becomes smaller. This shows that the batch number could be used to prevent overfitting by using a large number of batches which should be the starting value when beginning to optimise the architecture.

6.1.9 Type of optimiser

There are many different types of optimisers and the two that we will be investigating is sgd (stochastic gradient descent) and adam optimisers.

Type	Optimiser	High	Low	Open	Close	Volume	Adjusted close	Mean	Best	Worst
Train	sgd	0.07696	0.08413	0.14854	0.13978	0.06162	0.11152	0.10376	0.06162	0.14854
Test	sgd	0.86929	0.24770	0.88259	0.37700	0.43586	0.54068	0.55885	0.24770	0.88259
Train	adam	0.14019	0.08696	0.16425	0.09299	0.06528	0.04726	0.09949	0.04726	0.16425
Test	adam	1.05835	0.67630	1.05617	0.33493	0.83796	1.00345	0.82786	0.33493	1.05835

Table 6.2: Error for training and testing for ALXN stock

Overall, SGD performed better as the mean test error was lower however this could be due to the adam optimiser being more liable to overfitting. The overfitting is less likely to occur in SGD however adam is regarded very highly and one stock was better therefore both should be used when optimising the model however the better performing will be kept in the future experiments.

6.2 Optimal Architecture

After observing the experiments, the optimal architectures for MLP, LSTM, CNN and MLP with CNN were created. For each architecture, the hyperparameters were changed to minimise the error for each prediction.

6.2.1 MLP

The optimal architecture can be seen below using MLP.

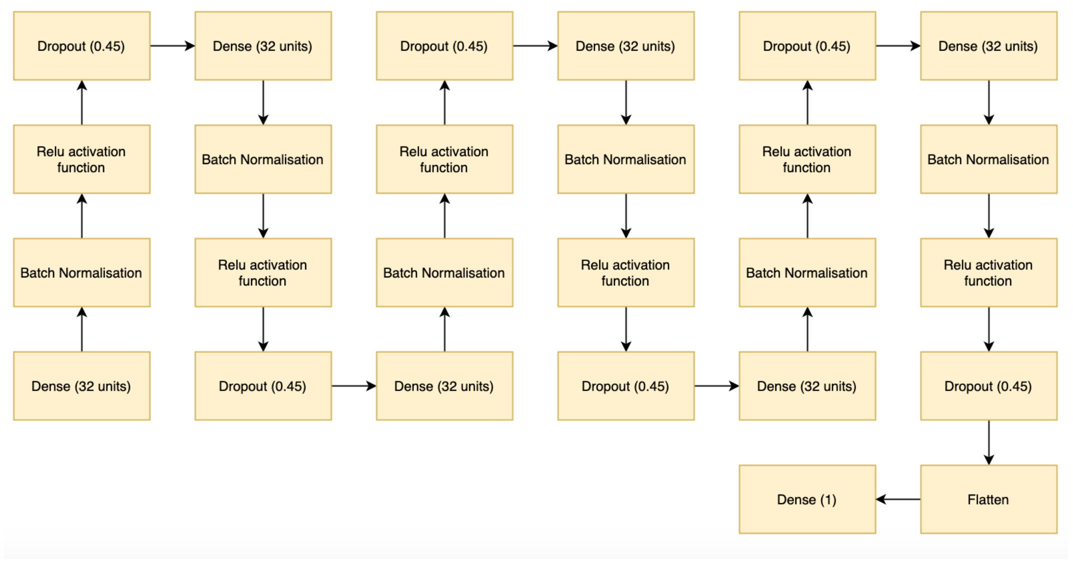


Figure 6.8: Architecture of MLP

The following results were produced using the architecture above.

Type	High	Low	Open	Close	Volume	Adjusted close	Mean	Best	Worst
Train	0.24834	0.16836	0.54129	0.65711	0.08104	0.42514	0.35355	0.08104	0.65711
Test	0.21606	0.10262	0.68400	0.52514	0.27209	0.32275	0.35378	0.10262	0.68400

Table 6.3: Error for training and testing for ALXN stock

The mean training error was similar to the mean testing error which is a good sign that overfitting did not occur. However, the numbers do vary which could show that this is not the most consist architecture for achieving low error.

6.2.2 LSTM

The optimal architecture can be seen below using LSTM.

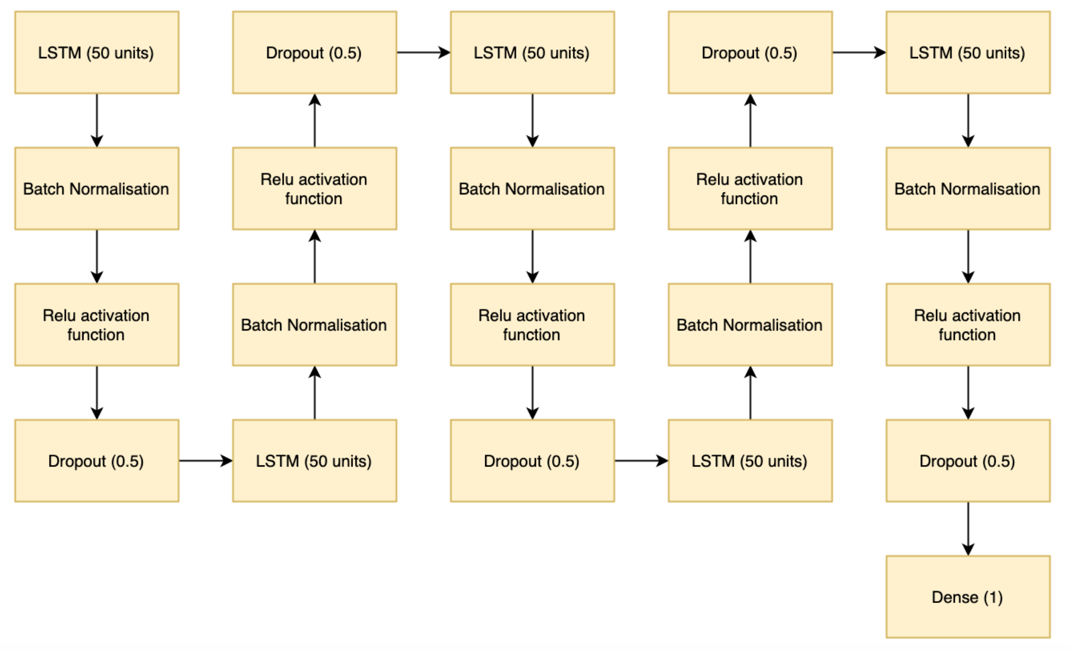


Figure 6.9: Architecture of LSTM

As shown below, the LSTM model outperformed the MLP model in both scenarios.

Type	High	Low	Open	Close	Volume	Adjusted close	Mean	Best	Worst
Train	0.12815	0.13685	0.13977	0.27998	0.07289	0.08482	0.14041	0.07289	0.27998
Test	0.47170	0.34013	0.39190	0.24948	0.15528	0.26079	0.31155	0.15528	0.47170

Table 6.4: Error for training and testing for ALXN stock

6.2.3 CNN

The optimal architecture can be seen below.

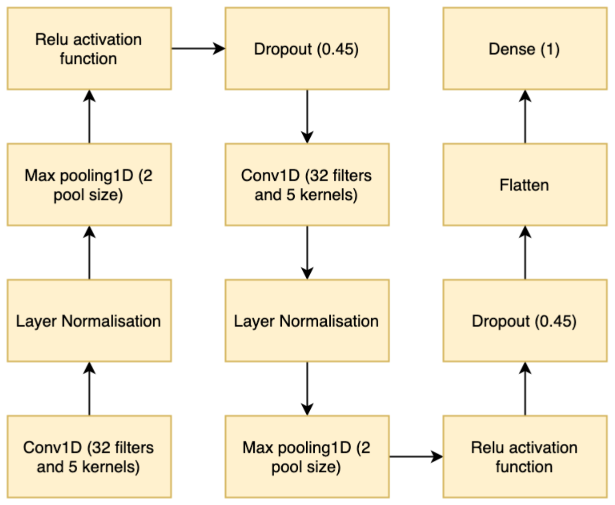


Figure 6.10: Architecture of CNN

Type	High	Low	Open	Close	Volume	Adjusted close	Mean	Best	Worst
Train	0.06219	0.02215	0.07101	0.03896	0.01264	0.03407	0.04017	0.01264	0.07101
Test	0.23409	0.11486	0.54875	0.10123	0.19807	0.55128	0.29138	0.10123	0.55128

Table 6.5: Error for training and testing for ALXN stock

The mean error was lower than LSTM by 0.02017 which shows that there was no clear better model and the feature prediction would have to be tested in both models to observe which performed better. In addition, there were a lot of samples so it would be intuitive to assume LSTM would outperform CNN as it remembers long-term dependencies which is beneficial for predictions.

6.2.4 CNN/MLP

The optimal model's common architecture can be seen below.

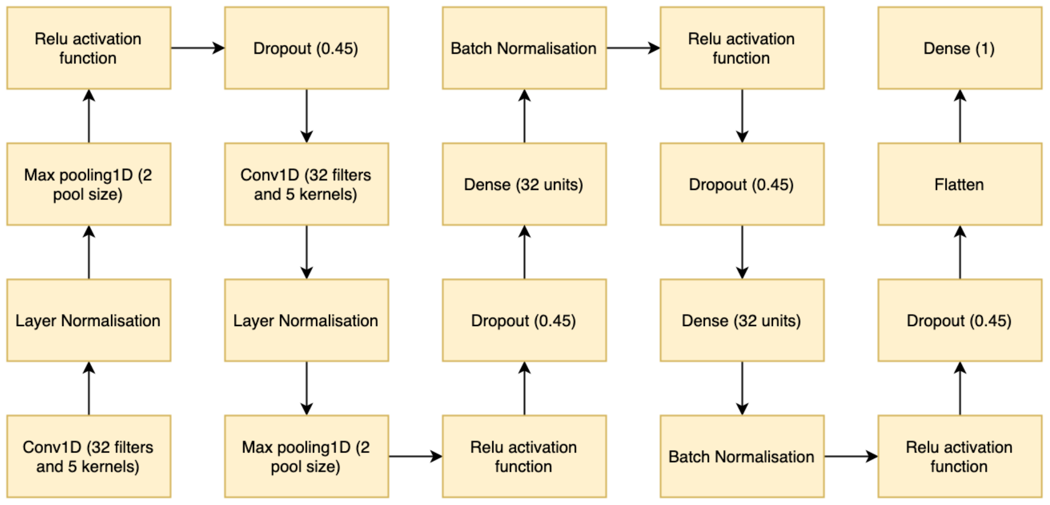


Figure 6.11: Architecture of CNN and MLP combined

The following experiment was produced to observe how CNN and MLP layers performed together.

Type	High	Low	Open	Close	Volume	Adjusted close	Mean	Best	Worst
Train	0.07316	0.11588	0.15256	0.07431	0.03215	0.07091	0.08650	0.03215	0.15256
Test	0.39607	0.57243	0.80738	0.41708	0.64022	0.21973	0.50882	0.21973	0.80738

Table 6.6: Error for training and testing for ALXN stock

As shown above, the CNN/MLP hybrid performed much worse than the CNN. The mean test accuracy was 0.21 higher which could be caused by an overly complicated layer structure. Despite this, the prediction for adjusted close outperformed CNN by 0.33155 which indicates that some predictions could have been benefitted from using MLP.

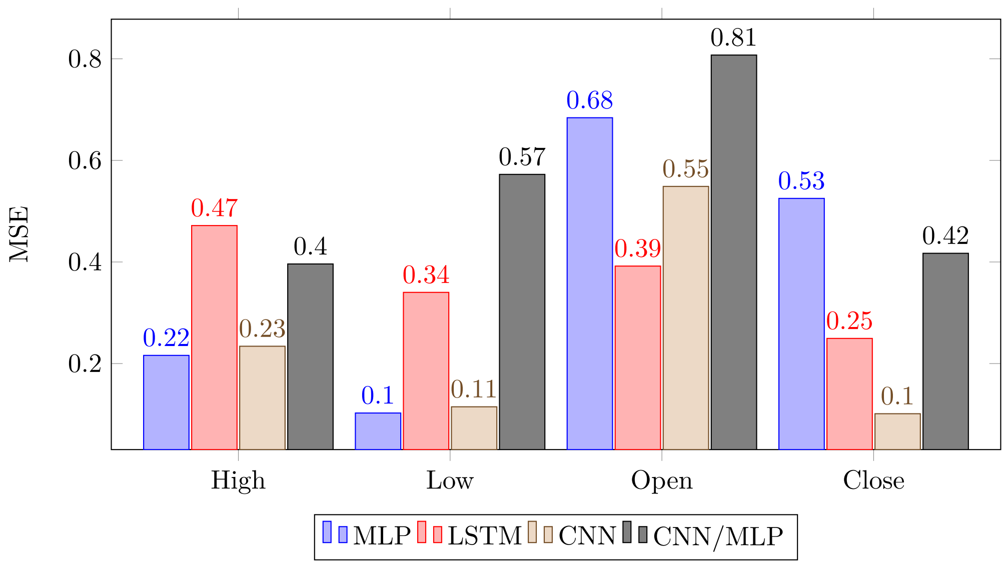


Figure 6.12: The MSE when predicting high, low, open, close for ALXN stock in the test dataset using different architectures

6.2.5 Comparing Optimal Architectures

From the graph above, it becomes apparent that the best architecture is not obvious as different architecture work best at predicting different factors. Therefore, the best approach would be to test the factor on each architecture to determine the best architecture. However, generally CNN worked best when observing the graph as it always had the lowest or second lowest error.

6.3 Summary of experiments

It was extremely difficult to optimise all the parameters given how many were present. The best general model was CNN however LSTM was just behind as the mean test error was so similar. The different models would work better in different scenarios so it was difficult to choose when predicting individual stock features. However, if there are a large number of samples LSTM Neural Networks should be tried first. If there are a small number of samples then CNN would be advised. The best strategy would be to use each model to observe which layer combination produces the lowest error rate. Also, the MSE may not always have been an accurate way to determine the usefulness of a models prediction capability. For example, the trends could have been captured by the model but the MSE could be greater. This can be seen below where the CNN/MLP model which performed the second worst on average captured the trends in certain cases as shown below. The predicted value stays high around 12-13 days as it correctly predicts that the price will go back up which it does on day 15.

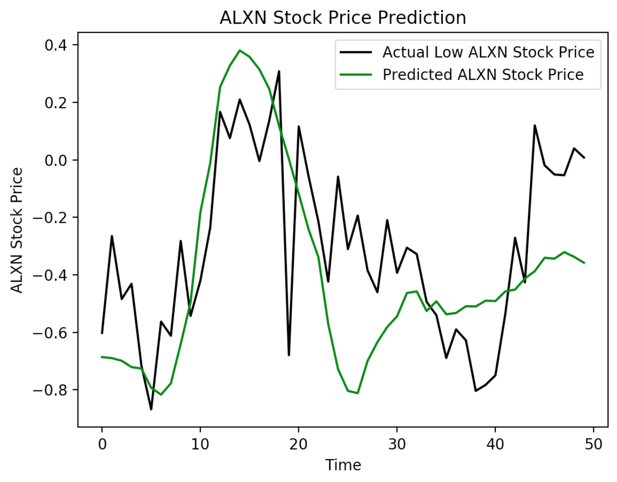


Figure 6.13: CNN/MLP prediction for ALXN low

A graph was produced using the 20 day moving average which is a technical indicator used to help identify trends as seen below. This assumes all the data was already present and not predicted. The graph below does not capture large changes in short time periods as can be seen at time 50. However, the graph above predicts the decrease before it happens at time 25 indicating that the real value of the stock was below the value it truly was. This shows that information gathered from bonds, stocks, natural resources and cryptocurrencies can predict certain trends which are unforeseen in traditional analysis methods such as moving average.

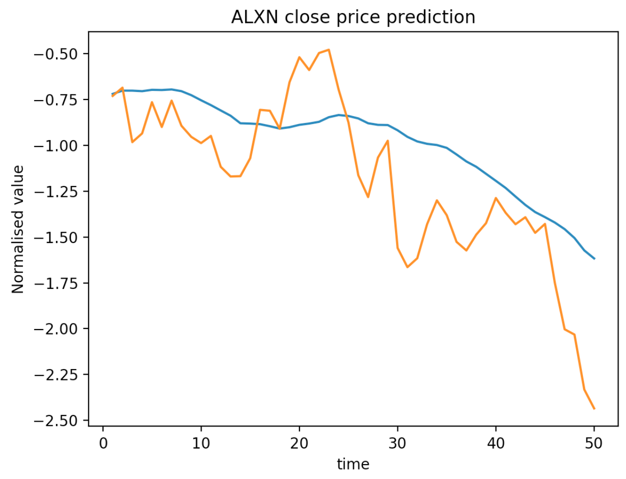


Figure 6.14: 30 day moving average for ALXN close. Orange is the actual price and blue is the 30 day moving average.

Chapter 7

Legal, Social, Ethical and Professional Issues

The British Computer Society are a professional body that have produced a code of conduct and code of good practice to ensure correct practices are followed in the UK. The code of conduct is split into 4 main categories: the public interest, duty to employers and clients, duty to the profession and finally professional Competence and Integrity. Not all parts of the code of conduct is relevant to my project. For example, no personal information is stored therefore there is no reason to protect any information collected. However, the duty of care to clients is relevant as I must not misrepresent or withhold the data produced from the model to benefit from the readers lack of knowledge or experience. For this reason, I produce the output of each model once and not repeatedly to produce the best error rates. Another relevant law is to not motivate the client into a product that I have a financial benefits from. I do not have any ties with any of the data collected such as american stocks, cryptocurrencies and index funds. In addition, I would like to state that the stock markets are extremely volatile therefore even if I produce a model that appears to make money, it does not ensure that you will make money as you are always liable to risk. This is amplified if you do not have good knowledge on how to tune the hyper parameters of the model or lack financial knowledge of the markets. For professional competence and integrity, it is important to increase my knowledge about Neural Networks to produce relevant work in the field. This is followed by optimising the model by researching the field therefore producing a low error due to the knowledge gained in this field already. The research locations are included in my bibliography to allow any readers to read other work to increase their knowledge around the subject.

Predicting stock markets have lots of ethical issues which can cause harm to the public. The models produced could encourage addicted betters to spend more and potentially lose a lot of money from the work I have carried out. It is extremely important to understand the risk involved with stock predictions and how volatile they are. Even experienced investors rarely predict the stock market in the long term which exemplifies the difficult nature of the task. Also, bookmakers use machine learning algorithms to help find reasonable risked bets to offer the public so they are more likely to produce money. They also lose money in bets sometimes due to inaccurate outputs from the machine learning algorithms. Therefore the machine learning algorithm should not be seen as an "instant money maker" but instead a tool for assisting in predicting stocks. Relevant market knowledge and optimising the model will also benefit greatly in predicting stocks.

Chapter 8

Conclusion and Future Work

During the data collection phase, it was very difficult in collecting certain pieces of data such as twitter data. I tried to collect twitter data in the form of word occurrence for all tweets in a give time range. A problem arised when I discovered that you could only collect a limited number of tweets with a certain word which was problematic as all the key words I wanted to collect would have the same number of occurrences. My idea was to collect key words like brexit which have an effect on prices such as stocks and even currency. An example of tweets effecting stock prices can be seen when Donald Trump tweeted he would put 10% tariffs on 300 million worth of imports from China. In future, I would collect twitter data by finding the time scale between the first and last tweet collected as it is produced from most recent to least recent. Then stored the number of days as a feature for a given word. If the time scale is too large and the word occurrence in tweets are very common then I would produce my own time scale.

The modelling phase was difficult as Neural Networks are seen as a black box machine learning technique so it was difficult to fully grasp everything that was going on. There was a lot of trial and error techniques to try to understand how the testing accuracy was effected. There was research online to help understand certain layers but each prediction is different therefore certain ideas may not apply. For example, the adam optimiser is recognised to produce better results over the sgd optimiser however in the test it was the opposite. During model optimisation by tuning hyperparameters there was a lot of comprise between features. Increasing nodes in the dense layer would make it more likely to overfit however you could increase dropout to drop edges therefore decrease overfitting. However the overfitting could not be too high otherwise the model would not be consistently accurate as it depended on which edges were removed. For this reason, there was a lot of testing involved and it was not always obvious how to present the results of my experiment.

It was interesting that the predictions using deep learning showed trends that were not captured using analysis of the real values which shows that money can be made using deep learning. In future, I may simulate trades using the predictions to observe how much it can make compared to using other techniques such as moving average. In addition, I may use k fold cross validation at certain parameter ranges. Due to the large number of hyperparameters, 2 or 3 could be picked to experiment on to see the results. The process could be repeated to show trends between multiple variables which could potentially provide useful insight on how to further optimise the architecture.

References

[1] Nils Ackermann. Introduction to 1d convolutional neural networks in keras for time sequences.

[2] Brian DeChesare. What is the sp 500, and why does it matter to traders at banks?

[3] N. Djeddaoui, L. Boukezzi, and L. Bessissa. Use of mlp artificial neural network in prediction of j-v characteristic of organic solar cells. In *2018 International Conference on Communications and Electrical Engineering (ICCEE)*, pages 1–4, 2018.

[4] google. Vanguard total international stock index.

[5] Thomas Kenny. Vbtlx.

[6] Nick Lioudis. The collapse of lehman brothers: A case study.

[7] R Reber and W Perrig. Perception without awareness, psychology of. *Psychological Science*, 2:119–22, 2001.

[8] Christoph Roser. Local optima global optimum.

[9] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647, 2017.

[10] Mich Tuchman. 5 warren buffett quotes for anyone who thinks they can pick stocks and get rich like he did.

[11] WSJ. Wsj journal markets.

[12] Mohamed Zahran. Research gate.