

Honours Project (CCIS) Final Report 2017-2018 Submitted for the Degree of: BSc Computer Games (Software Development)

Project Title: A Comparison of MLP, CNN and LSTM Variant Neural Networks

for Modeling Stock Market Time Series Data

Programme: BSc Computer Games (Software Development)

Student: EE_1718

"Except where explicitly stated, all work in this report, including the appendices, is my own original work and has not been submitted elsewhere in fulfilment of the requirement of this or any other award"

<u>Abstract</u>

The world's stock markets are known to be a source of great wealth for anyone able to succeed in determining the future values of any given stock though the risk and volatility associated with this endeavor is often too great for simple guesswork and unrefined processes. Due to this Artificial Neural Networks have seen frequent usage in algorithms for predicting and modeling financial time series data however the scope of such networks is truly vast and it is not always clear what variation of network might be best suited to any given task. This paper looks to give insight into the suitability of three different architectures of neural network for predicting 5 different financial time series datasets: the feedforward MLP model, the Convolutional Neural Network model and the LSTM variant recurrent Neural Network model. Each of these models are created using the Keras API on top of the Tensorflow Neural Net framework, trained on 5 different sets of financial time series data over a range of 5 years for each dataset, optimised for highest possible accuracy by varying network variables and tested against unseen sections of each time series dataset. The paper takes an in depth look at the training process for each network and how each variable affects the final accuracy of each model. The final testing results are then evaluated and explained with references to similar studies of the effectiveness of different models showing the benefits and issues inherent in each model for predicting future financial time series data.

1. Introduction	4
1.1 Project Background	4
1.2 Research question	6
1.3 Project Outline	6
1.4 Project Objectives	7
 Create a requirements specification for the application. 	8
2. Develop the neural network models.	8
3. Train the neural network models on first 90% of historical data.	8
4. Predict the last 10% of historical data and compare against actual prices.	8
Predict share price data for future daily intervals.	8
Evaluate results based upon live market data.	8
2. Literature review	9
2.1 Investigate the current prediction solutions used for forecasting time series data	a. 9
Fig. 1. A simple neuron (C. Stergiou, D. Siganos, 2017)	9
Fig. 2. A typical multi-layer feed-forward neural network architecture(A. Yilm al, 2015)	naz et 10
Fig. 3. An example convolutional network layout showing neurons with shar weights. C. Olah, 2014	red 12
Fig. 4. An example Recurrent neural network. D. Britz, 2015	13
2.2 Investigate the factors involved in determining a stock's market share value.	14
2.3 Investigate the implementation options of using neural networks for time series prediction.	15
2.4 Hypothesis	17
3. Methods	17
3.1 Neural Network development	17
3.1.1. Data formatting.	17
3.1.2. Model creation	17
3.1.3. Training	18
3.2 Time series forecasting	19
3.3 Forecast analysis	19
3.4 conclusion and feasibility of each forecasting model	19
4. Design and Implementation	20
4.1 Design Specification	20
4.1.1 Data Processing	20
4.1.2 Model Design	20
4.1.3 Training Script	20
4.1.4 Testing Script	21
4.2 Implementation	21
4.2.1 Data Processing	21
4.2.2 Model Classes	22

Fig.5 & Fig.6 indicating typical tanh respectively) (A. Sharma, 2017)	and relu activation functions (L & R	23
Fig. 7 Example of max pooling (A. I	Deshpande, 2016)	24
4.2.3 Training Script	·	25
4.2.4 Testing Script		26
4.3 Model Training		27
Fig. 8 Epoch training graphical repr	esentations	28
Fig. 9 Batch training graphical Repr	esentations	30
5.0 Testing and Evaluation		31
5.1 Model Testing and Evaluation		31
Figs 10-14. Graphical outputs of pre	edicted stock values against live values	34
Fig. 15-17 Error tables for testing or	า all 5 datasets	35
5.2 Conclusion		37
5.3 Future work		37
6.0 References		38
Appendix		43
Appendix A - Code		43
Process.py		43
MLP.py		44
Conv.py		45
LSTM.py		46
Train.py		47
Test.py		49
Appendix B - Results		52
Appendix C - 7 day forecast preliminary res	sults	57

1. Introduction

This section of the report will give a background into the project field and describe the basis of the research and development objectives before proceeding into the main literature review.

1.1 Project Background

The world's stock markets have always been a source of great possibilities for those few who can accurately predict and manipulate the seemingly endless amount of data generated each day dictating each stock's value. Countless methods have been generated over the years to try to reduce the level of uncertainty inherent in trading stocks and shares from pure intuition to complex mathematical models aiming to predict future prices based off of previous data and certain "markers" indicating a possible up or down swing in the stocks market value. Each of these methods has come with varying success though as the years have went on these past models have needed to be redesigned and redeveloped as the nature of the stock market itself evolves and adapts. What works one day might not work the next as other traders take into account the methods and strategies employed by their rivals and use this information to generate models to exploit their rivals systems. One of the most recent attempts at predicting future stock values is a technology called Artificial Neural Networks.

Artificial Neural Networks are software algorithms that are designed in essence to predict a specific output or action based upon a set input or state. They are formed of multiple interlinked components called 'neurons'. Each of these neurons carries a weight that is applied to any given input with the combined weightings of each neuron contributing to the overall output function of the network. These weights are updated regularly through training in order to optimise the output function. Through training a network can theoretically be taught to model and mimic any system and due to this is a topic of focus in data science.

Artificial Neural Networks were first introduced in 1943 (Warren McCulloch et al., 1943) and since then have grown in popularity within the data science field. The first application of neural networks for prediction was introduced by Hu in 1964 for the purpose of weather prediction (M.J.C. Hu, 1964). This was later adapted by Kimmoto to be used to predict stocks on the Tokyo Stock Exchange in 1990 with a resultant accuracy of 63% (T. Kimmoto et al., 1990).

Since being introduced as a possible method of accurately predicting stock prices the popularity of using neural networks within a trading environment has increased year after year. This growth in popularity is partly down to the fact that they are inherently nonlinear in nature. This is a contrast to old prediction models such as autoregressive integrated moving average (ARIMA) that relies on the data series being linear in order to achieve satisfying results. It has been shown through studies that real world systems are normally nonlinear in nature (Zhang et al., 1998) thus giving the potential advantage to neural networks as a primary means of predicting stock market data sets.

Along with the benefit of neural networks being able to model nonlinear data sets they have other clear advantages to older methods of prediction. As neural networks are self adaptive and data driven they can learn complex functions with little or no prior input regarding the model of study. Their ability to learn without input ensures they can be easily updated and adapted with changes in the data trends.

Within the scope of artificial neural networks there exist various different types of network providing unique characteristics that can be applied to the forecasting of data. Various studies have been ran on the differing types of neural network and their application within time series forecasting.

White studied the application of a simple, feedforward artificial neural network (White, 1988). Feedforward neural networks are artificial neural networks in which the connections between the layers do not form a loop, i.e. the data is moved from input to output through the hidden layers and is unable to move backwards at any point. Feedforward neural networks have mostly seen usage in pattern recognition and are generally considered the simplest type of artificial neural network. Feedforward neural networks come in two main types: single-layer perceptron and multi-layer perceptron. Single-layer networks consist of a single layer of outputs and the inputs are fed directly into this output layer. Multi-layer networks consist of multiple computational or hidden layers connected in a feedforward fashion.

Other networks have been studied such as multi-step river flow forecasting (Atiya and Shaheen, 1999), stochastic model based neural network (Poli and Jones, 1994), convolutional neural network (Borovykh, 2017) and recurrent neural networks (Connor 1994). Each of these studies showed varying levels of accuracy for each type of network and each network seems to be suited to different aspects of the prediction algorithm.

Convolutional neural networks are a variation of feedforward mostly used for image recognition. The main difference of convolutional network is the convolution operator. This convolutional operators purpose is to extract interesting features from the data set. In cases of convolutional neural networks applied to time series data the main purpose is to identify certain indicators that the data might be moving up or down in relation to its previous state.

Recurrent neural networks can have signals moving in both directions through the network by looping the data any number of times within the network. This ability to loop data can provide higher accuracy at the cost of making the network significantly more complicated than its simple feedforward counterpart. The network learns from not only the data but also the past computation within the previous loops.

It is worth noting that combining multiple types of neural network can enhance the probability of accurate time series forecasting by utilising and combining each models unique capabilities. Zhang found that combining different neural network models can have a positive effect on the predictive capabilities of an algorithm when utilising models that are quite different to each other (Zhang, 2007).

Another approach to increase effectiveness in predicting accurate fluctuations in time series data is to combine both linear and nonlinear based approaches within the same algorithm (Taskaya and Casey, 2005). For instance combining previous working linear models such as ARIMA with feed forward neural networks could help to create an algorithm that fits both linear and nonlinear aspects of a time series data set.

One of the largest advancements to neural network accuracy has been the introduction of back propagation to feedforward networks to reduce the error within the trained network model to as low as possible. The idea was first introduced in 1969 (Bryson and Ho, 1969) and became widely appreciated in the late 1980s. The overall idea of the network is to regularly update the functions within the model by repeatedly comparing the predicted outcome with the actual outcome based upon past data. This theoretically is able to reduce the error within the algorithm to effectively zero. While in theory this can produce one hundred percent accurate results, in actuality the accuracy is less than perfect due to the time series data forever changing though despite this the process can still drastically improve performance.

As neural networks have shown to be most accurate at predicting high frequency time series data their main usage within the trading industry is currently what is known as HFT (High Frequency Trading). This accuracy is due to the increase in available information in which to train the network. In HFT companies typically employ algorithms to make thousands of trades per second monopolising on minute changes in each shares listed price. It is estimated that between 2009 - 2010 sixty to seventy percent of all US Stock Market trading was attributed to HFT (Shobhit Seth, 2015). This usage of high frequency data could be used to create models that are able to predict long term changes in time series data with potentially higher accuracy than longer term frequency data with the downside of drastically increased training times for each model and increased cost due to availability of training data.

1.2 Research question

Can more complex neural networks such as convolutional and recurrent models provide higher accuracy than that of a simple multi layer perceptron model when used to predict stock market share price values for low frequency trading?

1.3 Project Outline

It is the purpose of this paper to measure the effectiveness of the above methods of time series prediction for predicting the prices of five different datasets: the Apple share price, the Google share price, the IBM share price, the Microsoft share price and the SAP share price. It is the hope that targeting five different datasets within the stock market that the all round effectiveness of each model will be tested and the possibility of a false result will be reduced. Predictions will be calculated for daily intervals to determine the accuracy for usage in long term or "swing" trading".

The algorithm will consist of three artificial neural network models - mainly a simple multi layer perceptron model with back propagation, a convolutional model and a recurrent model (LSTM variant).

The models will be created and trained using the Python programming language in conjunction with the keras and tensorflow open source frameworks. Each model will be trained on the same sets of data sourced from Yahoo finance API via the pandas python library to ensure the results are as accurate as possible.

During the development process each model will be trained on the first 90% of time series data and will be tested on the last 10% of the data to optimise the network parameters. Once each network is fully optimised the models will be trained on the full set of time series data before predicting future share price values for each stock. They will then be tested by comparing the predicted share price values against the real share price values for daily intervals. The accuracy of each model will be tested using the mean squared error and percentage error based upon its real price.

It is worth noting that no real money will be used within this project and any profitability that could occur within these trades could be offset by the cost of making such trades.

1.4 Project Objectives

The objectives of the main literature review are:

1. Investigate the current prediction solutions employed for forecasting time series data.

In order to fully investigate the effectiveness of the selected networks there must first be research into the methods currently used in forecasting time series data inside and outside of finance. This will involve looking at linear and nonlinear models along with the various types of neural networks available for time series forecasting.

2. Investigate the factors involved in determining a stock's market share value.

To provide a more accurate result of the effectiveness of each network and what the network is actually predicting there will be research into the main factors influencing a particular stocks market share value such as company net worth and earnings per share. This data could be used to help train each model and will give greater understanding of the underlying data within the time series.

3. <u>Investigate the implementation options of using neural networks for time series prediction.</u>

Lastly there will be research into the methods for implementing, training and testing neural networks for time series prediction. This will involve researching the various neural network frameworks available and looking into examples and tutorials for the development of neural networks with the purpose of predicting time series data within finance.

The main objectives of the project to be completed are:

1. Create a requirements specification for the application.

Documentation will be produced detailing each models specifications and requirements. This will include information on the type of network model, how it will be trained, the input parameters the expected output format and how each model will be tested in terms of accuracy and efficiency. This specification will also include details on the stocks being used within the algorithm.

2. Develop the neural network models.

Develop the training and prediction aspects of the neural network models along with the ARIMA linear model applied to each set of predictions. This will require different prediction methods for each prediction timeframe i.e. daily, weekly and monthly.

3. Train the neural network models on first 90% of historical data.

Import and process the raw market data for training on the neural networks before training and optimising each network model for the highest accuracy possible. During this training the mean squared error will be kept track of to adjust parameters in order to improve accuracy.

4. Predict the last 10% of historical data and compare against actual prices.

Run the prediction methods for the last 10% of the historical data to gauge accuracy for data the application has not seen. The error from this will then be used to back propagate each model for optimisation.

5. Predict share price data for future daily intervals.

Predictions will be made for each target stock at set intervals for each network model. The outputs will then be noted for comparison against future share price values from live markets.

6. Evaluate results based upon live market data.

Once predictions have been made and checked against live market share price data the accuracy will be calculated. Each model will then be compared across multiple factors to determine the best performing variant and its feasibility for live usage.

2. Literature review

This section of the report will detail the findings of each or the primary research objectives listed in section 1.4 under the main literature review objectives.

2.1 Investigate the current prediction solutions used for forecasting time series data.

Autoregressive moving average or ARIMA for short is one of the longest standing linear time series data forecasting algorithms and models are generally estimated using the Box-Jenkins approach (Box, Jenkins, 1970). The models typically take on the form of ARIMA(p,d,q) where p is the number of autoregressive terms, d is the number of non seasonal differences required for stationarity and q is the number of lagged forecast errors within the prediction equation (R. Nau, 2017). For more than 50 years ARIMA models have dominated time series forecasting (M. Khashei, M. Bijari, 2010) this is due to the flexibility and statistical properties inherent within ARIMA models. Their main drawback however is that the model assumes a linear data structure within the time series data set and this is not always the case.

Artificial Neural Network models on the other hand are a flexible nonlinear forecasting method studied in depth for their usage in predicting time series data (Zhang et al. 1998). One of the major advantages of using ANN's for time series prediction is that there is no need to specify what form the model will take as the model will form based upon the data itself. This advantage is useful for data sets that are abstract or nonlinear in nature as it can pick up features that might not be easily determinable using other models.

Artificial Neural Networks are information processing algorithms that have been inspired by the human nervous system and how it processes information. The key link between Artificial Neural Networks and the human nervous system is the structure of the algorithm which is built up of a number of interconnected processing elements called neurons that work in unison to solve a given problem (C. Stergiou, D. Siganos, 2017). They also learn by example the same way as the human brain so once a network has been trained it can recognise new data points it hasn't seen before and relate them to what it has been shown previously.

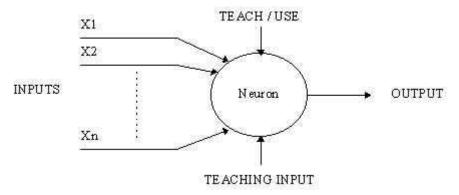


Fig. 1. A simple neuron (C. Stergiou, D. Siganos, 2017)

Figure 1 above shows a typical neuron setup within a simple feed-forward neural network. The model is designed to take in various inputs shown on the left hand side as x1, x2, ..., xn. The neuron will then apply a weight function to each of the inputs (generally denoted w) in order to produce the desired output shown on the right hand side of the diagram. This process is achieved through training the network by passing in the desired output (see teaching input above) for each set of inputs. This process is repeated multiple times improving the weighting each iteration until the model is able to accurately predict the desired output on its own.

Artificial Neural Networks over the years have taken various different forms and come in a variety of different models each specialising in a specific aspect of data analysis. The simplest of these is generally considered a feedforward neural network. This model consists of an input layer, an output layer and at least one hidden layer that processes the information based on the learned function. The defining feature of a feedforward network is that all data within the network moves from input to output and is unable to move backwards at any point.

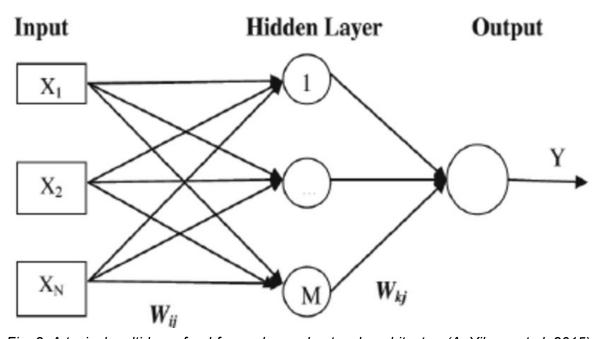


Fig. 2. A typical multi-layer feed-forward neural network architecture(A. Yilmaz et al, 2015)

The number of hidden layers can be increased as much as desired though the best number of hidden layers is usually a topic of debate in data science. George Cybenko stated that feed forward neural networks with one hidden layer is capable of universal approximation (g. Cybenko, 1989). Opposed to this several studies have shown the benefit of multiple hidden layers as opposed to the single hidden layer proposed by Cybenko, one such paper states deep networks can compute an output to the same accuracy as shallow models while requiring an exponentially lower number of training parameters (H. Mhaskar, 2016)

For any given network, assuming each input has a weight within the algorithm, the output of a feedforward neural network is determined by some function of the weighted sum of the input values multiplied by their weights.

$$y = f(v)$$
 where $v = x_1w_1 + x_2w_2 + ... + x_nw_n$

Where x indicates input and w indicates the weighting applied to the input within the neuron.

It is possible to increase the accuracy of the network by introducing back propagation. Back propagation is a method of reducing error within the algorithm by calculating the cost or error for each cycle of data moving through the network and updating the weights for each input based upon the derived error from the previous data cycle. This method is explored in the paper "Theory of the Backpropagation Neural Network" (R. Hecht-Nielsen, 1989). Using this method it is possible to reduce the cost function or error to effectively zero for any given data set.

The method for reducing the error within the network is defined by the following function with E being the combined network error.

$$\nabla E = (\partial E/\partial w_1, \partial E/\partial w_2, \ldots, \partial E/\partial w_n).$$

With each weight in the network being updated incrementally.

$$\Delta w_i = -\gamma \partial E / \partial w_i for i = 1, ..., n,$$

(R. Rojas, 1996)

Convolutional Neural Networks or CNN's are a variation of feedforward neural networks that have seen usage mostly in analysing visual imagery though its usage does spread to applications such as language processing and even automated game AI agents. The main distinguishable feature from typical multilayer perceptron feedforward models is that the network is built up of layers of neurons in a two or three dimensional arrangement of filters as opposed to the two dimensional arrangement in typical feed forward networks. Within the network each neuron is only connected to a small selection of inputs and other neurons instead of being connected to each neuron in the other layers of the system.

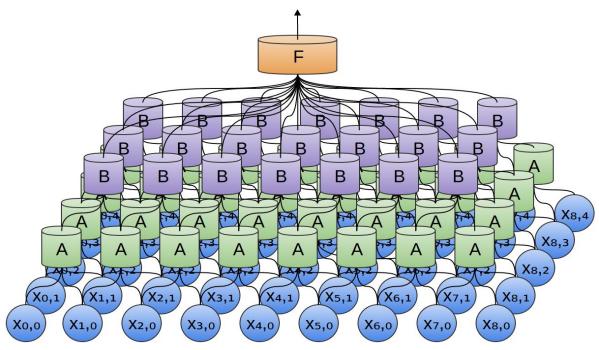


Fig. 3. An example convolutional network layout showing neurons with shared weights. C. Olah, 2014

Figure 3 shows a typical convolutional neural network layout. The network is structured with the inputs (indicated in the figure as the blue x's) being processed by only a small selection of the overall number of neurons for each layer. These neurons (sometimes referred to as filters) have shared weights within each layer. This means that each neuron in layer A will share weights with every other neuron in layer A and so forth. These outputs are then joined in the convolution layer (indicated with the orange F in the above figure) before finally being returned as an output.

While seemingly complex, CNN's can be simplified by thinking of them as networks built using layers of identical neurons with shared weights each looking at different aspects of the input data (C. Olah, 2014). This is particularly useful when dealing with large data sets such as images and video.

Although Convolutional Neural networks are primarily used for tasks such as image processing and recognition as shown in various papers such as ImageNet Classification with Deep Convolutional Neural Networks (A. Krizhevsky et al., 2017) and Face Recognition: A Convolutional Neural-Network Approach (S. Lawrence, 1997) there has been some study into its usage within time series forecasting. Wang and Yan discuss the usage of fully convolutional networks for classifying time series data in a comparison with MLP and residual variant models. The study shows promising findings regarding the accuracy of fully convolutional networks with the highest accuracy showing across 18 out of the 28 datasets (Z. Wang, W. Yan, 2016). Borovykh presents a study on the usage of deep convolutional WaveNet architecture for forecasting time series data and while the study concludes that the convolutional model is less accurate than a state of the art LSTM model there is however room for improvement on an already reasonable accuracy rating (A. Borovykh, 2017). As with MLP's it is possible to introduce back propagation to reduce the error found within the CNN algorithm in much the same way as described for the previous simple feed forward example with the exception of weights being shared across multiple neurons.

Recurrent neural networks have recently become a popular method of time series forecasting with recent studies such as Speech Recognition With Deep Recurrent Neural Networks (A. Graves et al. 2013) in which the recurrent neural network proved to be a highly accurate method of predicting speech and is described as "state-of-the-art".

At their simplest level the basic structure of a recurrent model is similar to a multi layer perceptron model in with inputs moving through multiple hidden layers to achieve an output based upon the weightings of each neuron within the network. The largest difference between feedforward and recurrent models is that while data for each input can only move one way through a feedforward model from input to output, a recurrent model will take into account the inputs that have came before it and update the neurons weights accordingly giving the network a form of memory (D. Britz 2015). A typical layout of a recurrent model is as shown below.

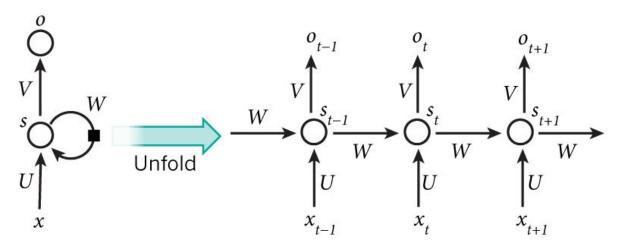


Fig. 4. An example Recurrent neural network. D. Britz, 2015

This memory that is inherent within recurrent network models allows for a more complete overview of the data that is being presented allowing for an output with a potentially higher accuracy than with no context in which to make its decision.

The output of a basic recurrent network is formed of various factors with the input vector 'x' represented as:

 $x(t) = w_t + s(t-1)$ where s is the output of the context layers neurons (T. Mikolov, 2010).

Within the scope of recurrent neural networks there exist multiple variants with differing attributes and advantages to each with the most prominent of these being the Long short-term memory recurrent model (S. Hochreiter, J. Schmidhuber, 1997). LSTM variant recurrent networks were developed as a model that bypasses the vanishing gradient problem (S. Hochreiter et al. 2001). The vanishing gradient problem as outlined in Hochreiter's paper is the problem of error signals during neural network training either vanishing completely or blowing up exponentially when used with backpropagation and gradient based learning systems. This occurs for instance when the gradient used for

training the neural network becomes so small that each iteration makes a very low impact on the neurons weighting that the number of iterations required for meaningful improvements becomes an infeasible length (R. Kapur, 2016).

LSTM models avoid the vanishing gradient problem by introducing what is known as a LSTM unit or block. LSTM models can be made up of multiple blocks and each block comprises of an input gate, an output gate and a forget gate. These gates compute a linear function followed by a logistic function giving an output between 0 and 1 and allows control over the information retained and passed through the block (R. Grosse, 2017). Each gates value at any point will dictate the behaviour off the LSTM block. For instance an input gate value of 1 will pass the input value to the memory unit and a forget gate value of 1 will remember the previous value stored within the memory unit. This behaviour will lead to the values being combined when the forget gate is on or the value being overwritten if the forget gate is off when the input is passed into the block. This structure allows the LSTM variant to effectively avoid a vanishing or exploding gradient as the new hidden state of the neuron has an identity function with a derivative of 1 meaning the gradient will remain constant in place of vanishing or exploding (D. Britz, 2015).

The method of calculating the new hidden state for each neuron/LSTM block is as follows where U and V are input/output vectors respectively, g is the candidate hidden state and c_t is the internal memory of the LSTM unit:

$$i = \grave{o}(x_{t} U^{i} + s_{t-1} W^{i})$$

$$f = \grave{o}(x_{t} U^{f} + s_{t-1} W^{f})$$

$$o = \grave{o}(x_{t} U^{o} + s_{t-1} W^{o})$$

$$g = tanh(x_{t} U^{g} + s_{t-1} W^{g})$$

$$c_{t} = c_{t-1} f + gi$$

$$s_{t} = tanh(c_{t})o$$
(D. Britz, 2015)

2.2 Investigate the factors involved in determining a stock's market share value.

Due to the nature of the world's stock markets determining what factors directly influence a share's value is an incredibly difficult task though some factors have been seen to correlate with increases and decreases in a typical market share price.

Such factors include the earnings per share and P/E ratio (the ratio of price per share over the earnings per share) these two factors are described as fundamental factors for determining prices in an efficient market (D. Harper, 2017). The Price per share is simply the payment a shareholder would receive in dividends per share of stock owned through investment in the company. The P/E ratio is the price for any given share within a company over the earnings per share in the company. This figure then gives an estimation of how much money is required for each pound or dollar return. For instance take a company with a share price of £40 and an earnings per share of £2. The resultant P/E ratio would be as follows:

$$P/E = 40/2 = 20$$

Indicating a required investment of £20 for each pound of return in dividends.

A study on the Bahraini stock market listed 8 separate factors when determining a market share price of a stock. These factors include the return on equity, book value per share, earnings per share, dividend per share, dividend yield, price earnings, debt to total assets and the firm size (T. Sharif et al., 2015). A similar study into the market price of stock in Jordanian taka looked at factors including net asset value per share, dividend percentage, earnings per share, lending interest rate, inflation rate and gross domestic product (F. AL-Shubiri, 2010). Between these two studies there are some similarities in the factors listed though not all listed factors are the same across both studies. This hints at the idea that various stocks will have differing influencing factors based on market location, market type and economical situation with potentially no definite rule to cover all basis.

Another way to determine factors indicating a share prices future value is look at patterns and momentum. When looking at any time series data over a long enough period of time certain patterns are prone to develop that could potentially be used to indicate an upward or downward swing for instance a stock in a company that focuses on summer time equipment or services might see an increase in its stock price leading up to the summer season in anticipation for increased earnings. Momentum is also a potential indicator of a stock's future price and is discussed in the paper Profitability of Momentum Strategies: An Evaluation of Alternative Explanations (N. Jegadeesh, S. Titman, 2011).

For the purposes of this paper the algorithms will look at the past and present market share values only for each stock to forecast future market values due to availability of reliable data and to keep the networks as simple as possible to ease analysis between the variants.

2.3 Investigate the implementation options of using neural networks for time series prediction.

Neural network frameworks with their increase in usage and popularity come in various different options and are designed for the usage of various languages. Some of the most popular open-source frameworks in use currently are Theano, Torch, Caffe and TensorFlow.

Theano is an efficient and flexible library built from the ground up for the Python programming language. Using Theano it is possible to attain speeds rivaling C implementations for problems involving large amounts of data (Theano documentation, 2017). Using Theano a computational graph is defined for the network and the flow of information takes place once the graph has been compiled. This can pose problems as it is harder to debug information during run time. Another issue with Theano is the requirement to specify that the algorithm is to be processed on the GPU when other frameworks automatically compute on the GPU if available. It is also worth pointing out there is currently no support for Theano backend to utilise multiple GPUs when paired with APIs such as Keras.

Torch is a neural network framework with similar performance to Theano. The framework is built for the Lua language and due to this it lacks the standard data processing capabilities of Theano when used with the well known Python language. Recently an effort to create a Python oriented version of torch has seen improved popularity in the Torch library. While still in beta the library has seen usage and improvement from companies such as Nvidia and Facebook (PyTorch documentation, 2017). Due to this still being in early development the level of documentation when compared to frameworks such as TensorFlow or Theano is somewhat lacking. PyTorch does however fix the issues of data processing in original Torch and works in a similar fashion to numpy. For this project one of the more established frameworks will be used though future work implementing the algorithm in PyTorch could be an option.

Caffe is a neural network framework that has seen usage primarily in image processing. The framework is written for C++ though can be used in conjunction with a Python wrapper for easier prototyping (Caffe documentation, 2017). The advantages of using Caffe as a framework is the speed and usability for developing convolutional networks though its usability outside of convolutional networks drops significantly as it is primarily built for using CNNs for classifying images.

TensorFlow is an open-source neural network framework developed by researchers working on the Google Brain team and since its release as an open-source library it has seen increased usage within the community and is used in over 6000 open-source repositories online (TensorFlow Documentation, 2017). The framework has a flexible architecture that allows computation across multiple CPUs or GPUs within desktop, server or mobile devices through a single API. Due to the backing of Google in particular the continued support and development of TensorFlow is assured (Slant Community, 2017). One downside of using TensorFlow is that only some aspects of the framework have been made open source meaning less control than other fully open source frameworks.

Alongside the above frameworks there also stands the Keras Python deep learning library. Keras is a neural network API that is capable of running on top of TensorFlow or Theano with a focus on fast experimentation and prototyping (Keras Documentation, 2017). The main purpose of Keras is to allow for easier and faster prototyping than the standard Theano or TensorFlow libraries while keeping the efficiency and flexibility of each framework. It is due to this increased simplicity that it will be used for the purposes of this paper as an API running on top of the TensorFlow framework.

As mentioned most neural network frameworks are capable of running on both CPUs and GPUs when processing data. With the introduction of the CUDA parallel computing platform developed by Nvidia, processing large datasets using the GPU has become not only an extremely efficient but also simple option beating the performance of CPU units. This is due to the architecture of CPU and GPU modules.

CPU Units are primarily made up of a few cores optimised for sequential processing while GPU units consist of thousands of smaller cores in a parallel architecture designed for running multiple tasks simultaneously (Nvidia Accelerated Computing, 2017). Due to this increased efficiency the neural network models studied within this paper will be developed

and trained on a GPU utilising the Nvidia CUDA technology. This is a process that is made simple within Keras for as long as the machine training the network has CUDA installed in conjunction with an Nvidia CUDA enabled graphics card the TensorFlow backend will automatically enable GPU computing.

2.4 Hypothesis

It is my hypothesis that both the convolutional neural network and the LSTM variant recurrent neural network models will perform with a higher forecasting accuracy than the simple feedforward multilayer perceptron model. It is expected that out of the convolutional model and the recurrent model the recurrent model will have a the best overall accuracy due to its ability to resist the vanishing rule.

3. Methods

This section of the report will focus on the methods that will be used to fulfill the project topics.

3.1 Neural Network development

Development for the project will follow a prototyping strategy wherein an initial prototype for each model used within the study. The application will follow a class based object oriented structure to allow ease of use, readability and efficiency. External data files will be stored by and processed within the application. This will include the time series data in the form of a csv document which will be downloaded from the Yahoo! Finance API using the Pandas Python library (Pandas Documentation, 2017).

3.1.1. Data formatting.

The time series data will be formatted to ensure a uniform structure when it is passed into the algorithm for training. At this point in the application the data will be split into two parts with the first 90% of the data being stored in an array of initial training data and the last 10% of the data being stored in another array for use in testing the forecasting capability of each network during training.

3.1.2. Model creation

Each of the three neural network models will be instantiated and each parameter will be added accordingly. Using the Keras Python library each model will be instantiated using the Sequential() command. This will create a blank sequential neural network model with as an object within the application. Layers will then be added to the blank model to customise it in the way required with options including dense (simple feed forward), LSTM and convolutional layers.

An optional dropout layer will be added to each model to prevent overfitting. Overfitting is a problem inherent in complex neural networks where a model will optimise to perfectly fit the training dataset but when the model is tested against new data the accuracy becomes very

poor as the algorithm has overfitted itself to the training dataset including the noise within data and so does not generalise well (N.Buduma, 2015).

Dropout layers combat the effects of overfitting by removing neurons and their connections from the network during training and effectively training multiple smaller models in varying combinations of neurons from the original model. The results of each trained model are then combined with a new weighting associated with their dropout chance during training to fit the parameters originally set with a higher degree of accuracy. This increase in accuracy is due to the model behaving as a combination of models as model combination nearly always improves performance of machine learning methods (N. Srivastava, 2014).

Once each layer is added to the model object the model is then compiled with an optimiser and a specified loss function (in this instance mean squared error).

3.1.3. Training

After compilation the model is ready to begin training with the first 80% of the imported dataset. This can be done using the fit() function passing in the training data, number of epochs and batch size with each epoch indicating one pass over the entire dataset and batch size indicating the number of data samples being trained at any time in parallel. The number of epochs and batch size will initially be set at an estimated value for the first iteration of training each model before optimizing to determine the optimal parameters for the network. Once training is complete the evaluate function will be called passing in the last 20% of the time series dataset. This in turn will output the mean squared loss of the algorithm based upon the test data passed into the evaluate function. This will be used as a basis to compare future implementations of the model in order to optimise the input parameters to reduce the evaluated loss.

To optimise the algorithm certain criteria will be modified and tested against the MSE output from the initial prototype checking for any improvements in performance and accuracy. To achieve this the algorithm will iterate through the number of epochs and the batch size before returning a MSE loss output for each iteration.

The number of neurons will be kept as low as possible to reduce the chance of homogenisation resulting in the algorithm overfitting the data set.

The number of epochs will need to be fine-tuned through several iterations as accuracy improves up to a point before the algorithm starts to overfit the dataset, an early exit option will likely be placed once it is apparent the MSE is increasing with increased epoch size.

Batch size will be a compromise between accuracy and efficiency as larger batch sizes will give higher accuracy at the cost of drastically increased training times. This increase in training time can become exponentially larger when combined with an increased number of epochs so another early exit will be created to prevent extreme training times.

Once the most optimal MSE loss output is determined by iterating through the number of epochs and batch sizes the last 20% of the time series dataset will be used to further train each model. On completion of training the full dataset each model and its associated weightings will be saved to file for later use in forecasting. This model can then be loaded in

at any point or a new model can be created with the saved weightings by loading the weights from file after declaring a new sequential model.

This process will be repeated for each network model and for each time series dataset with each having its own unique file to ease usage in forecasting.

3.2 Time series forecasting

Once each model has been trained and optimised for accuracy a series of predictions will be made using the algorithm for daily intervals for each time series dataset that is being analysed. Each forecasted price will be recorded along with the net increase or decrease. This will allow the algorithm to be tested on accuracy to exact price and accuracy in predicting increasing and decreasing price.

The forecasting will be calculated using the predict() command for each model passing in the historical dataset and the required interval. This will produce an output in terms of share price for the specified time series dataset.

3.3 Forecast analysis

For each prediction forecasted by the algorithm a record will be kept and compared against future live market prices for each time series dataset. The results will be compared in terms of accuracy against the live price and accuracy in terms of predicting an increasing or decreasing price. This will be measured over each time period forecast and the resulting price accuracy will be calculated in mean square error.

3.4 conclusion and feasibility of each forecasting model

Once all data has been collated and analysed the results will be arranged in a series of graphs and tables to illustrate the accuracy of each model for each time series dataset. Utilising these graphs and tables an overall accuracy will be determined for each model across the range of time series datasets indicating which model gives the higher percentage accuracy.

3.5 Note on randomisation

It is worthwhile pointing out that during this paper the training of each neural network will be in some part random resulting in a varying output performance from the network model. This is due to the input vectors for each model being randomised to increase the network's overall performance as training on identical batches can lead to overfitting the dataset and the algorithm not being generalised. Weights are also initialised at random as initialising each neuron with the same weights will lead to all units in the hidden layer being the same.

4. Design and Implementation

This section of the report details the process undertaken when designing and implementing the algorithm that will be used for the predictions. This process includes the decisions made when creating and training each model and the reasoning behind each decision made.

4.1 Design Specification

In order to allow ease of use when training and testing the models on 5 large datasets an object oriented class based approach was decided upon to facilitate the sorting of the data and handling of each model through encapsulation.

4.1.1 Data Processing

The data being used within the algorithm consists of 5 csv files containing 5 years worth of open and close prices for each day during that time period. Due to the nature of this the algorithm first requires a reliable way to load and process this data to a format compatible with the neural network models used for prediction. In order to achieve this in a practical and reusable way a processing class is required containing functions that will deal with the opening of a desired csv file, loading the desired data into an array and sorting the data for processing in the specified model. This decision was made after looking at various online tutorials and examples.

4.1.2 Model Design

To facilitate the usage of three differing models within the algorithm to train and test 5 different data sets a class structure was chosen to be the optimal approach to reduce complexity during implementation. This would consist of three separate classes, one for each model. These model classes would need a way to initialise and build the model given a set of parameters along with separate functions for training and testing the model. An option to load and save the models would also be required in order to prevent the need for recreating and retraining each model when testing as the training process can take a substantial amount of time. The basis for these models were designed due to the influence of certain tutorials and examples with expansion and modifications made to ensure the models fitted the scope of the investigation.

4.1.3 Training Script

Alongside the model classes the algorithm requires a way to join the separate scripts together in order to train each of the models. This script will need to consist of a way to create 15 models (one of each type for each data set), import and process all 5 datasets, load and train each of the 15 models on the appropriate dataset and produce detailed graphs and statistics for the training process.

4.1.4 Testing Script

Similar to the training script, a script will be needed to facilitate testing of the neural network models across the 5 datasets used within the algorithm. This script will follow the same rough layout as the training script in that it should be able to load the datasets and the trained models when requested and output the results to an acceptable format to assist in the evaluation of the algorithm.

4.2 Implementation

With the design and flow of the algorithm established the next stage of development is the implementation stage. This was achieved using the class based structure previously outlined and unit testing regularly to ensure correct functionality. The development of both the processing class and the network model classes took inspiration from a tutorial found on the online website "Medium". The tutorial can be found at the following link: https://medium.com/machine-learning-world/neural-networks-for-algorithmic-trading-part-one -simple-time-series-forecasting-f992daa1045a

4.2.1 Data Processing

As detailed above the first task in the implementation of the prediction algorithm is the ensure the data is accurately loaded, stored and split for testing and training. These functions take place within their own python class script called "process.py". The class requires usage of the numpy and sklearn libraries to achieve the desired results and in this script 4 functions handle the loading and processing of the data for further use. These consist of load close, split, shuffle and training data. The load close function simply opens the targeted csy file and reads all lines from 1 till the end of the file storing them in an array called stockFile. The function then iterates through each line of the csv file stored within the array storing the close price and close date as variables and appending them to their respective arrays. This is accomplished by splitting the lines into smaller chunks through the commas in the file and assigning the data at the expected location to the appropriate variable. For instance the close price is stored within the csv file as the 5th element in each line and therefor when split will appear as the element [4] in the array of that line (with the first element starting at element [0]). Once each line has been read with the appropriate data stored the function then returns the close prices and the associated dates as arrays. Once completed the function was unit tested passing in each of the downloaded csv files to ensure accurate results.

The split function takes in a data array and attempts to split it into two arrays containing training data in one array and teaching data in the other. This is achieved by creating two arrays within the function and by iterating through the data array in amounts of the desired timestep. The training array is then appended with the value from which to predict and the teaching array is appended with the correct value of the stock for the predicted point in the future. Within this function there is also the availability to scale the array using the preprocessing sklearn library. This removes the trend from the dataset ensuring the model

focuses only on the relative delta movements in the stock as opposed to any long standing upwards or downwards trends.

The shuffle function works simply to shuffle the values of each array passed in to ensure the model does not overfit the dataset by reducing variance and making the model more generalised. This is achieved by randomising the order of elements within the arrays using a permutation of the array length.

The training_data function is used to create the arrays of data that will train and evaluate the training of each model. It takes in two arrays and a floating point value for a percentage and splits the arrays by this percentage. The goal of this is to get accurate results of training by training on the first 90% of the array and evaluating the model on the last 10% before moving on to full predictions.

4.2.2 Model Classes

With the data processing class completed and tested thoroughly the next stage of implementation is the creation of each neural network model class in order to instantiate and manipulate each model during training and testing. Each model used within the prediction algorithm is separated into its own class with each class containing functions for building the model, training the model, evaluating the model and both saving and loading the models associated weights.

Each model is built using the keras api on top of the tensorflow neural network framework. All three models contain the same functions and are largely built using the same structure with variations in certain aspects of the model.

Starting with the Multilayer Perceptron model the class is initialised with an input shape of [20,1] and an output size of 1 before proceeding to the buildModel function. This buildModel function facilitates the generation of the models layers and the compilation of the model. The model itself is built using the Sequential command to initialise a standard blank sequential model. Once created a series of layers and activation functions are added to the model. The model consists of a dense input layer with an input array dimension of 20 and an output size of 500, a dense hidden layer with input size of 500 and output size of 250 and a dense output layer with input size of 250 and output size of 1 where dense layers are simple feedforward fully interconnected layers. The reasoning behind the large input size of the hidden and output layer is due to a shallow networks ability for universal approximation (see previous literature review) when the breadth of the layer is sufficiently large. All three layers within the network use linear activation functions with the input and hidden layer being rectified. The reasoning behind this is two-fold. Linear activation functions do not suffer as greatly from the vanishing gradient problem previously discussed in the literature review section. Another beneficial aspect of the rectified linear (relu) activation function is that it is computationally inexpensive when compared to non linear activation functions such as sigmoid or tanh as only half of the neurons will be firing at once drastically reducing training and running time. This can be seen in the below figures comparing a typical tanh function with a relu function.

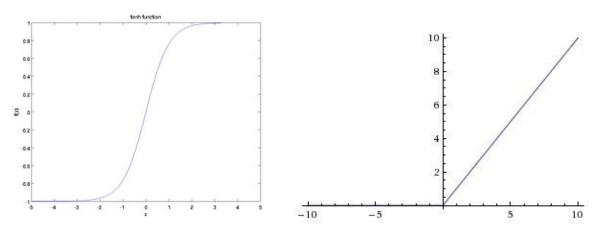


Fig.5 & Fig.6 indicating typical tanh and relu activation functions (L & R respectively) (A. Sharma, 2017)

The last layer is fully linear to ensure all neurons are firing when producing the output value for the network. Between the input and hidden layers a dropout layer is added to prevent overfitting. This is achieved by once again cutting the number of active neurons within any given pass of the system. The dropout layer in the MLP model is set to 0.25 indicating that 25% of the neurons in the hidden layer will be switched off for any given epoch of training. This helps the network become more generalised as the network cannot count on any given neuron being active forcing it to adapt and generalise better.

The model is then compiled using the Adaptive Moment Estimation (Adam) optimiser and finally returned as an object. Some benefits of the Adam optimiser is the optimisers resistance to vanishing gradients and the decaying average of past gradients working as a momentum like characteristic. Due to these characteristics Adam has been known to work well in practice and compares favourably to other adaptive learning-method algorithms (S. Ruder, 2016).

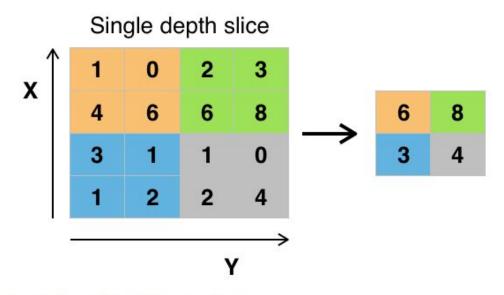
The trainModel Function operates to train the model to be able to predict a stock by showing it previous time series data for that stock. It takes in up to 4 variables with the first two mandatory variables consisting of the training data and the teaching data. The next two variables are for the number of epochs and the batch size and are set to default values within the model, this allows these variables to be optional when calling the function overriding the set values when provided while otherwise using the preset values. The function then uses these variables to train the model using the fit() command passing in the training data, the teaching data and the number of batches. Verbose is set to 1 to produce console output during training and the validation split is set to 0.1 to test accuracy during training. The validation split functions by taking, in this instance, the last 10% of the batch to test the training from the first 90% to determine the error.

The test Function within the class is utilised to evaluate the model outside of training. Similarly to the training function it takes in two arrays: an array of values to predict from and an array of real results. Evaluate is then called on the model passing the two arrays in as

variables with a batch size returning a mean squared error score regarding the models performance.

The last two functions within the class are the load and save functions. These functions when called load and save the current models weights to and from a file allowing a model to be reused between sessions removing the need to retrain the model each run of the algorithm.

The Convolutional model is set up largely the same way with an identical class layout. The main difference between the two models is found in the buildModel function. Within the buildModel function the model is again created using the Sequential command and adding the appropriate layers to the blank model. The layers are built up in this model with two convolutional layers followed by two dense layers. The two convolution layers take in an array of shape [20,1] across 64 filters with each filter having a length of 2 specified with kernel size. Padding is also set to be valid allowing any empty space in the filter to be padded with 0's. The convolutional layers strides are set to 1 to ensure the layer looks at each element couple within the array ie [0,1][1,2][2,3] as opposed to a stride of 2 with [0,1][2,3][4,5]. This is important to ensure the model has the highest accuracy possible by looking at every available element. Max pooling is also set for each convolutional layer with a pool size of 2. This effectively down samples the output storing only the highest result from each section looked at by the filter. An example of this process can be seen below.



Example of Maxpool with a 2x2 filter and a stride of 2

Fig. 7 Example of max pooling (A. Deshpande, 2016)

Figure 7 shows the process of applying max pooling using a 2x2 filter and a stride of 2 (as opposed to the stride of 1 used within this stock prediction algorithm). The filter is applied to each block of 4 elements on the left and takes the highest output from each storing them in a smaller array seen on the right.

Below these convolutional layers two dropout layers are added once again to improve generalisation for the network along with a call to flatten the output from the two convolutional layers before being fed to the dense interconnected layers. This is to allow the network to produce an appropriately sized output as opposed to the 2x2 pooled vector produced by the convolutional layers. The activation functions used for each layer are once again set to rectified linear and linear for the same reasons as previously.

Another difference between the MLP and Convolutional models is the input array dimensions. MLP models with dense layers are able to accept input with shape dimensions of 2 whereas convolutional layers require in an input shape with 3 dimensions. In order to adhere to this during training and testing of the model the training and teaching/evaluation arrays have their dimensions expanded on the axis value of 2 to ensure the arrays have an input shape dimension of 3. For instance given an array of shape (2,2) expanding the dimensions on axis 0 will result in a shape of (1,2,2), expanding the dimensions on axis 1 will result in a shape of (2,1,2) and so on.

Lastly the the LSTM model once again follows the same basic structure as the two previous models when it comes to its class structure. Much like the Convolutional model the LSTM variant recurrent model requires an input shape with 3 dimensions when training and testing therefore to accommodate this the dimensions of the training and testing arrays have been expanded similarly to the convolutional model on the 2nd axis of each array. Along with this the models layers once again differ from the previously described models. The LSTM model consists of 4 layers with 3 LSTM layers followed by a dense output layer. The first two LSTM layers are identical outputting vector sequences of dimension size 32 (these layers return sequences of vectors instead of singular vectors due to the return_sequences flag being set to true). The third LSTM layer receives a series of input vectors of dimension size 32 and returns a singular vector to the fully connected dense layer resulting in a singular output. Activation is once again linear for the dense layer with each LSTM not requiring a specified activation function due to the makeup of the LSTM blocks (see literature review). Once the layers have been added the model is compiled once again using the Adam optimiser before the model is returned as an object.

4.2.3 Training Script

With the model classes and data processing class completed the next stage of implementation is to combine the functionality within these scripts to train a set of 15 models (5 of each variation, one for each dataset). In order to achieve this each of the previous classes required imported along with matlplotlib pylab to produce graphical outputs for the training process. Once imported a series of lists are instantiated holding names for the various neural network models and the stock csv file names along with blank lists to hold the time series data and dates. 15 models are then created consecutively and stored within three lists (one list for each model type) and the training variables are set (training array dimension size, prediction timestep in multiples of days and the lag/time step between predictions). When each of the initial global variables have been initialised the time series list is populated by iterating through each of the csv files stored within the stock list and using

the process scripts load_close method appending the time series list with the result. Following this Training commences on each of the time series datasets.

Iterating through the time series list three empty local lists are created to hold the score of each model after training has been evaluated. The data sets are then processed and split into appropriate scaled training and test arrays. These training arrays are then used to train each model calling the trainModel function on each model within the three model lists with the score of each evaluation following and being appended to the relevant score lists. These score lists are then drawn to a graph for each dataset using pylabs's plot command with the graph being saved to file for reference.

4.2.4 Testing Script

The testing script is structured in a similar fashion to the training script in that it contains the same import requirements and setup variables. The testing script also requires the pandas package in order to output results to a csv file for interpretation. As stated the starting variables and lists are nearly identical to the training script however once each model is loaded in the testing script it is loaded from file using the models load function passing in the trained models file name. This is to prevent the need for retraining each model every time the testing script is ran. Two new lists are also instantiated with a list of csv file names in which to save each csv file and and array of arrays in which to store each of the 5 sets of results during testing.

Once the initial variables and lists are set up the stock data is loaded in an identical fashion to the training script wherein the time series list is populated by iterating through each of the 5 stock files and calling the load close function.

Iterating through this list of stock data the data is once again processed and split into appropriate training and testing data while also generating an unscaled version of each in which to create complete results by scaling each models output to match the original unprocessed format. In order to do this an array of parameters is instantiated and for each time step in the unscaled testing the mean and standard deviation of the time step is recorded in the parameters array to allow for the scaled neural network outputs to be adjusted bringing them in line with real results. The scaled test data is then fed into each model using the predict command generating a result on the corresponding timestep. These values are then paired with their corresponding elements in the parameter array to adjust the stock price values to real values by multiplying each prediction by the standard deviation and adding the mean.

Following this an array is made up of the actual stock value at the predicted date along with each models adjusted prediction for that date. This array is then stored as a pandas dataframe and saved to a csv file. Along with the csv output a series of graphical representations of the predictions are also created against the actual stock value with both scaled and unscaled representations and saved to file.

4.3 Model Training

For the training process there are many variables that can affect the accuracy of the models prediction and each one requires fine tuned to determine the optimal settings for any given task. Two of the main variables when training a network are the number of epochs of the system and the batch size of the training samples. The number of epochs is how many times the training is ran over any set of batches, too few and the model won't be able to build an accurate enough picture of the time series and the predictions will be inaccurate, conversely if the number of epochs is too high the risk of overfitting the dataset increases and the model could suffer in performance when predicting previously unseen data points. The batch size variable also suffers a similar problem, too short a batch size and the model will have a less accurate picture of the gradient whereas too large a batch size can once again lead to the model overfitting the problem and becoming less generalised.

In order to determine the optimal number of epochs and correct batch size each of these variables were tested in iterative loops with the accuracy of each network being tested regularly to determine the best cut off point for each. Firstly each network was trained with a default batch size value of 128 elements per sample (this number was determined through previous experience in developing neural networks and through various online examples) and a variable epoch count starting at 10, increasing by 10 each iteration and ending at 200. This training produced the following results in accuracy.

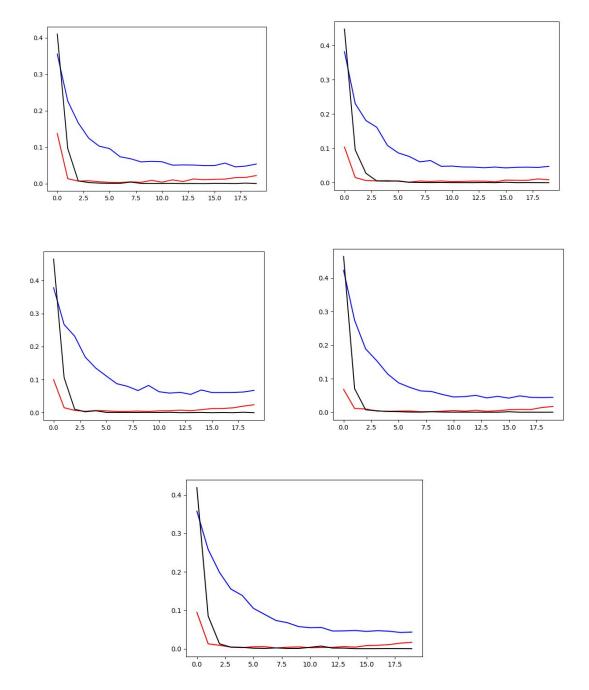


Fig. 8 Epoch training graphical representations

The 5 graphs above feature each of the models accuracy in mean squared error (left hand axis) against the number of iterations (bottom axis, units are in multiples of 10 starting at 10). The graphs show the differences in each models accuracy after training for so many epochs with the multilayer perceptron model being shown in black, the convolutional model shown in blue and the LSTM variant recurrent network depicted in red for each graph. As can be seen in the graphs starting at the first iteration of training with 10 epochs the MLP network has the lowest accuracy with a MSE greater than 0.4 with the convolutional model having slightly better accuracy averaging a MSE just short of 0.4. The LSTM has a remarkably low starting MSE value averaging at 0.1 showing that at least for low exposure systems LSTM would

likely produce significantly better results than the other networks models. This high base accuracy is due to the additional information of sequence within the LSTM model giving the model effectively more information to base its decision than the opposing network models. Iterating through the number of epochs during training an improvement can be seen in the accuracy of all three models to a certain point wherein the accuracy either stagnates or diminishes. The point where this happens for each network is seen to be the optimal number of epochs for training to gain accurate results without overfitting the problem. Both the MLP model and the LSTM model reach this value quickly with an optimal epoch value for each being determined at 60 epochs. The Convolutional network however takes significantly longer to plateau due to the complex structure of the network and its shared weightings between neurons. Due to this the optimal epoch count for the Convolutional model was taken at 150 epochs opposed to the 60 epochs of the other two models.

Following on from determining the optimal number of epochs for training the same process was applied to the batch size during training. This time the network was trained with a variable batch size starting at 16, progressing through iterations of 16 up till a maximum batch size of 160 with each network being trained through its previously determined optimal number of epochs. The following graphs were created as a result of this process.

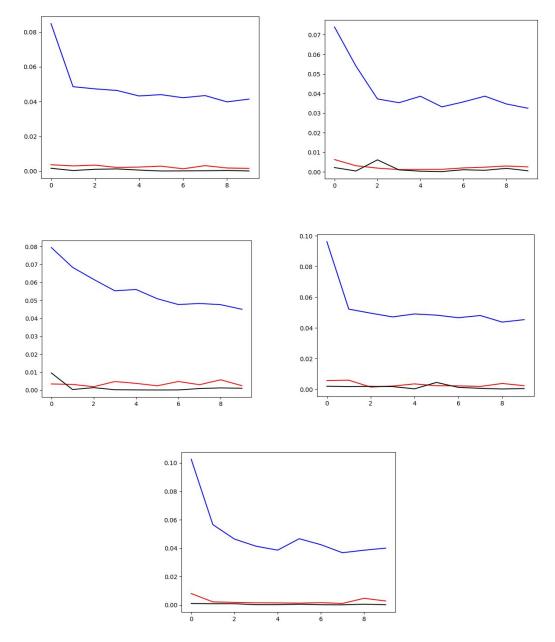


Fig. 9 Batch training graphical Representations

These graphs with are once again laid out in the same format with the MSE on the left hand side and the number of batches on the bottom axis (units are in multiples of 16 starting at 16 and ending at 160). Once again the Convolutional network model has a high starting MSE with low batch size and the error drops with increased batch size. This is due to the nature of the convolutional network and the way it analyses data. Due to the filters within the convolutional network looking over an array for certain characteristics a smaller batch size makes it harder for the network to notice any trends or markers that a stock price is going to shift one way or another. The LSTM and MLP models remain fairly consistent with their MSE value throughout the range of batches trained so the original value of 128 was kept for all three models as this coincided with the lower error values found when changing the batch size for the convolutional network.

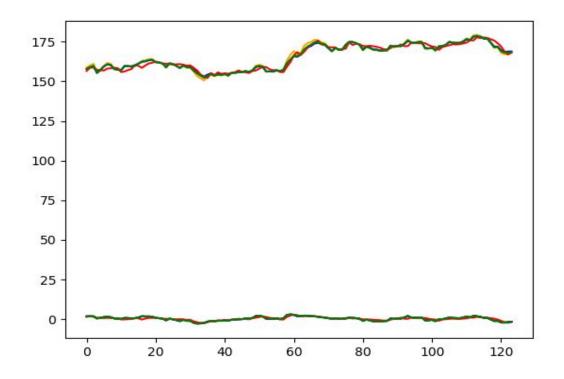
Once each of these values had been optimised all 15 models were trained once more and saved to file for usage in testing the models on unseen data.

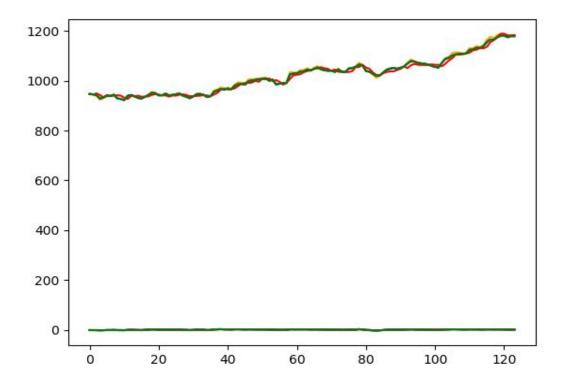
5.0 Testing and Evaluation

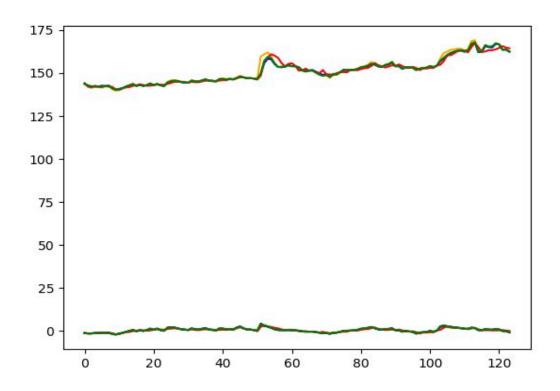
This section of the report covers the testing environment and success criteria for each model. The suitability of each model will then be evaluated and compared against the others to answer the research question and an explanation for each models performance will be given. This section of the report will also discuss the possibility for future work in this field and outline the possible direction for such a study.

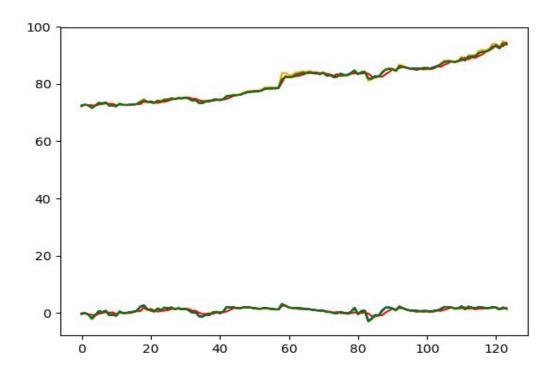
5.1 Model Testing and Evaluation

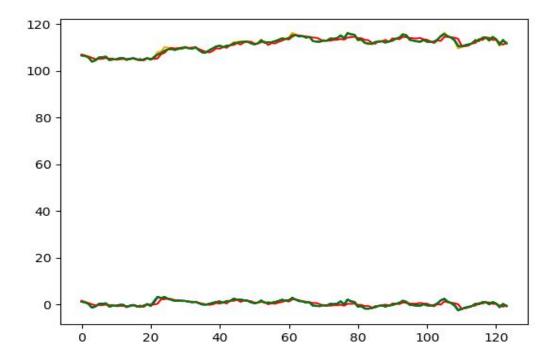
Testing of each model was completed by taking the last 10% of each time series dataset stored within the test arrays generated from the training_data function and calling each models predict command passing in this test array. Due to the models being trained and evaluated so far only on the first 90% of the data sets this last 10% is entirely new to the model and as such cannot be predicted due to familiarity. The outputs of each predict command are stored within several lists with scaled versions being created for comparison against real stock market values. These values are all then plotted against each other for each dataset resulting in the following graphical representations.











Figs 10-14. Graphical outputs of predicted stock values against live values

Although difficult to see, each network model is colour coded on the graphs with stock values on the left hand side against time series steps on the bottom axis. The topmost lines on each graph represent the live prices and the restored predictions whereas the

bottommost lines represent the scaled price and the network prediction outcomes. The live price of the stock is indicated in orange, the scaled live price indicated in black, the MLP models predictions (scaled and unscaled) are represented in blue, the convolutional models predictions depicted in red and the LSTM models outputs shown in green.

As can be seen from each of the graphs each models output predicts the changes in share price remarkably well and appears to be highly accurate when comparing curvature alone. It is worth pointing out however that although a series of outputs are given and each output is a prediction they are not sequences of compounded predictions. Each prediction made for the next time step this test is done using an actual result for the current time step i.e. the model is predicting one time step ahead at each iteration.

The results of each set of predictions is also stored as a set of values in csv file format for further analysis as graphical representations alone are difficult to make accurate conclusions from especially as the predictions are very similar.

These csv result files were then collated and analysed generating percentage and mean squared error figures for each dataset which can be found below.

TOTAL MSE	
MLP	18.04993984
CONV	117.2847921
LSTM	16.13550099

MSE Per Stock	Stock1	Stock2	Stock3	Stock4	Stock5
MLP	0.8853748371	15.32925425	1.584690924	0.1556430585	0.09497677083
CONV	3.80296345	108.4250572	3.381935231	0.6584614628	1.016374687
LSTM	0.4907045028	14.23798449	1.177279967	0.1531673655	0.07636466329

Average % Error Per Stock	Stock1	Stock2	Stock3	Stock4	Stock5
MLP	0.3712951143	0.276656032	0.3550128971	0.3048075758	0.1805927642
CONV	0.955299936	0.8191997291	0.764162989	0.732727986	0.6955296373
LSTM	0.2424281584	0.2692191742	0.2474589093	0.3074090796	0.1374447639

Fig. 15-17 Error tables for testing on all 5 datasets

From the above figures it is much easier to compare the results of each network models prediction accuracy. The top figure shows the combined MSE for each network across all 5 datasets with the middle figure showing the MSE for each network model for each dataset. In each table for each set of results the prediction accuracy is pretty consistent between each model with the LSTM recurrent model performing with the lowest MSE with MLP slightly behind and the Convolutional model falling far behind. The lower table shows the average percentage error of each prediction compared against the live price and once again consistently shows the LSTM network model ahead of the other two and the Convolutional model bringing up the rear in terms of its error percentage. It is worth noting however that although the Convolutional Neural Network is producing a significantly higher error score in each scenario that error score is still remarkably low considering the application. With the Convolutional model providing the worst accuracy levels across the testing at 0.955% and total MSE across effectively 10 percent of 25 years of data (roughly 913 samples) it becomes apparent that all three models perform exceptionally well in single step time series prediction for usage in stock market share price prediction.

The LSTM network model as expected performs the best in almost every scenario through testing when compared with the other two models. This is due to the sequential nature of the model and its ability to make predictions based upon previous observations and outputs along with the current state giving it a slight advantage over non recurrent models. This increase in performance is in line with other studies comparing the effectiveness of LSTM networks against MLP networks such as a study by Zachary Lipton et al in its usage for clinical diagnosis (Z. Lipton et al, 2017) and a comparison of recurrent and MLP networks for determining harmonic contributions from non-linear loads (J. Dai et al, 2008).

The MLP model performs exceptionally well with error scores slightly behind the recurrent LSTM model. This is higher than originally expected when conducting this study and is likely due to the large breadth of the hidden layer allowing for very detailed analysis of the dataset as seen in previous studies such as "Artificial Neural Network: Deep or Broad? An Empirical Study" wherein the breadth of the network not the depth of the network leads to a lower bias in the model (N.Liu, 2016). The usage of rectified linear activation functions also reducing the chance of overfitting the problem while training allowed the model to better generalise the fundamental approximation function.

Although the Convolutional model performed to a high degree of accuracy it was not able to match or beat the performance of the other two models used within this study. This is likely due to the difference in approach to detecting increases or decreases in price as opposed to the other two models. Due to the filters used within the network each prediction is based upon certain markers within the data that might indicate an upward or downward trend as opposed to making a prediction based entirely off only the current state like in the MLP model or based off state and momentum used within the LSTM model. Similar studies have been done indicating high performance with Convolutional models due to sudden changes in the datasets (S. Selvin, 2017). This ability to predict off certain markers can be beneficial for identifying anomalous spikes or dips indicating a more drastic change (this can be seen at multiple points through the prediction outputs where the convolutional model drastically

outperformed the other models) but can be detrimental when such clear markers are not apparent thus producing the lower overall accuracy.

5.2 Conclusion

Based upon the results of the study it would have to be concluded that the Long Short Term Memory variant Recurrent Network model does indeed perform with higher accuracy than the standard feed forward Multilayer Perceptron model. However contrary to the previous hypothesis the Convolutional Neural Network model on average performs with a lower accuracy than the standard feed forward Multilayer Perceptron model though could potentially be more useful in a more volatile stock or market with more distinct markers indicating an upward or downward trend.

5.3 Future work

Based upon the findings of this study there is scope for further work in the effects of combining different model architectures such as the Convolutional network model and the LSTM recurrent model. This could potentially increase effectiveness giving the model an accurate picture of the overall trend through its ability to track momentum while having the benefit of being able to distinguish certain anomalous markers. As mentioned previously the outputs of each network were predicted for one time step ahead at daily intervals (some testing was done on single time step prediction at weekly intervals with similar results). Another focus of study could be to attempt prediction for multiple timesteps. This could be interesting to see the long term predictive effects of the network though risks compounding errors.

Another option is to introduce other variables to the network to attempt to improve accuracy though this also drastically increases the complexity of the problem and will likely vastly increase training costs.

6.0 References

- W. McCulloch and W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity" http://www.cse.chalmers.se/~coquand/AUTOMATA/mcp.pdf [Online][Last Accessed 15/11/2017]
- M. Hu and H. Root (1964). "An adaptive data processing system for weather forecasting". Journal of applied meteorology, vol. 3, issue 5, pp.513-523.
- T. Kimmoto, K. Asakawa, M. Yoda, M. Takeoka (1990). "Stock market prediction system with modular neural networks".
- https://pdfs.semanticscholar.org/25b0/d0316ece493899d74cfb98ce7b77dca8352e.pdf [Online][Last Accessed 15/11/2017]
- G.Zhang, E. Patuwo and M. Hu (1998). "Forecasting with artificial neural networks". International journal of forecasting, vol. 14, issue 1, pp.35-62.
- H. White and A.Refenes (1988). "Economic prediction using neural networks: the case of IBM daily stock returns".
- https://pdfs.semanticscholar.org/a059/ffef2efc6f7f2efd68a5bbf313dcd3072c42.pdf [Online][Last Accessed 15/11/2017]
- C. De Groot and D. Wurtz (1991). "Analysis of univariate time series with connectionist nets: a case study of two classical examples". Neurocomputing, vol.3, issue 4, pp.177-192.
- K. Chakrabotry, K. Mehrotra, C. Mohan and S. Ranka (1992). "Forecasting the behaviour of multivariate time series using neural networks" techreports [Online][Last Accessed 15/11/2017]
- I. Poli and R. Jones (1994). "A neural net model for prediction", Journal of the american statistical association, vol. 89, issue 425, pp.117-121.
- A. Borovykh, S. Bohte, W and C. Oosterlee (2017), Conditional time series forecasting with convolutional neural networks" https://arxiv.org/abs/1703.04691 [Online][Last Accessed 15/11/2017]
- J. Connor and D. Martin (1994), "Recurrent neural networks and robust time series prediction" http://www.macs.hw.ac.uk/~dwcorne/RSR/00279188.pdf [Online][Last Accessed 15/11/2017]
- G. Zhang (2007), "A neural network ensemble method with jittered training data for time series forecasting" http://www.sciencedirect.com/science/article/pii/S0020025507003003 [Online][Last Accessed 15/11/2017]

- T. Taskaya and M. Casey (2005), A comparative study of autoregressive neural network hybrids." Neural Networks, vol.18, issue 5, pp.781-789.
- A. Bryson and Y. Ho (1979), "Applied Optimal Control: Optimization, Estimation, and Control", IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, issue 6, pp.366-367.
- S. Seth (2015) "The world of high frequency algorithmic trading", https://www.investopedia.com/articles/investing/091615/world-high-frequency-algorithmic-trading.asp [Online][Last Accessed 15/11/2017]
- G. Box and G. Jenkins (1970). "Time Series Analysis: Forecasting and Control" https://s3.amazonaws.com/academia.edu.documents/30152084/176871241.pdf?AWSAccesskeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1510925340&Signature=BFIPSkVqO20QiiH9ch42wtnk34g%3D&response-content-disposition=inline%3B%20filename%3DTime_series_analysis_forecasting_and_con.pdf [Online][Last Accessed 15/11/2017]
- R.Nau, 2017, "Statistical forecasting: notes on regression and time series analysis" https://people.duke.edu/~rnau/411home.htm [Online][Last Accessed 15/11/2017]
- M. Khashei and M. Bijari (2010), "An artificial neural network(p,d,q) model for timeseries forecasting", Expert Systems with Applications, vol. 37, issue 1, pp.479–489.
- C. Stergiou and D. Siganos (2017) "Neural networks", https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html [Online][Last Accessed 15/11/2017]
- A. Yilmaz, C. Aci and K. Aydin (2015), "MFFNN and GRNN Models for Prediction of Energy Equivalent Speed Values of Involvements in Traffic Accidents" http://ijaet.academicpaper.org/article/view/1072000110 [Online][Last Accessed 16/11/2017]
- R. Hecht-Nielsen (1989), "Theory of the Backpropagation Neural Network", https://pdfs.semanticscholar.org/4d3f/050801bd76ef10855ce115c31b301a83b405.pdf [Online][Last Accessed 16/11/2017]
- C. Olah (2014), "Conv nets: a modular perspective" http://colah.github.io/posts/2014-07-Conv-Nets-Modular/ [Online][Last Accessed 16/11/2017]
- A. Krizhevsky, I. Sutskever and G. Hinton (2017), ImageNet Classification with Deep Convolutional Neural Networks http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf [Online][Last Accessed 16/11/2017]
- S. Lawrence, C. Giles, A. Tsoi and A. Black (1997), "Face Recognition: A Convolutional Neural-Network Approach",

http://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Fall.2016/pdfs/Lawrence_et_al.pdf [Online][Last Accessed 16/11/2017]

- R. Rojas (1996), "The backpropagation Algorithm" https://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf [Online][Last Accessed 16/11/2017]
- D. Britz (2015), "Recurrent neural networks tutorial" http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/ [Online][Last Accessed 16/11/2017]
- T. Mikolov (2010), "Recurrent neural network based language model" http://www.fit.vutbr.cz/research/groups/speech/servite/2010/rnnlm_mikolov.pdf [Online][Last Accessed 16/11/2017]
- S. Hochreiter, Y. Bengio, P. Frasconi and J. Schmidhuber (2001), "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies" http://www.bioinf.jku.at/publications/older/ch7.pdf [Online][Last Accessed 16/11/2017]
- R. Kapur (2016), "The vanishing gradient problem" https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b [Online][Last Accessed 16/11/2017]
- D. Harper (2017), "Forces that move stock prices" https://www.investopedia.com/articles/basics/04/100804.asp [Online][Last Accessed 16/11/2017]
- T. Sharif, H. Purohit and R. Pillai (2015) "Analysis of Factors Affecting Share Prices: The Case of Bahrain Stock Exchange" http://ccsenet.org/journal/index.php/ijef/article/view/45844 [Online][Last Accessed 16/11/2017]
- N. Jegadeesh and S. Titman (2011), "Momentum" https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1919226 [Online][Last Accessed 16/11/2017]

Theano documentation (2017) http://deeplearning.net/software/theano/introduction.html [Online][Last Accessed 16/11/2017]

Pytorch (2017) http://pytorch.org/ [Online][Last Accessed 16/11/2017]

Caffe documentation (2017) http://caffe.berkeleyvision.org/ [Online][Last Accessed 16/11/2017]

Tensorflow Documentation (2017) https://www.tensorflow.org/ [Online][Last Accessed 16/11/2017]

Keras Documentation (2017) https://keras.io/ [Online][Last Accessed 16/11/2017]

Nvidia Accelerated Computing (2017) http://www.nvidia.com/object/what-is-gpu-computing.html [Online][Last Accessed 16/11/2017]

Pandas Documentation (2017) http://pandas.pydata.org/pandas-docs/stable/ [Online][Last Accessed 16/11/2017]

N.Buduma (2015), "Preventing overfitting in neural networks" https://www.kdnuggets.com/2015/04/preventing-overfitting-neural-networks.html [Online][Last Accessed 16/11/2017]

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov (2014), "Dropout: a simple way to prevent neural networks from overfitting" https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf [Online][Last Accessed 16/11/2017]

Slant Community (2017) https://www.slant.co/options/14689/~tensorflow-review [Online][Last Accessed 16/11/2017]

- G. Cybenko (1989) "Approximation by Superpositions of a Sigmoidal Function" https://www.researchgate.net/profile/George_Cybenko/publication/226439292_Approximation_by_superpositions_of_a_sigmoidal_function_Math_Cont_Sig_Syst_MCSS_2303-314/links_cont_Sig_Syst-MCSS-2303-314.pdf [Online][Last Accessed 18/04/2018]
- H. Mhaskar (2016) "Learning Functions: When Is Deep Better Than Shallow" https://arxiv.org/pdf/1603.00988.pdf [Online][Last Accessed 18/04/2018]
- S. Ruder (2016) "An Overview of Gradient Descent Optimization Algorithms" http://ruder.io/optimizing-gradient-descent/index.html#adam [Online][Last Accessed 18/04/2018]
- Z. Lipton et al (2017) "Learning to Diagnose with LSTM Recurrent Neural Networks" https://arxiv.org/pdf/1511.03677.pdf [Online][Last Accessed 18/04/2018]
- J. Dai et al (2008) "A Comparison of MLP, RNN and ESN in Determining Harmonic Contributions from Nonlinear Loads" https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4758443 [Online][Last Accessed 18/04/2018]

Nian Liu and Nayyar Zaidi (2016) "Artificial Neural Network: Deep or Broad? An Empirical Study" https://link.springer.com/chapter/10.1007/978-3-319-50127-7_46 [Online][Last Accessed 18/04/2018]

S. Selvin, V.Gopalakrishnan and V. Menon (2017) "Stock Price Prediction Using LSTM, RNN and CNN-Sliding Window Model"

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8126078 [Online][Last Accessed 18/04/2018]

<u>Appendix</u>

Appendix A - Code

Appendix B - Results

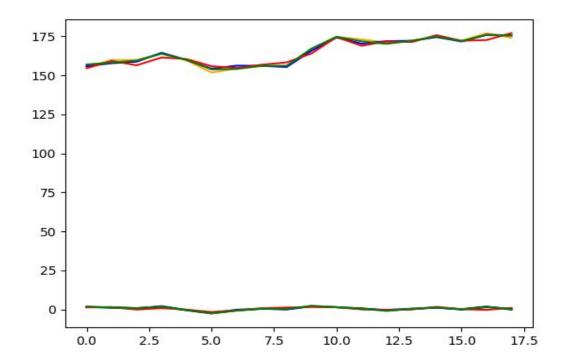
LSTM e*2 % Error MSE Average % Error	CORVOLUTION n*2 % Error MSE Average % Error	Target MLP 4/2 % Enter MBE Average % Error	LSTM e ¹ 2 % Enter MBE Average % Emp	CONVOLUTION e*2 *s Error MSE Avarage % Error	Target MLP 4°2 % Empr ANSE Average % Emp	USTM 6*2 % Empl MSE Average % Empl	COMVOLUTION p*2 % Error MSE Average % Error	Target MLP 0*2 % Error MSE Average % Error	LSTM e*2 % Linor MSE Average % Error	CONVOLUTION e*2 % Enar MSE Avience % Enar	Target MLP a/2 % Error MSE Average % Error	LSTM e*? % Error MSE Avaprage % Error	COMVOLUTION e*2 % timor MSE Average % Error	Target MLP e^2 % Error MSE Average % Error
105.49908 0,0101869854 0,09467196679 0,07036496529	107,02798 0,1031008004 0,4014021764 1,016374087 0,8065296273	0.005.000.000.000.000.000.000.000.000.0	72,3485 0,00255025 0,00975138122 0,1631973605 0,3074090786	72,22048 0,0022274304 0,2479556011 0,6084814628 0,732727986	0 72.4820.36 0.02254820.30 0.169430686	143.63121 0.2127146641 0.3214679027 1.177279667 0.2474589093	144,16348 0,4809140104 0,4833822154 3,281935231 0,784162989	0 143,475 143,7483 0,0774346769 0,1942078483 1,78469024 0,3559128671	046.469 0.469.950 0.07405475811 14.23788446 0.2682191742	047.228 2.784.64 0.158278082 108,4295672 0.8191997291	0 patche patchero 1,23677794 0,1171134021 16,33929428 0,276696022	157,87094 0,6618036836 0,5913168300 0,4907048828 0,2424261564	156,59108 4,923605966 1,3072166 3,80296345 0,955259636	156.81 157.84688 0,027657923 0,0094784408 0,285374837143
105,3288 0,0000944 0,01691729323	108,059746 0,02551078452 0,189137218	106.41998 0.018769099010 0.0187690985	72,76308 0,007241481 0,03608036392	72,76973 0,0001100729 0,02784723176	72,79 72,81398 0,005755901 0,01296792388	142.48918 0.3354494724 0.4075575258	142,10712 0.0000002944 0.002028899113	142,11 142,76264 0,4129061696 0,4622130744	944,8692 0,49131284 0,07193467417	944,4599 0,07070291 0,02816170474	944,78 945,54000 1,82426422 0,143048538	158,94707 1,263630365 0,7077273863	158,62157 2,12701006 0,9110632164	160,08 159,94614 1,290,17794 0,7080577211
105,85303 6,0013657800 6,00497065749	106,10627 0,1712607129 0,3808094001	2 105.59 105.7706 0.00649638 0.07628076281	72,48628 6,0000138384 6,000133158548	72,581024 8,01239632868 6,1531989448	2 72,47 72,560,8 0,0096000241 0,1350,265369	142,97335 0,3539612225 0,355048907	141,53348 0,0559417104 0,1658338007	2 141,77 142,2791 0,25910281 0,3591027721	941.49603 2.005140981 0,1006299899	943,34786 86,46952186 0,9658373755	2 940,08 941,0531 3,1438391 0,186(110075	159,91063 1,321051397 6,7136264614	156,6907 5,59937569 1,469204023	161.06 150.37166 2,00491936 1,046267726
103,93715 0,0161671225 0,1234613631	105,58203 3,140090321 1,756903346	103.81 103.9629 0.033466000 0.1761900003	71,56732 0,0247495824 0,2203082794	72,61098 1,44235296 1,88180927	3 71.41 71.70016 0.6788488246 0.6146702143	142-23985 0,1461229025 0,2665772702	142,67415 0,696262226 9,5660922166	3 141,84 142,25186 0,1687028025 0,2504328821	928,1632 21.09740824 0,497302385	942,15713 344,7531703 2,010364989	3 923,50 923,39405 214621575 0,50017150390	155,2947 0,00124000 0,001277727273	157,27928 3,638778118 1,381447336	155.32 155.3459 0.00072341 0.017311008318
504,44436 0,5002345936 0,01478077733	105,01408 0,308053438 0,5304039234	104.45 104.545395 0,007282333633 0,00174889485	72,4666 0,00001936 0,00001936517	72,39271 0,0110111441 0,147682080	4 72.52834 0.0008031556 0.009038666617	143.15699 0.0985496001 0.2227791978	142,0797 0.05740809 0.1689832318	4 141,94 142,20341 0,132666291 0,2562112239	932,8756 7,76901535 0,2864871454	933,9811/ 15,14431706 BA16407698	830.09 832,7111 6,87016621 0,2816114376	157,34042 0,0194825794 0,08983347727	157,07801 0,1011530021 0,2648631086	157,48 136,06781 0,262281681 0,3251778004
165,70063 0.0000000488 0.000598029480	106,12597 0,3052757408 0,5478051088	5 105,7 106,74584 0,0021013058 0,04336802271	73,50468 0,0072777861 0,1159260768	72,83829 0,5650679241 1,021483807	73.59 73.4122 0.0316128 0.241008914	142.54948 0.2626610704 0.1612422709	141,88143 0,4077716449 0,4496960566	5 142,00 142,82102 0,0906130404 0,2115062749	940,1402 1,48644964 0,1264496358	942.4856 12,64229136 9,378686377	5 939,14746 0.02710040599	159.54663 0.7633013669 0.3140014701	100,04184 9,0400,0566 1,061864348	150.55 150.3730 0.226426446 0.297822850
105,84161 0,50234 15021 0,04569636623	100,184875 0,4972012666 0,6859002969	8 105,89627 0,000,000,000 0,000,000,000,000,000,00	73,149508 0,004965175296 0,09623690109	72,99067 0,0528381246 0,3133433488	73.22 73.271736 0.002679613696 0,07086028008	142,3108 0.08240004 0.1758280168	142,60367 0.2548036589 0.3756387686	6 142,07 142,40453 0.1119103209 0.2354684311	839,59646 0,836959318 0,99798528261	939,3788 1,67909764 0.1381332083	808.08 908.0842 908.0542 0.00270029482	160.83173 0.800287909 0.475414604	158,03856 12,66,365467 2,303861366	161,6 160,6594 0,872460038 0,8780074267
106,09051 0,0003180001 0,07487754033	105,48702 0,4539020804 0,4339239171	7 105,16 106,06078 0,0002759004 0,07462321025	73,584564 0,004251670096 0,08664725051 1	73,24359 0,1651690981 0,5518128273	7 73,65 73,563 6,0036,440 6,007,4160,000	143,61290 0,0127987401 0,07929122807	142.05312 0.1997017344 0.3130	7 142,6 142,60182 0,037591924 0,136140351	944,7056 0,19000881 0,04616284416	22,41359649 0,5313714288	7 944,27 943,097 1,3769040 0,1243008172	150,39288 0,3103826944 0,3481447665	158,21975 7,47985862 1,688198188	180,87 160,1601 0,829768 0,8270768
504.54 128 0,0002763876 0,0005688638	105,21491 0,4447889461 0,6378861789	904.59 504.7905 0.05313020 0.2504690783	72,39799 0.000000000010 0.0007753453094	73.061905 0,438118229 0.9142334254	8 72.4 72.43975 0.001154300625 0.04633619650 (141.04096 0.1194117136 0.2456005690	141.98721 1,905821194 0,9009487962	8 140,7 141,22346 0,274013716 0,372023601	928.64825 0.9789134225 0.1086900005	941,61346 194,691046 1,504159895	827.66 800.0311 5.40211521 0.2556001121	157.58572 0,6787795104 0,1737486014	158,80254 0,8519598436 0,478532356	157.86 157.81512 0.0020142144 0.02843025400
104.798 0,00045796 0,02041595115	105,18767 0,1351612366 0,3607632131	9 104.92675 0.00428875825 0.004246422438	72,49098 0,0000000000 0,001365708615	73,06860 0,3325713601 0,705544213	72.49 75.457 10.0000000	140,29475 0.3537275425 0.4257337151	140,54428 0,4057830225	139,7 140,4160 0,5134561 0,5131710809	927.22208 1.017139878 9.112576369	940,2341 197,5177268 1,517428418	0,000,000,000,000,000,000,000,000,000,	157,24402 0,96503057004 0,1625260641	155.327 0,700569 0,5314285714	157.5 157.60991 0.012002091 0.017412092
104,910504 1,000303512308 0,01817020871	104,73979 0,0361780441 0,1912732298	12296496201 91296401 01	72,17854 0,0028031556 0,00927927938	72,58928 0,1929669184 0,6088426188	72,75 72,75 72,23594 0,0731004016 0,1180080625	140,64435 0.0988150225 0.2240078901	140,12587 0,0416690569 0,1454642628	10 140,03 140,41078 0,2311301778 0,3425824608	922.55023 2,823172863 0,1824611808	932,1887 126,1129697 1,229131148	10 920,07 142,000m 1,418120722 0,1280173276	150,95105 0,0679801005 0,164718985	155,83493 1,890617505 0,8746768225	157.21 157.13972 0.0043392784 0.04478453533
105,37179 0,0054600041 0,00624890274	194,923225 9,2369499006 9,4617920501	105.452415 105.452415 0.001799002220 0.04023811783	73,121544 0.001478863936 0.00288424276	72,97184 0,0530361856 0,3636784352	73.16 73.16 73.04578 0.0130462084 0.1181219940	141.1854 0.00076516 0.1243803413	141.12347 0.0128754409 0.0046947028	141,01 141,0218 141,3218 0,09721924 0,2271190596	939,72046 G,4617746116 0,57229074011	986,7643X 196,2643X97 1,461648X83	91 940,4 930,2318 1.38469124 0.1242237346	150,46442 0,0800907364 0,197509075	156,2503) 12,46821734 2,339049944	159,78 159,75186 0,0007007344 0,0175991989
505.51181 6.0023222761 6.0486178203	105,019023 0,2160111529 0,4402898825	12 105,56 105,56178 0,000001884 0,00168624479	72,69895 0.0010969225 0,0400669398	72,96 0,0039 0,0039 0,0039 0,0039 0,0039 0,0039 0,0039 0,0039 0,0039 0,0039 0,0039 0,0039 0,0039	12 72,72 72,7466 0,00079796 0,00007965797	142,19919 0,0030469391 0,03882791614	141,864 0.226076 0.3348811031	12 142,14 142,14735 0,0000640235 0,000517995821	941,51227 1,140947303 0,1132773076	938.4385 17.1020225 0.4392791508	942,9408 942,9408 0.0389144888	193,89754 0,0797838616 0,1785865696	857,12291 8,162963268 1,787804488	150,38 159,46041 0,2446152801 0,3001573947
104,6181 0,00000004 0,00776218462	105,08895 0,2106351025 0,4387247073	13 104.81 104.81666 0.215709666	72,653406 0,001325542016 0,0602187371	72,54546 0,0019030116 0,06127300388	13 72,69 72,69713 0,000508369 5,000608778666	055969505000 62165100000 0	141,8881 1,14890901 0,74(8950809	13 142,6079 0,01042441 0,07142857143	894119000 12000010 0,0700010 1000010	2,007562559 2,007562559 0,1542165103	13 206.69 906.1812 1,61594944 0,130662901	159.09277 0,0314191779 0,1112764488	157,77802 2,23040251 0,50377032712	159.27 159.55171 0.0703605341 0.3768757406
2 105.117615 4 0,050005688225 9 0,052216835616	104,97243 0,0217768048 0,140824201	14 105.12 105.2493 0.01678669 0.1230622831	72,78533 0.0012020069 0,04761054005	72,63311 0,5349278721 0,2568465357	14 72.82 72.91221 0.00000941	143.66183 0.002300489 0.0035198648	142,5327 1,46757329 0,4359192907	143,74 143,600 143,600 0,08645378 0,185284865	9031 PF95800*0 903106300*0 9031055	80,70450988 10,1217947,1	14 930.5 931,57635 1.156529023 0.1156743066	159,65742 0,5410380364 0,120725383	160,40532 0,2973736034 0,341123463	159,56 160,16038 0,0502341531 0,1879081688
105,3429 0,00073441 0,00571496532	185,31033 0,003505589 0,0662902184	15 103,37 105,408914 0,0143703074 0,1130527602	72,78524 0,0020034576 0,06145619001	72,783865 0,00435780025 0,00084250234	16 72,83 72,87577 0,0020948929 0,00209489146	142-45238 0.0033200644 0.04043225097	142,92378 0,1712138884 0,2903515543	142,51 142,5124 0,006115776 0,05520458665	926,36264 0,0532871066 0,02487151564	926,9478 81,11799084 6,8423173478	15 928,173 928,1750 0.00000764 0.004004603051		158,78022 2,02347646 1,064694003	161,47 161,19403 0,0759605040 0,1703536061
194,77027 0,000096729 0,000396123009	104,805244 0,000637259636 0,02409238404	105,000,000,000,000,000,000,000,000,000,	72,99832 0,0026708224 0,07074608434	72,561624 0,007510317376 0,1206801506	16 73,05 73,05163 0,000026569	143.13986 0.0000000144 0.000000038	143,51372 0,1396660384 0,2810879476	10, 143,14 143,03976 0,0123743376 0,07771412803	935-5328 0.04717684 0.02321132781	0.0296,00720 0.036,007070 0.0296,007070	16 905,75 904,0247 0.4740115400 0.00775174033	182,29562 0,3774027944 0,3771284758	198,92192 19,25912371 2,663922245	162.91 150.5403 0.3545609 0.346763599
101.56059 0.0009418761 0.0098599235	105.097336 0.3216701366 0.5427494469	17 194.53 194.576#76 0.002980299079 0.04389654646	71,78459 0,0536096681 0,3040669604	72,97337 1,074901757 1,400662073	17 74.01 73.708666 0.0008308423 0.407288666	142,53825 0,0004313929 0,01456830415	143,0873 0.25736329 0.3558501864	17 142,56 842,513/8 0,0021548,964 0,0025617284	943,06086 0,06031495366 0,06031495366	017,13873 42,13658621 0.6679041573	17 943,633 942,6057 1,54116049 0,1085488016	0.0	159,59936 11,23365772 2,051921243	153,36 0.501930 0.501931 0.501931 0.501931
104.95911 0.0030657721 0.007727469281	104,450294 0,2497360991 0,4761657932	19. 104.9874 104.90874 0.0017023878 0.40801395903	74,29412 0,2264617744 0,6364564726	73,84532 0,8550331534 1,236693211	18 74,277 74,280226 0,23(8697552 0,0000281548	143,07083 0,0018428809 0,02833671288	142,50234 0,1916338176 0,3063616654	18 143,00 142,07540 0,002654065 0,00767317345	962,0043 8,618,03448 0,3073,259078	942,79439 166,7913371 1,308649638	981,204 981,2043 17,48701244 0,44027911991		161,2781 7,40873961 1,658896122	163.25803 0.5506194009 0.4524207317
105,3403 0,300000129 0,00034175063	105,40739 0,00454156123 0,0039778963	105.3 105.4996 105.49250 105.60181.0	75,65487 0,0072471189 0,1151338923	73.602294 0.1133708304 0.456377333	73,8402 73,8402 0,009944078 0,134980107	143.08916 0.0099690256 0.0092948362	142,67941 1,961696321 0,9720780122	144,0 143,7125 0,135028651 0,255038667	0.1446229498 0.1941003299 0.144622949 0.1941003299	945,1846 33,70266916 9,80381723	961 JU 965 JU 948 J 1684 5,63141578 0,2492738288	163,5836 0,217477898 0,2842689011	161,70921 5,479297894 1,428878992	164.0 163.4656 0.00.60691.0 0.00.60691.0
9,477349 198,024635 0,0001819901-0,000571903255 0,000187705231-0,0427692818	105,178905 0,178523292 0,3989188622	20 000010010000 000010010000 000010010000 00001001	71,59035 0,0063441225 0,1682054089	74,0386 0,5883186 0,5419025132	20 73,61 73,61005 4 1,000352,9225 7 1,0003783,912	142.99808 0.0017572964 0.0293064877	0.00781185982	20 8 143,04 4 143,03264 6 0,0606541696 3 0,0654541387	941.04103 0.1919093249 0.9485289316	9.60578984 9.60578984 0.3305846429	20 941,48 940,6805 0.35840025 0.06307833338	151,99316 0,0075111656 0,05317847976	162,49077 0,1667319329 0,2634396748	162.08 0.0001267676 0.0001767677
136,024635 0,002657160225 0,04278682818	105,10725 0,9256876825 0,997485322	21 105,07 105,734535 0.1128387862 0.3183975892	71, 23955 0.0049632025 0.09696050543	73,64046 0,0679403025 0,0275685599	21 73,4 73,41531 0,0002343961 0,022960431003	143,69674 0,0154405476 0,08639968425	\$43,26749 0,3052973601 0,3641677096	21 HJJØ HJJØ13 Q36971769 Q1699713531	941-2043 0.81222849 0.07597503238	941.10034 0,5457745106 0,09752937736	21 942,02 941,132 0,786544 0,19429551453	181,88223 0,0007711729 0,01715150002	161,96294 0,0029095236 0,03331480462	191.94 191.84021 0.00465484
107,00963 1,002741877 0,9271086011	105,29796 7,316180902 2,510915656	22 108.01 108.08015 1,162186023 0.871860334	74,20969 0,0170093760 0,1754371805	73.39483 0.9173497289 1.254898829	22 74.33 74.1339 0.0459893721	143.8614 0.00148088 0.02701188643	143,25885 0.1287733225 0.2511190641	22 142,9 142,81453 0,0073651209 0,059e1105668	\$48,7386 1,3269682 0,521286711	960,9545043 80,95456043 0,9426144311	22 943,89 948,0733 3.30039880 0.1912537241	161,27277 0,000:030779 0,0070:088875	161,10098 0,0252673604 0,03661063888	191.26 0.01136-03770,0 0.01136-03770,0
107,82558 0,4540423364 0,6227331487	107.44335 1,337638223 1,36800364	23 506.3 107.506296 0.65907533996 0.6590754635	73,9977 0,00529864 0,0052986497432	73,575364 0,09210207546 0,4104028116	23 73,98 73,985966 73,9503559 0,1275128413	142-45285 0.0000011225 0.002000712001	143.18607 0,5431247809 0,5173534514	23 142,48 142,49635 0,0023377225 0,00236173394	940,7186 8,4700308 0,07344302885	936,9908 29,36772864 0,5756471675	2 941.4 941.2779 0.017840465 0.0141107467	158,7307 0,0101075777 0,0435186804	161,13074 6,250700548 1,578460547	158.6301 159.76301 0.2840999601 0.3960983213
109,600784 2,52507485 1,442250658	107,7607 5,95010449 2,219721783	24 110,19 100,4016 2,45907656 1,423369652	74,568756 0,03657426754 0,2866105909	73,79892 9,9236747664 1,285553772	24 74.76 74,463 0.00840769 0.4196004168	144.09605 0.0699696025 0.1622104101	143,6973 1,35107129 5,8526375289	24 144,08033 0,2417393888 0,3384104653	942.815 9.225635 9.56035697005	940,48527 7,860901913 0,197228848	24 943,296 943,33307 0,301855024124 7 0,004865034124	161.476 0.000076 0.0148608111	161,01617 0,2360601289 0,3002043344	161.5 181.85(6) 0.0566602235 0.1945201338
106,17847 0,5790279409 0,6926778243	109,23646 0,45468,26025 0,45468,26025 0,45468,26025	25 108,94 108,2037 0,6429/27569 0,479/291068	74,53777 0.0202293729 0.1104525978	74,11288 0,3219051884 0,7588018211	24 74.60 74.50 9.03259752 0.2405081590	145,2953 0,21573,23609 0,31965,39517	144,25201 2,2740384 1,454677627	25 145,76 145,17049 0,347522400 0,34752400	945.0185 9.63809225 0.87727248719	940, 185 41,796225 0.8820345687	25 946,65 943,6102s 1,296030062 0,1203982464	160,82857 0,0011175649 0,0007920465	160,4305 0,18447025 0,2870023623	160,68 165,61836 0,0034047226 0,00627377844
199,250534 0,301913882 0,500434408	109,51468 0,0012027024 0,03158,469646	28 109.39021 0.250885441 0.464250175	74,96143 D.066654449 D.0407973674	74,51581 0,4618967561 0,9210022803	2 75.2 74,89548 0,0962923927 0,418864141	145-09108 0,1564199204 0,272835112	144.JR206 1,JT5401962 0,J*10350024	26 145,59 145,63417 0,30(6)489,889 0,380731567	949.2553 1.4036.1403 0.1349475317	943,6468 43,47029024 0.6626947921	28 981.44 948.47504 3.861067802 0.2067421408	159,67737 D,0057491108 D,01714375108	160,78999 1,39729622 0,7134293768	159,708 159,708 0,00209,003 11C7848(C90)
106,89328 0,0)40771364 0,2)60791369	909,544946 0,1186187637 0,2187930403	27 109.2 109.599.43 0.0545549449 0.2138919434	74,63825 0,0173560825 0,1762071348	74.8212 0,00202144 0,00847466178	27 74,77 74,08285 0,1075510488	145.54282 0.0388796534 0.1354818545	144,95103 0,3459,363249 0,4040907360	27 145,54 145,20152 0,1145657104 0,2325663661	808.47705 0.4283437025 0.66845378073	945,56983 29,4008194 0,5775403402	940, 13 940, 1108 940,01108 0,014,491008	158,43591 0,0243079391 0,00850265353	1.612MEE1 97.1 ZEC1.9059'9 1.6228'791	158.28 158.507.8 0.05.10016868 0.1435178190
109.23477 0,1642113529 0,3692697142	109,77681 7 0,0012000001 0,00220000144	28 109,74 2 109,44024 0 0,0348500576 4 0,2731547294	75,09891 0,0454073481 0,2820504714	74,0581 0,1392707761 0,4055364411	75.31 75.070045 5 0.05750280023 0.3188628002	\$44,7605 0.00354095 0.04108548543	0.0216326276 0.1013257637	26 164,7081 164,7081 0,0124300001 0,07608522304	804,756 0,285156 0,6870945908	0.7541204333	26 935,29 935,4499 1,3453601 0,1240150114	159,87608 0,0000103064 0,000451838875	160,0296 2 0,4167999 0,400,00099	159.88 159.00717 0.0007582089 0.0007582089
109,65733 0,1021431269 0,3656460741	109,42529 0,4028667841 0,5796945303	20 110,05 0,050,001 0,050,	74,985,224 0,02715113018 0,2102335349	74,84941 0,0964661461 0,4132384247	29 75,16 75,00665 0.0235162225 0.2060313697	144.02542 0.0006041764 0.01700449071	144,85564 0.080692836 0.2116489461	24,450 144,65 164,45007 0,009080346 0,00903174831	928,96357 0,0457403799 0,022300295778	70,05187809 0,0002097336	29 929,75 929,6784 0,06512958 1,007700994991	158,74413 0,0054967509 0,04671980673	1,490565932 0,7616685725	158,67 156,11711 0,1980075821 0,3817965959
1/26/2015/20 E,15094335/29 0:157/301390	110,00398 0,1042603004 0,2064060322	0,000 (0,000) 0,000 (0,000) 0,	75,258484 0.02594805828 0.2408097961	75.21178 0.0526990625 0.3026583245	75.44 75.45286 0.0359465478 0.346633079	2412980122 0212980122	144,29798 0,0694713816 0,06374402659	144,39 144,2949 0,0002901	0,54,547.46 0,54,517.32516 0,52,26664206	018,3195 1,96805025 0,1403819781	30 506,3000 0,310,000 0,310,000 0,310,000 0,310,000 0,310,000 0,310,000 0,310,000 0,310,000 0,310,000 0,310,000	158,74588 0,0000571744 0,01000441	CHEFFORT DEST	158,72 156,126 157,124,02 0,750,000

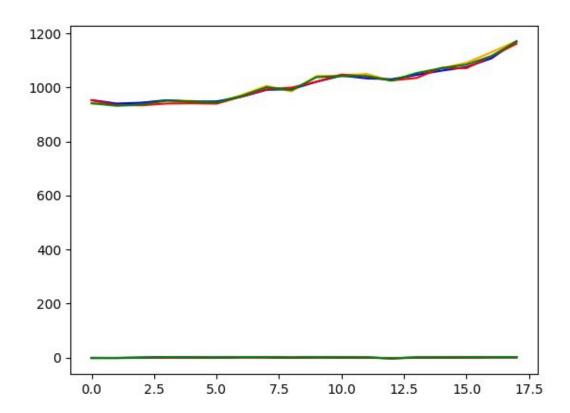
106.627246 0,07714728402 0,055733044	109,782474 0,01881340068 0,1281373670	31 109,9 109,00162 0,0062177434	74,84387 0.0982409789 0.1282730841	75.303375 0.1326113906 0.4848879103	74,04 74,91911 0,0004383921 0,0278763304	145.72217 0.0249103099 0.1081910844	544,93161 0,7862367921 0,00000000005	31 145,08 145,36313 0,2611543999 0,3641151311	946.5588 0,33779344 0,06133777984	938,7912 78,12438544 0,6328155912	31 947,54 940,6430 3,89481444 0,210871846	156,70403 0,4019540409 0,4063471368	194,51973 5,901598223 1,567072467	31 186,07 187,2606 1,421664726 0,7638111194
100.39531 0,0000750461 0,2140746146	110,01129 0,145362041 0,3477971388	32 108.63 109.64429 0.0212314041 0.1329106966	74, 16499 0,0020009001 0,06060230021	74,25475 0,4157025025 0,9588132166	32 74.21 74.29776 0.00753007478 0.1189030078	145, 18188 0,0081027344 0,05377942899	144.87874 0,3401822276 0.4015888941	32 0,0139449138 0,0139449138	947.1116 0.19219456 0.04636868777	540,05403 56,1760743 0,7909646705	32 947.55 946.7394 1.02030469 0.106788578	154.96419 2,478074106 1,006,26638	156,89781 12,304731 2,288857031	32 153,5484 6,16073191 1,4723778
105,982156 0,08285474462 0,2810965467	105,531546 0,1142805069 0,3060693299	110,27 110,29094 110,09094 0,02934574964 0,1553514102	74,28131 0,000031191 0,000006061	74,541125 0,1050687050 0,6790812100	74.41 74.415245 90.000027519025 0.007048787810	145-00128 0.00(1642498) 0,05416090229	144,87603 0,2052724249 0,3121821815	10 145,19 145,08044 0,0024861938 0,0341486627	942,9679 0,0652241 0,0090707165	1,666,1023 1,666,10235 0.1376606762	33 36.17 36.	152.53825 2,716778063 1,085160313	154,2447 5,54461209 1,55628684	13 151,49 154,6981 4,039164036 1,446391527
108.79422 0.0242874084 0.1429830197	109,297424 0,1307034358 0,3168538917	108,903 108,903 108,903 108778978 1,07501102341	73,903408 0,001084000000 0,00034024925	74,31783 1,119004309 1,443036364	34 73,422424 0,02638155578 0,2217088817	145-707 0.024549 0.1117433331	145,14200 0,0290529681 0,4690128186	34 145,87 145,58115 0,0369643228 0,1296648918	934,5963 0,0121183 0,0242218668	940,6349 42,98671401 0,7015990923	34 534.28 594.35364 0.0054236486 0,007662005365	162.17962 2.08533613 1.082377948	102,92969 5,643902978 1,578007307	34 150,00 152,00363 5,967415439 1,622,003798
107,94154 0,0001009716 0,07277000148	100,673294 0,5674518504 0,68601149	35 107.92 107.02941 0.0082065481 0.00394180875	73,315806 1,58312771743 0,0763390663	74,04159 0,0108529281 1,088871417	75,26 73,26426 0.01544676123 0.1686681948	146,28213 0,0772006235 0,188501009	145,54041 1,021292148 0,0895401301	145,55 146,232 146,232 0,1972562 0,223467869	837.4191 0.00011881 0.001162753486	937,83909 0,1673219025 0,0436362561	35 937,452 937,86206 9,1866758436 117,8636964137	163,49783 0,1280423089 0,2318520086	102,10172 1,078025358 0,8778938934	35 151,14 114,29536 1,2020635 0,7168200731
107,54963 0,0301223389 0,03012557881	108,14854 0,0284057316 0,1860844601	36 107.56 107.565 0.0286568 0.151231758	73,849724 0,00000076176 0,000073750854	74,08353 0,0545362809 0,3182220718	70,865 73,8688 73,8688 0,001519464 0,01276386757	140,545M 0,0128051686 0,07786776803	145,66702 0,0000452804 0,000019442537	36 145,5786 145,5786 0,007883428 0,00134148016	856,63054 8,417892892 0,3363509741	944,1667 247,5367289 1,636056152	36 363.9 954,88706 24,1387944 0,5116176877	154,44102 0,0140254404 0,13882163	159,27074 1,593572148 0,678039282	36 154,23 159,00138 0,5945728384 0,5900842887
169,78526 0,0002166784 0,01352941176	107.8772 0.85155934 0.8481617647	105.9 105.9	73,67847 0,0002717409 0,0114d608108	74,275734 0,1640200786 0,1649254056	37 73,87 73,8648 8,000106854	145-57332 8,0076134224 9,05963844432	145.18073 0.2290997329 0.3290933064	145,50977 0,0228750628 0,103151177	901.6001 9,92187001 0,3264767686	961,4593 179,3109466 1,387910673	37 964,54 960,46265 18,04074656 8,4402356014	163,6074 0,05171076 0,1463559409	153,70418 0,1799289724 0,2797353862	57 163,28 194,0673 0,61954129 0,5136351775
109,60325 0,0021906025 0,04290566891	100,568146 1,170406077 0,3866429549	38 109.65 109.675.05 0.0058275025 0.00284541734	74,48639 0,000130321 0,004646389012	74,15566 0,1118508085 0,4485864411	36 74,49 74,47808 0.0001420884 0.01000214794	145,01796 0,0038489619 0,04278261573	145.48562 0.1460256704 0.3857292508	38 145.08 145.20241 0,0149842341	967.32086 40.9459274 0,657154617	994,9179 84,97894441 0.9459458007	36 973.72 965.3441 54.40390081 0.7574670217	154.20564 0.0073342566 0.0856706056	155,67847 2,428628741 1,011206564	38 154, 12 104.57758 0.2000754664 0.20008865206
110,43923 0,014865929 0,1092349546	109,71785 0,709,2196225 0,761713097	39 110,56 110,47606 0,0070450236 0,007542257548	74,62101 0,0001212201 0,01470673502	74,38889 0,5480896321 0,2003443761	74,02907 74,02907 0,0003048161	145,55075 0,0187059425 0,0812900595	145,50064 1,14363081 0,7391422337	39 146,06 146,2396 0,178941225 0,2866153007	984.8365 6.93532225 0.2722046228	988,4775 1,01505925 0,104137598	39 967,47 964,27803 10,18292774 0,3238365641	153,85785 0,9022886225 6,25110561061	154,7551 0,9021401 0,614450398	39 154, 16807 0,1286441689 0,2331502867
110,747804 0,03191622083 0,1734234721	105,47838 2.13457944 1.316946097	110,34 110,7601 0,00236401 0,162159726	74,2682 0.000048568 0.0111500134	74,82448 0,1338450704 0,4608180917	#0 74,26 74,308205 0,002123772075 0,00411381032	146,64764 0,0176191900 0,00017517327	\$45,03430 0,7100062721 0,5781070001	40 840,78 940,61086 0,6774360366 0,183360866	968,3031 94,26497381 0,3865379604	964,3644 59,530-653-9 0,792720681-4	972.08 972.08 968.36224 13.82173942 0.382404119	154,4876 9,0005776 9,004919730709	150,0354 0,0046916 0,3565287416	95 154,46 154,04000 0,7162822016
110,029694 9,036171244636 0,0903771169	110,28702 0,5278356904 0,1516709045	110,12 110,12 110,15081 0,00094925951 0,002947856893	8 74,703319 4 0,000177315056 7 0,01183850721	74,54041 0,5223771081 0,3902811821	41 74,60 74,7114 0,00045796 0,02965176061	546,37634 0,6107453956 0,617675134025	145,885 0,354020 0,4061987985	41 140,48 545,18478 0,5871548484 0,78154348727	964.0179 4.67467841 0.7220362975	96,5398 9,61027344 9,61025434949	41 965,78 966,146 0,401959 0,56557951938	153.52727 0.0022344029 0.0027380115	1315998900 1,315998900 0,8780231952	153.48 154.14060 0.4454834025
110,75876 0,0256983376 0,7453980298	109,86006 1,108577784 0,9483772088	42 110,92 110,79790 0,0149010849 0,4100002899	75,77198 0,0390198416 0,2536816481	74,788 1,397124 1,65587732	75,72911 75,72911 0,05770001385 0,31770057133	148,36848 0,022880104 0,1032710378	146,44283 0,0768232089 0,1869168508	42 146,72 540,5725 0,02175625 0,505316249	CHELIBRAGOY O BESILBEROY OF CHELIBRAGOY O	980,14676 257,4179581 1,626543733	42 985,19 979,5248 32,08448104 0,57563962874	155,38366 0,0004137166 0,01308064541	156,3111 0,00622521 0,05577546818	42 155,39 155,43622 0,00300509325
110,80928 0,0206309184 0,1446319386	111,10797 0.0190081389 0.1242407869	110,97 110,96246 0,0078630516	75,8612 0,01836544 0,1826519789	75,273705 0,507500427 0,91085131188	43 75,879005 0,0146,987000474	185,31416 0,0275029866 0,1132188214	145,20602 0,0750600404 0,1670425697	43 48,287 48,287 48,189 48,189 48,189	981-239 54,774901 0,7448371644	978,83356 219,230665 1,490121171	43 593,64 993,6044 91,39600051 0,7865713136	195,23141 0,0547545881 0,04418813607	155,81502 0,2652456014 0.331629105	43 166,3 166,478/2 0,0312334629 0,113796565
111,79615 0,294011925 0,4570347	111,044914 1,800442587 1,126423293	44 112,31 111,7691 0,20257281 0,4816133913	76,0694 0,0184234036 0,2684018096	75,866126 0,1883475156 0,1887189436	64 76,39 76,1034 0,03461175	147.14548 0,0597900304 0,105999932	146,88831 0,2736231481	97,000,000,000,000,000,000,000,000,000,0	987.46375 23.48613808 0.4863808472	987,6304 22,84457816 0,4816639461	44 962.31 987.5197 27.98727499 0,6331297679	155,76381 0,006049161 (0,04689988706	156,5438 1,01767744 0,6473305965	150,000 (200) 100,000,00 100,000,00 100,000,00
112,0131 0,11350161 0,2908664887	111,824484 0.1810638863 0.378741433	45 112.06 111.98062 0.1234578044 0.3127547842	76,102554 0,03513600292 0,2457019269	76,13827 0,0230019629 0,1966658304	45 76,2964 0,0265731903 0,2215700236	147.8803 0.36402009 0.4175063873	147,53807 0,9253093249 0,6477043098	43 146,5 147,62529 0,4562335941 0,4542501694	0,30294975 0,30294975	990,7327 0,60073829 0,296892083	45 983,887.2 17,08003844 0,4183842863	155,84343 0,0032001549 0,0362608043	0,011289456 0,011289456 0,01128456	45 155.7670 0.0176730436 0.085277851385
112,353825 0,1268626366 0,316160636	111,24923 1,960271985 1,262236819	48 112.68 112.321205 0,114782052 0.3007234156	76,22833 0,0367373869 0,2556113059	76,243774 0,03106560308 0.2308019367	76,45 76,25023 0,02349,216529 0,2221536864	141.04829 0.04899041 0.0476001300	147.40855 0.0147001025 0.08227204888	46 147,4623 147,44623 0,0291623829 0,1166021869	998,6809 49,57950809 0,6830741312	991,3852 203,494519 1,418465859	48 1005.66 998.00578 99.12003088 0,7581400081	158,50519 0,007070301 0,02862344199	165,74109 0,6543353881 0,5167103162	48 186,55 196,47214 0,00621736 0,04973490897
112,38615 0,001610825 0,2342210366	112,185656 0,2084294703 0,4033236686	47 112.65 112.43114 0.0478509508 0.194283178	76,79634 0,1047567968 0,41968361	76,78027 0,1261249128 0,4680747928	47 77, 12 76, 72087 0, 1500-07569 0,5176-00871	147,0178 0,00714884 0,00207626335	147,07837 0,002398569 0,00389804802	47 147,03 148,97987 0,0028130169 0,03409500364	969-0233 32,705/0289 0,58945/25028	902,668 163,43068 1,27121762	47 1005,65 1000,07434 35,7623,2444 0,0047143638	166,0831 0,0009661	168,25004 0,5621400578 0,4806163848	47 558 110-1056 0,00563136 0,04769230769
112,49099 0,092459216 0,2001907544	112,42254 0,0884824516 0,2638828316	48 112,72 112,50716 0,045300865 0,1886216595	77,1321 0.5260241 0.401600472	77.17635 0.0960045255 0.4660523039	46 77,49 77,047366 0.1969337107 0.5712272861	147,080% 0,0001860496 0,000272863671	147,04532 0,0029899024 0,00717199184	48 147.1 146,96589 0,0177182721	1002.06554 25.04401989 0.498536244	959,4825 79,85216625 0.8522005614	48 1007.87 1003.051 20.421.081 0.4483713777	156,98376 0.0000385376 0.000974775463	15A,53236 0,204-243696 0,2945090133	48 196,89 196,60064 0,0324216036 0,1148962035
\$11,78764 0,0027410096 0,04681000\$120	112,50585 6,443366225 9,685368421	49 111,54 111,635 0,0008864 0,02342632330	77,3067 0,115624088 0,4363971651	77.24227 0.1802A377529 - 0.6250980246	49 77,85 77,24388 0,184034544 0,5230139222	145,02724 9,0000079176 0,001879724852	146,40002 0,0000001024 0,000017939110	49 146,83 546,77121 0,0034560841 0,0400009744	1004,78976 20,90491034 0,4518987467	995,70734 186,1221719 1,351626276	49 1000.35 1000.4271 12.41082441 0.3400260013	159.27429 0.3605677979 0.3785904178	156,90012 8,375664014 1,863822967	40 129,40 138,46763 0,8325467636 0,5707030273
111,31851 0,0004490161 0,0190017946	111,67504 0,1149481218 0,3945087121	111,34 111,30124 0,0015023376 0,00461228696	77,31904 0,076,706,016 0,366,052,056	77,544554 0,00978130916 0,67031060704	00 77.59 77.29161 0.0000028011	146,5426 0,000,0876 0,001774258588	146,19624 0,1167998976 0,2332196667	146,54 146,54 146,55787 0,0004897399 0,01510167872	1006.7527 16.03958729 0.4201089032	10,7767784 10,7767784 0,3247082097	50 5011 1007,10735 14.71505922 0.3603940762	159,79898 0,490297840A 0,4181591575	168,09317 5,930140449 1,518557886	50 160,47 110,61291 0,9160212851 0,5964282391
111,71085 0,0004899025 0,000487170797	111.73229 6.0014220441 0.00373892816	51 111.77 111.73104 0.0014081656 0.00418104006	77,34576 0,969627776 0,3404715887	77.59707 0,0001671849 0,0168602242	51 77.61 77.4008 0.0415833884 0.2627488407	149,74834 96,73098,38 0,1315489,55	148,54007 120,7780614 0,889942519	\$1 159.53 589.48153 591.7740881 6,3113750	1008.7598 15.84198204 0.3600130142	1006,1697 43,3003,4609 0,6487521575	81 1012.74 1000.42773 10.87113266 0.3270032534	150,36598 0,555517604 0,2666524497	159,11409 0,4171997281 0,4043002003	51 199,76 199,00406 0,5706995368
113,161295 0,01606485122 0,0109070647	112,27881 1,357687016 0,6867637037	52 113,4 113,10118 0.088292924 0.280399302	77,61323 0,0800724329 0,360013875	77,71614 0,0375816896 0,348831688	52 77.91 77.60366 0.066574494 0.315664165	157.10805 14.3708048 2.306712244	156,92143 15,82901924 2,473697328	82 140.9 155,78756 26,34194035 3,186032194	996,3526 6,18715976 0,348283159	1008,9312 50,28511744 0,797a176156	52 1531,84 990,44244 3,984251934 0,196299377	154,17024 0,0361912576 0,1213643544	158,92183 8,654363749 1,68603026	52 155, 1861, 551 0,511, 551 0,1381, 551 0,1381
111,000 151001210 15100 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 151001210 15	112,54046 0,2605694116 0,4506438092	53 112.03 111.9757 0.002844369 0.046703724	78,363774 0,1996537743 0,7669661211	78.15732 0,4259911824 0,6261690141	53 78.81 78.3758 0.18507204 0.5458638135	156,39816 7,5873856 1,54871803	150,11224 8,74034218 1,62499500	23.0927117	1002.97815 4.37593423 0.208129782	908.29199 46,4656000 0,676365886	23 205,07 204,250 0,645,000 0	158.4147 0.02712509 0.105408	137,64874 1,948090148 0,8822738	53 156,30 156,300,00 0,000196036 0,000196136
112,25488 0,0030302144 0,04907407407	111,06161 1,5835-6392 1,120361-657	54 112.32 112.21199 0.0116274939 0.06016274939	78,43682 8,5546965124 0,496759554	78,36426 0,21091,17470 0,50001166700	64 74.43 76.47838 0.4239273721 0.444787200	138,75693 0,8373631049 0,5736316382	160,89175 1,800293962 0,840836947	54 159,55 159,01904 2,34,9036522 0,5095487308	884.06146 2.186080502 0.100233375	1002.34073 202.6670227 1.705636504	\$4 \$65.24 \$85.29007 \$45.1501949 \$45.001911.00294	156,25434 0,0059000356 0,005040853917	157,04909 0,7727992281 0,5629059078	54 150,17 110,08038 0,0071774784 0,00424857927
112,24945 0,0025503025 0,04501335708	111,869 0,47864461 0,3731907943	16 112,3399 0,01808121 0,057460392	78,525314 0,7120147186 0,4244003752	78,28054 0,1692324096 0,7600104337	16 78.36 78.56936 0.0730460081	155,67430 0,0422918235 0,1318284060	160,13057 18,1439368 2,752596686	55 155,294 155,294 0,4194637156 0,4190979471	987.71964 0.503608116 0.50794313537	995,82904 63,89160812 0,7424495948	55 588.49 588,1998 0,502489198 0,00238677174	157,15782 0,003931524 0,0080458307	8000889410'0 9209800'0 9410'141	56 157.1 156.63077 0.0779693833
112,70759 0,0085396081 0,08192375887	111,72877 1,147533713 0,9496719888	112.8 112.74665 0.0028348025 0.04711676433	78,376915 0,06405201722 0,0216663437	78,6975 0,0455635 0,0464500720	78,83 78,83 78,455005 9,0000575302 0,2224094115	153.49919 0.0500175541 0.00272973081	158,85685 28,74904286 3,493102188	96 153,485 153,925 153,925 153,925 154	990, 68035 0, 603 1483225 0, 67833397212	984,13636 53,63584932 9,7386732667	951.46 951.46 950,1004 1.30427568 0,1277509900	196.46774 0,003339076 7 0,0081579022	155,83022 0,3303720464 0,3674828975	56 196,41 196,27171 0,0191241241 0,0041505286
113,24172 0,025020084 0,1395767196	112,46836 0,6869539844 0,6206526673	57 113,4 115,243164 0,0245975309 0,136303351	78,52865 0,0543589225 0,2060259035	78.693216 0.02701090000 0.2117023159	176,7% 76,7% 76,61234 0.0218034756 0,1874009648	183,71386 0,0128956736 0,07743229167	155,84671 5,947705824 1,462701823	153.4 153.375.26 0.0505080878 0.1403151043	990,70764 0,5074087896 0,0718024944	091,1081 0,09726581 0,033145962817	57 501.42 591,07806 0.1160228436 0.03446982264	157.42822 0.000331565H 1,011157486818	155,85419 2,420544756 0,985380662	57 157,41 167,260,37 0,0136256929 0,0741569698
113,841415 0,09022470223 0,2703928953	112,90401 1,95249108 1,991537451	56 114.15 2052011 205205 2007305 2007305	81,823786 4,779462302 2,508525236	80.54129 10,69446606 3,900143181	68 83.41 81.572075 5.00830856 2,070236348	1:0-77629 8:092717641 0:0926916608	153,00124 0,0031210376 0,1034825613	153,67154 153,67154 0,000718718	1028,3708 63,27832064 0,7051441272	1607,2414 885,47088 2,559773438	08 1033,67 1024,6164 81,03146256 0,675676866	161.51736 2.34896537 0.5093816057	109,26866 14,2970197 2,31900644	58 161,051 161,03616 3,250116546 1,112444036
113,65034 0,02221951166 0,1315114236	113,02043 0,0322453648 0,1877963761	50 113,8 113,8733 0,01975288 0,1113356786	82,510864 1,902016106 1,643(8)404	82,77374 1,340030308 1,350623436	59 53,80 82,492704 1,077549172	1943/125 0.007280107564	155,31439 0.8108922721 0.8182854167	59 154,38 154,69781 0,1141155961	1029, 1399 15,92069901 0,3862147068	1021.4 137,569 1,135354706	59 1033,13 1027,9190 27,14514201 0,8043024585	153.52752 8.364706152 1.734752870	162,1682 26,81031924 2,736204415	50 50 17 201 50 17 201 50 17 201 1 10 10 10 10 10 10 10 10 10 10 10 10 10
113,9716 0,00210000 0,0	113,439575 0,5792461606 0,6688712785	114.2 113.36566 0.054005824 0.2051838876	82,370364 0,6410100045 0,362514263	82,36122 0,6704000884 0,5843471988	83.18 83.18 82,400734 0.6072554988 0,8388426911	103-0629 0.00943841 0.0072791183	155.72676 2,77908888 1,001890173	80 154.96 153.99581 0.0041075281	1029,1027 12.51249129 0.342410557	1023,80484 83,63395143 0,8852697854	80 1033,04 1029,1436 14,87027844 0,3712866104	186,0707 8,81674249 1,756566403	166,40848 6,92229524 1,556448178	90 188,04 188,76913 10,56323836 1,917220776
115,286150 0,9875400163 0,8481718414	114,64348 2,64596731 1,386914315	01 116,27 115,22942 1,04486896 0,4856296746	80,51537 0,417330369 0,7890363062	82,454836 0,5391529025 0,8770413591	83,18 83,10 82,50127 0,4506773864	41590518250 199598791 64608791	154,22778 0,0391199294 0,1394035977	51,03 153,54946 0,0524486916 0,05224651622	1038.4764 27.488.47896 0.5873383439	1026,1128 271,6277836 1,581154307	61 1012.6 1035.0666 45.26569204 0.5655206134	185,9204 0,94012416 0,9800814948	168,39742 2,272315056 0,3022416562	61 595,89 195,47193 2,611206140 0,6497533172
115,24017 0,0783040286 0,2422351108	115,342735 0,0344288022 0,1534496191	62 115,02 115,21344 0,000700336 0,2653734612	83,26944 0,6092739136 0,9286850064	82,69822 1,827309168 1,808304681	94.055.152.0970 941.052.09 941.052.09 941.052.09	150.22798 0,0148800004 0,019888612	151,44067 3,640541048 1,245879860	64 67.031 84.137 84.136	1037.5864 26,96314096 0,5161197559	1034.1311 78.12815321 0,8474743405	62 1042,97 1009,4100 20,710,9100 10,710,910,00 10,710,910,00 10,710,910,00 10,710,910,00 10,710,910,00 10,710,910,00 10,710,910,910,910,910,910,910,910,910,910,9	186.58448 1,31221007 0,5814105803	106,03448 2,11853647 0,8659142602	62 168,11 166,8540 1,87602916 0,7407728403

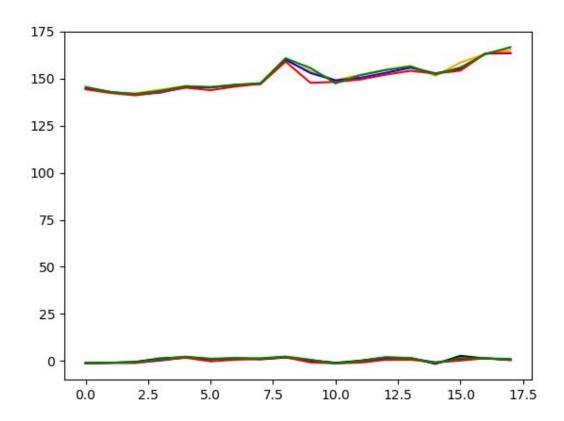
114,90834 0,010347556 0,0884882979	115,19396 0,0817731216 0,2488896261	114.91 114.92 114.62767 0.183431856	83,45902 0,4623253624 0,608965025	82.91007 1,612727805 1,481798104	83 84,14 83,189935 0,0021167029 0,0021167029	151.51907 0.0037124949 0.0037124949	151.56152 0.0003415194 0.012191582	63 151.88 111.42014 0.0256682190 0.1056621621	1044.042 36.378704 0.5664815955	109,5784	63 1049,99 1043,784 27,41150736 0,4996333203	170,18483 5,360912129 1,342127536	168,68445 14,3684216 2,211913043	63 172,5 165,67972 7,663979278 1,654944928
114,86901 0,0148615901 0,1060874859	114,91098 0,0082425901 0,08871032264	94 114,59 114,619756 0,02898301954 0,1480511349	83,84522 0,2802600464 0,736447212	83.22424 1.551917978 1.474795785	84 84.47 83.96276 0.2572534178 0.8004872179	150-89469 0.0002157961 0.000738796075	152.59276 3.072167818 1.16198947	94 193,04 18 1,31403 0,22548,36249 0,3147305065	1039-5818 9.7969859 0,30003-4984	1041.0265 0.96719225 0.07226569903	64 1942,68 1941,4777 1,44502529 0,1152086278	172,0702 4,79157804 1,25081280	171,4151 8,0365801 1,636615352	64 174.28 171.33415 8.602181223 1,673(71893
114,19851 0,004618201	114,78150 0,203108994 0,4741501392	114.22 114.24794 0.00796438 0.0044813654	83,777084 0,3429661931 0,584924647	83,69468 0,336,463000 0,6623643372	65 64.27 83.7686 0.264391368 0.6101598827	151-3291 0.00043831 0,0130908197	151,3460 0,00000049 0,00045504129	65 151.05 151.06400 0,0462207001 0,1425462256	1045.341 16,394401 0,3647452989	1047.9882 23.24975624 0.4581781514	65 1052,39 1067,3241 25,46334281 0,4813709746	173,16034 2,771378116 0,9436574321	172,97786 0,9687,0898 1,04875083	65 174,81 172,53789 6,111327852 1,41418967
114.5384 (LODS-633296 0.66267015707	114,43161 0.0263651921 0.1469371728	98 114.8 114.5(50) 0.0072219004 0.007415057786	84,08957 0,2119857849 0,5445008481	83,82401 0,54 16812801 0,9703760643	86, 84, 1916 84, 1916 0,430,666,9912	151,48804 0.0067174436 0.0540740252	151.40872 0.0260112384 0.1084082889	96 191,67 191,80273 0,0541932529 0,1535462163	1053.452 23.498244 0.4571525783	1040.397 79.065449 0.8403178713	86 1054,29 1052,9016 29,/1115044 0,6089531225	174,53579 2,573489774 0,9109417138	174,5612 2,61166824 0.9514299884	96 176,24 173,86697 5,627,780281 1,34534159
112,58848 0,000M03104 0,04303977273	114,257256 2,61561697 1,436774148	67 112.04 112.762005 0.02018542002 0.1260887798	83,73416 0,1266221006 0,423166608	84.0157 0.0052049 0.0853571198	87 84.09 83.85027 0.0574704729 0.2800874064	150-16403 0.0184028409 0.0004007302	150,34288 0,001838984 0,02852960745	87 193.7 193.71916 0,172857859 0,2762209916	1045-3947 5,83387409 0,835839149	1054,467 45,522009 0,6439697529	87 1047.72 1045.0076 4.87587376 0.2263910205	174,72826 1,20660009 0,854644212	175,70699 0,0296024601 0,09538620659	87 174,88 174,27925 2,56340063 9,8101375936
112.83457 0,0519854549 0,03958610889	114,185 2,65364 1,41936897	112.59 112.46085 0.0167236824 0.114858237	83,57973 0,0845382129 0,3472874587	84.09401 0,060180401 0,267081828	85 83.87 83.89427 0.0003474329 0.0107552841	145.2537 0.01125969 0.07126575489	140,00164 0,550028895 0,4972110485	08 149,140 0,4714058291 0,4603042711	1042-1556 3,58675136 0,1814388712	1041,7146 30,9477316 0,5329319923	68 1041,15 1041,2527 8,39434729 0,277479294	173,02561 0,0541104721 0,4261603473	173,2924 1,82543378 0,7944123254	174,07 173,54426 1,267313062 0,5445510591
112,4388 0,0047446544 0,004729749033	112,53419 1,36633636 1,036032749	19.37 112.47536 0.0111346444 0.01077947851	83,68479 0,0703363449 0,3150806161	83,31976 0,3723928576 0,7270826822	69 81,93 83,69672 0,2544195584 0,2779456773	146-36371 0.0021427641 0.03119272237	151,76053 11,29316188 2,264526086	08 140,4 140,897 0,237109 0,3281571159	1005,5479 3,42027441 0,1778612808	1047,4216 36,70630658 0,5975412865	1011.2 1011.47 1020.647 1.850859 0.1299462159	173,56963 0,3053719369 0,3278553774	173,07463 0,8016874369 0,5146891958	173.97 172.96340 172.963784 0,6395097803
113.12807 0,002107246 0,04250872783	112,74152 0,114565104 0,299279084	70 113,08 113,06254 0,00140370159 0,0011028672	83,79288 0,0962238754 0,3061748969	84,00383 0,0021316989 0,05460158854	70 84.05 83.8854 0.0235498718 0.1525617805	0,0000000 1,299	140,16507 0,0736600049 0,1947471289	70 148,69 148,6226 0,40119603	3.4634.5736 3.4634.556 0,17617867	1031.8848 3.90220516 0.586432549	70 1041.64 1040.6627 0.01442209 0.08190318272	171,01875 0,10201505 0,1674027546	171.50822 0,0719419884 0,158542547	70 171,34 176,9172 0,17675964 0,2467596294
112,88151 0,000009901 0,001424283409	112,83884 0,003,6621456 0,002,62615167	71 112.88 112.957634 0.004027037898	82,60644 0,0301230736 0,2001668336	83.53545 0,306741366 0,9696131588	83,0158 83,0158 0,0128154 0,04314292001	147.70271 0.3632593441 0.4097280781	140,25943 4,863127925 1,46800136	71 147.5************************************	3,42953361 3,42953361 0,1786841115	1045,0222 74,1698004 0,8309645797	71 1036.41 1036.3309 1.16445881 0.1041190283	168,90775 0,079070000 0,1018748501	171.42793 5,508549421 1,388118051	71 186,08 166,9502 0,0161298294 0,07510254412
113,92934 0,0002149150 9 0,01296642092	112,849844 0,8804080043 0,8680152712	72 113.84 113.800004 6.0.81937552842 0.1221680523	83,07196 0,0163942416 0,1539942308	83,3311 0,91716721 0,1575721154	72 0.105555688 0.105617788	149,29453 0,000495599 0,1171070079	148,8274 0,8681476 0,1962178112	72 HQ.17 H0.54345 0.1784671049 0.2832819006	3,5193,5854 3,5193,5966 0,17933,53664	1036,7354 162,1700372 1,214588877	72 1045,47 1045,1100 11,227,1040 0,31957,2040	170,91236 0,0052453076 0,1097253088	171,47336 0,139,5976696 0,219,2111722	72 171,04-06.0 171
113,727036 0,000527345398 0,00018813187	113,242874 0.2571767790 0.4458250048	73 113,75 113,69676 2,002917728258 0,007748659341	82,312584 0,007641557056 0,1060873786	82,87980 0,2303328049 0,5824283334	73 82.63674 0.165949878	149-1403 0.0290804809 0.1144727128	1,09839419 9,7(3946412	73 146,091 140,00010 0,1842040001	1054-255 2.739025 0.189765980	1005,6381 0,1236361 0,0338707884	1035,4631 1037,4631 1,67025961	170,00078 1,002272578 1,003771084337	169,70483 0,1980873049 0,2815750808	77,18 170,03 170,0313 0,01400968 0,0140747473
114,00584 6 0,000103858 1 0,0046952593	0.4983230464 0.4983230464	114,00 114,00544 8,0,00418447608 0,01790915802	62,435875 6 0,028858515635 9,1140494565	83,4742 0,69151364 1,144068523	74 82,63 82,64000 82,040000 0,01710094000 0,1594600881	150J8002 0J081036004 0J090098T94	150,28403 0,0156425048 0,05309746861	160.81 160.810 160.6025 0.0054025	1033,7388 0,84658401 0,06932775371	0,01679616 0,0125259518	74 1034.66 1005.1180 0.21022225 0.34421407419	169,01434	0.136384 0.2186482764	74 169,87171 0,000687241 0,00487704358
115,23307 0,00330868 0,004720271	2,70369664 1,426229634	15.19034 9 115.19034 9 0.0178385805 0.1159128133	5 83,569396 5 0,022698409 0,1799617714	0,9934300941	83.72 83.64775 03.64775 0,0052200625 0,00528607	0.0000019044 0.0000019044	150,56453 1,91936966 0,9157539664	75.95 161,794.6 0,036664225 0,1262596277	1,20318961 0,194636678	1034,7181 242,764448 1,483471389	75 1050.3 1040.009 5.348681 0.2181281539	172,98671 0,023,497,8241 0,028,653,038038	8,45793744 1,467713669	75 173.14 173.95483 0.042047289 8 0.1164994027
113,9134 0,00401956 0,05566730786	113,283106 0,2370667612 0,4276627141	76 113,85 113,980(8) 0,012318(604 0,00747913622	82,9731 0,01674165 0,164721455	83.054656 0.003040971028 0.06636182289	24 83.11 83.2266 0.014067316 0.1426107328	151-76094 0.0000620836 0.0000620836	150,44345 1,758734502 0,8740528431	76 151.77 151.00025 0,00652056235 0,06520556633	1050,8078 1,23678641 0,1007209674	1034.01 150,4061 1,322343904	76 1051,9147 1,01060809 0,0855661pep4	174,71408 0,0001705094 0,1405579418	174,65863 0,0847217449 0,1863637403	76 174,36 174,8609 0,0056115001 0,0428105008
116,1687 0,058211212 0,007258896	114,25329 4,522895424 1,820913486	77 116,41 116,04016 0,1367668324 0,3176074630	83,19184 0,004042514496 0,08163521499	82,96896 5 0,0077042816 0,3405555688	83.38 83.2865a 0.00048196505 0.00773240452	151-E3858 0.0000020164 0.0000021764942	151,97948 0,0194480916 0,09194968072	151,04725 0,0071602729 0,1266660963	1065.0735 2,09236225 0,136911748	1014,125 5,73625 0,2266878159	77 1086.52 1085.7838 0.58911936 0.07140423278	174.75892 0.0479800464 0.125210036	172,9348 4,14203934 1,782170829	774,97 174,97 174,92786 0,0003747078 0,00157959536
115,724365 9 0,00024445325 6 0,01350672648	1,742558404 1,742558404 1,140539138	79 115.74 115.59223 0.0218358729 0.1276740971	83,85744 0,0001577538 0,01407536741	83,5849 0,58128201 0,3300306454	79 03,87 83,86738 6 0,000068644 0,000123862189	151-56293 0.007327549 0.01761150415	151,71347 0,0710382408 0,1753717594	78 161.96 151.86946 0.0074691716 0.05694170286	1084,7571 52,5040,6841 0,6766701812	1064,9701 290,350190 1,589528083	78 1072,01 1084,4442 57,2413,284 0,7057853418	174,06596 0,0011094026 0,0195684887	173,6821 0,1856800241 0,2474524571	78 174,08 174,18761 0,0007623121 0,01585901284
115,47142 5,0023600164 0,0420533341	114,8153 0,81648209 0,7831644321	79 115.02 115.39757 0.0148891048 0.1059816482	84,743095 0,01874297902 0,1812004128	83,75031 1,276199496 1,330628013	79 84.88 84.68391 0.0346254881 9 0.2192368255	152-46006 0.0000525256 0.000322555257	151,63869 0,6910763161 0,5452285698	79 192,47 192,22714 0,056,000,0716 0,1562,8379,05	1061,7424 2,39506576 0,1455462512	1043,050s 0,44523025 0,05296494841	79 1061,200 1051,8438 2,00140444 0,1360118124	173, 19362 0,0102819044 0,071427746	173,44829 0,1431033241 0,2185762988	78 173,07 172,97783 0.0034863000 0.00328590001
113,11656 0,021479536 0,129733576	114,97796 1,227575362 0,9807669629	112.87 113.97227 0.1618211529 0.3590856865	83,40000 0,0000000000 0,0120002NF88	83,72925 0,1515155025 0,467062635	80 83.34 83.53642 0.0385905164 0.2358851452	153.40787 0.020209069 0.09206081317	152,27843 1,82,1980545 0,82,9417128	80 163,20063 0,0872727989 0,1689166627	1039,7139 1,7928921 0,128583647	1062.753 236,329129 1,481905341	00 1037.38 1040.4810 0.42178361 0.2590128979	189,68906 0,0437060836 0,1233537861	172.3274 8,10768976 1,680080245	90 100,48 170,4170 0,87590376 0,8552276191
113,33112 0,000000444 0,01665437406	113,500% 0.4810027336 0.8300926334	81 113.26 113.56776 0.0014273284 0.00333059256	84,19457 0.000103649 0.0291101(Q313	83,6366 0,26452226 0,6336362837	61 64.17 84.09477 0.0096505659 0.009677803847	533-7161 0.0667953921 0.1676164000	152,846% 1,2825686 0,7297785263	18 BRB90021C0	1035.755 0.190999 0.04217454665	1017,880 137,3162112 1,130914811	81 1030,17 1030,2802 0,01214694 0,01063532046	171,78318 0,0078377124 0,00083090000	972,17723 0,1070794729 0,1904190606	81 171,85 971,73932 0,0122500634 0,08442500436
111,59461 0,5460574521 0,1919032009	113,267815 2-213690474 1-331020768	82 111.78 112.07385 0.0953478225 0.2628024477	84,24101 0,000,900,001 0,002,807,9420	83,6387 0,17749309 0,5	82 84,267 94,267 9,0001843524 9,0152149113	154.33679 0.1791057041 0.2734021349	152,0988 3,10182544 1,13902018	82 154,70 153,97249 0,620295049 0,620297548	1027-9598 8.35094404 8.3519124548	1032,9075 61,42640825 0,764581931	82 1025,077 1027,9778 8,465,30684 0,2836684324	170,93132 5,5140849424 0,68339322126	172,4453 1,94686209 0,8157263958	82 171,06 171,0566 0,000,044966 9,100,057,0656
111,88597 0,860333489 0,230453469	113,141464 2,884979743 1,826798277	83 131.44 111.72295 0.0798993403 0.230548897	81,84905 0.591994225 0.3489776517	83,82802 8,49240592 3,142599501	83 81.08 82.00738 6.860522121 1.143736281	195.40519 0.5116621361 0.610258213	154.07283 5.70998029 1.528057778	85 160.40 166.12664 1,7490549664 0,850056711	1019,1618 53,46389161 0,7228125688	1023.53186 134.1856524 1.153296372	83 1011,8732 1019,8732 80,11211024 0,7982249103	189,95856 0,0231729560 0,020400000	172,10167 5,297694789 1,356518257	93 199,8 170,0804 0,07582416 0,1601304635
111,96212 0,0687085044 0,235607637	112,03466 0,8397253156 0,8600716776	84 111,580 0,000,000 0,000 0 0,000 0 0 0 0 0	\$1,772484 0,000001100 0,000001100	82,65194 1,127716364 1,301596563	81.89 81.80000 81.80000 0.0710650336 0.331297903	156,05006 0,0726676036 0,1727924718	155,88905 0.2873496025 0.345659543	94 194,048 174,04803 0,2503913669	1021.91565 5.362234923 0.327113674	523,90314 5,30445306 0,2258866222	04 1019.4 1021.017 2.0081724 0.1389855764	199,80902 0,0780977604 0,00983452016	171,76791 4,52600768 1,254265074	84 186.04 176.06674 0.1821670276 0.2515882267
112,3223M 0,014,860,60736 0,10,900,20456	111,61229 0.3454147984 0.8238146168	112.2 112.389404 0.005877487522 0.1988002002	82,75725 0.0005175425 0.02748248360	82,235985 0,3959623202 0,8571816864	85 82,78712 62,78712 0,0005234944 0,02753952846	154.05022 0.0024780454 0.002303989	154.59274 0.3427927076 0.3197534069	85 154.1 100,01331 0,0021911561 0,1860415315	1003.2922 0.22741284 0.65540108033	1030,0626 7,11602276 0.2582887908	102,72 102,192 102,192 0,303,064	169,20949 0,0387902801 0,1180344359	171,18591 4,734584329 1,26744453	85 169,01 169,0056 0.345964025 0.3954441749
112,558896 0,007902468516 0,07907469771	112,74497 0.155605000 0.3890677016	88 112.42 112.4869 0.004694278 0.00671873332	82,49458 0,0000009764 (0000001180144	82.647 0.024649 0.1903261001	85 82.49 82.53188 62.63188 62.6301327721 6.172087283	150-20/92 0.0001459264 0.0071899119665	153,09627 0,1452046129 0,2515269009	86 153,57 153,49613 153,49613 0,05717197369	1044.901 0.109561 0.011678601	1004,8887 93,72758598 0,9258215829	86 1044,57 1042,9347 2,47429699 0,156552457	169,41827 0,0090703728 0,00903624144	170,23172 0,6312333684 0,6384597212	86 109.32 169.63007 0.1018026648 0.1864428033
112,122234 112,508896 112,802096 .0.01489000779 0,00790048891 0,002930279 0.1090020896 0,07907489771 0,04732119908	112,70519 0.0340830391 0.1378803843	87 112,66 112,72368 0,0302342544 0,1544913372	84,00216 6,0016022666 0,0000030602	82,51709 2,699163260 1,662128601	07 84.16 83.0097 0.0745453809 0.0244177767	134,70381 0,0112783161 0,00009376000	153,21136 2,555681622 1,553652929	87 154,81 114,40607 0,080507789 0,2025002719	1069,4413 0,00275769 0,000641544531	1037,8066 141,0037303 1,131572929	87 1045,38 1047,7178 2,78390425	159,43005 6,0036060025 9,00545492118	170,06133 0,4779371699 0,4081773832	87 108,37 186,710,06 0.159575236 0.2043210908
112,081635 0,003666173225 0,04606033295	113,24944 1,46703914 1,086494193	08 112.83 112.83613 0.0110523169 0.0018629566	85,00473 0,0507165729 0,3643083421	83,42181 3,26951076 2,121541711	86 85.23 84,9999 845754491 8,2817083187	136-23105 0.0320231025 0.1151470304	153,77524 2,672440258 1,051901422	88 185,41 154,91764 0,3425186516 0,3168779358	1051.4989 0.22193521 0.04478394089	1037,2386 216,9840042 1,400266068	86 1051,87 1051,2073 0.37540129 0.05604310079	172,59755 0,0002490025 0,04155804045	170,94308 2,962252906 1,000127411	98 172,67 172,49923 0.0533534084 0.1336537805
112,081635 112,002775 112,00180001010,02050010000000000000000000000000	112,387726 0,87587861108 0,2808316411	89 112.07 112.65035 0.0049511225	85,27403 0,0900677049 0,0664734751	84,180854 1,65732971 1,634781401	85,56 85,2825 0,08850025 0,5476275605	196,37857 6,1320813649 6,2316660618	154.50706 4.582047203 1.404046191	89 156,74 136,00600 0,5392111761 0,4684892178	1048-0739 0.49405621 0.04637384431	1942,7184 36,64607296 0,577209403	99 1040,77 1048,0045 0.56600025 0.077299020479	171,65154 0,0023483716 0,002368388	172,36209 0,4383831881 0,3856086197	86 171,3 171,04261 0,0588596121 0,1412587769
112,8908 0,00008464 0081486425	113,9097 1,011979041 0,8110274679	90 112.9 113.016464 0.013633833 0.1031567759	85,10058 0,9622103944 0,2622310850	85.112176 0.06650773063 0.2786467487	90 85,26 85,26215 0,0077178225 0,1028291164	153-91026 B-0000000.0 0.000188629994	154,15607 0,0600004449 0,1596791502	90 180,91 194,011 0,011881 0,07082363840	1050.5947 0.60250209 0,07594272059	1947,5416 14,61090225 0,3690902433	90 1051,00 1002,2427 0.7284 1529 0.08100411512	172,22296 0,0022127616 0,00273597319	172,14243 0,0162741049 0,07405235967	88982821007 2748711 0.0000283
113,769745 3,0009153850021 0,00088111599	113,74327 0,003143009 0,04980001611	91 113,2 113,78694 0,0007705694 0,0011476074417	84,50109 0,0056969811 0,000007799	\$4,847.19 0,0247030901 0,185608329	91 84.00 84.07082 0.003878724	154.08784 0.0077158666 0.00703866194	155,02405 1,050522502 0,8655519481	91 154 153,1428 0,00327184	1050,3416 1,2728091 0,1006791403	1067,0171 0,20511941 0,2428383818	91 1907,47 1006,646,0 0.3656,644 0.066743963,0	572,18614 0,002900896 0,00127305192	173,26784 1,017969686 0,9084310766	91 172,22 173,23671 0,0001746241 0,007718873780
114 23636 0,0018510086 0,03856177809	113,54778 0,5361481284 0,6407240302	92 114,28 114,267896 0,001043548416 0,02806741337	86,293724 0,3096429892 0,640802015	85,84518 5,451591202 1,461242072	92 86.05 86.223976 0.2919060486 0.750810683	152,42016 0,0144364266 0,07679344282	154.07048 2.48640743 1.63862261	92 152,6 152,63707 0,0039601649 0,04126587377	1048.7871 10.2272941 0,2987109309	1090,4126 464,2504405 2,010018657	92 1072 108,473 12,436729 0,32901194	173,84987 0,0144312109 0,08005213543	172,14593 3,312854805 1,04619762	0.000000000000000000000000000000000000
115,69027 0,0527758720 0,1861797790	114,57545 1,60797912 1,509996128	93 115.62 115.45949 0.220439401 0.4050293308	86,08983 0,0841986299 0,000029973	85,919014 0,2125090922 0,5236721483	0.001722100.0	150-921 0.03630241 0.1252903323	152,02885 0.1608213225 0.2616252527	93 153,300 153,300 0,00442225 0,0434633827	1078-0722 27-02747864 0,9630057873	1061,0854 530,0225412 2,129279599	93 1085,09 1077,8426 52,53925425 0,667916847	175,77425 0,4 109030025 0,3605300266	17,22075 17,5634355 2,376856385	93 175,47765 0.8890236225 0.5341514368
115,115908 0,009021241800 0,07221701389	114,43434 0,694334339 0,694334167	94 115.2 115.311815 0,01245790823 0,09688852000	85,64051 0.0389054901 0.2201734220	86.85127 0,0004521129 0,02478154401	94 85.40 85.707986 0.01488766023 0.1427586188	155.41853 0.0355435909 0.7230272842	553,0722 0.02490084 0.7028624447	94 0.000363000 0.0003630	1078.8418 8.63201924 0,2721109888	1067.3336 154.512873 1.153879249	94 179,711 1076,8486 9,8656669 0,2590005152	174,43564 0,0108910000 0,05099149162	174,38959 0,0236231681 0,0981150888	94 174,54 174,3668 0,6900098224 0,0990001982

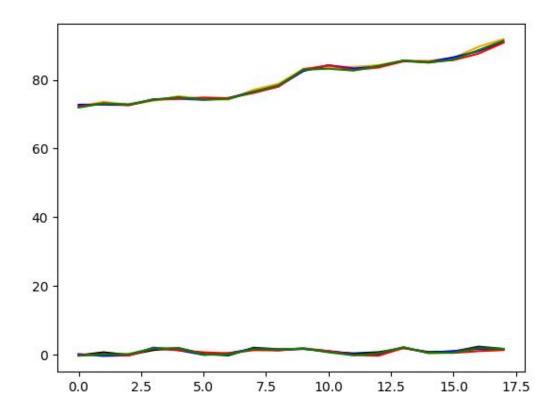
113,21146 0,0026481316 0,04547543302	114,13149 0.9437928201 0.8585100742	95 113,16 113,36504 0,0508430216 0,1598686583	85,37913 0,8198443569 0,1647217025	85,29717 0,0490532099 0,2605580336	95 85,449 0,0654261 0,06547708136	153.16033 0.0421604089 0.1342464858	153.62555 0.4563678025 0.4416802877	95 152.05 153.16376 0,0236421376 0,100626548	1071.838 2.585284 0,1604008081	1064,1564 89,42788286 0,8750288229	95 1073.59 1071.5504 4.03840216 0.1671902828	174,25833 0,0084033889 0,0525781474	174,23968 0,0121682961 0,06326628502	95 174,35 174,20189 0.0219365721 0.08494981359
112,95963 0,0008775069 0,0262491141	113,839935 1,123250227 0,938904146	96 112.86 112.9028 0,0126087984 0,0126087084	85, 97568 0,0103609604 0,1450926016	85,18499 0,0992313001 0,5684327485	95 85.5 85.52943 0.0006498849 0.02974289008	151.81646 0.0395150116 0.2082244224	152,83272 2,05268698 0,945689769	96 161.5 161,8899 0,15202201 0,257369736	1069-5211 1.76597521 0.1240976794	1053,0883 60,24398689 0,7248187344	96 1070,85 1069,7333 1,24701889 0,1042816454	174,90833 0,0107474689 0,05023681505	174,10986 0,6102520196 0,5143363236	95 175,01 174,87969 0,1091046961 0,1097377864
112,58586 0,6000343386 0,00526187422	113,86377 1,666816213 1,158683141	97 112,58 112,74139 0,0209407321	86,39231 0,0138509361 0,1376330264	84,854254 0,3791308217 0,720074845	97 85.55 85.48097 0.0003621400 0.02227470705	152,68639 0,0244578321 0,1025508167	151,71933 0,6094456489 0,5119147541	97 152.5 152.5 1805 0,000258025 0,0118360657	1087.8423 1,0357.1329 0,6862.1359368	1083,2401 31,43219661 0,5249424821	97 9088,86 908,1647 0,45803000 0,06317940223	174,93237 0,0060264169 0,04435748529	173,83836 1,14827001 0,611758328	97 175,01 174,7552 0,08452304 0,1455916805
112,37721 0,0002961841 0,01531683873	114,10839 3,058967592 1,566060876	98 112.32 112.5285 0.025753984 0.1504146985	85,29672 0,0100967884 0,1200367881	85,253914 0,021341194 0,1710938899	98 85.47305 0.0053365025	152.86384 0.0152960496 0.08050034579	152,63206 0,0391802436 0,1236164562	98 152,9539 162,96399 0,0153598449 0,08109010011	1065.0894 0.57851236 0.07130088506	1082,4748 11,39332518 0.3168862129	98 1085,36 1084,3607 1,00881089 0,08413144439	170,7443 0,03038049 0,1021867855	174,06238 12,19671606 2,04747611	98 170,57 171,13673 0,3234538129 0,3334280907
113,148544 0.002308515036 0.04292130858	113,505455 0.184393757 0.3584624845	99 113,1 113,513576 0,04561470778 0,1660391963	85,60103 0,0118744609 0,1271380236	85.088585 0.3861508022 0.7250204177	99 85.71 85.903995 0.02131740002 0.1703478541	153,24454 0,0131194116 0,07479919023	152,83101 0,248910201 0,3256803801	99 153.13 153,02905 0,0101909025 0,06502437798	10:56,3:58 9,70761744 0,07834353009	1065,6907 30,14778649 0,5178628504	99 1060,2 1099,999 0,581201 0,5862741747	170,67883 0,0059028489 0,04503516999	172,86705 5,1395,5703 1,328668699	99 170.6 170.79704 0.0388247616 0.1154582415
112,424965 50,000623251225 0,02221085409	113,28882 0,7900000024 0,7907851246	112.4 112.53911 112.53911 0.0193515921 0.1237835452	85,58584 0,0154157056 0,1446436771	85,31696 0,1636717136 0,4716152123	100 85,75 85,60311 0,0136492498 0,1392925905	154.13803 0.0382217803 0.0323408004	153,03787 1,004264537 0,0505647884	100 154.04 153,81505 0,0506025025 0,1460334978	1054,7413 1,46095569 0,114465947	1063,6796 59,74671616 0,7320042563	105,435 105,6763 505,6763 0,07491169 0,02591976787	171,11682 0,0013657124 0,02162203403	171,8944 0,63612736 0,5344867896	171,05 171,21472 0,0181494784 0,07674678513
112,40168 0,0016875864 0,03856105378	112,590156 0,052,969,73802 0,2048,727366	102.34071 0.000387023 0.01711463154	85,43027 0.0120406729 0,1282701678	85,160965 0,1436075312 0,4431064873	101 85,54 85,472850 0,004508451021 0,07849644073	153.51718 0.0094439524 0.05334245861	152,96985 0,1850290225 0,2863741364	163,42 163,42 153,48161 0,0017313921 0,02712162691	1052,286 1,240996 0,1057528006	1063,7733 107,6053629 0,0847446364	101 1083,7 1037,7151 0,090,29901 0,0299129079	169,40448 0,0394362916 0,1036904688	171,65956 5,902713203 1,435649707	100.23 169.73381 0.3178817161 0.3331819688
112,437836 0,002288282866 0,04258250558	111,89942 0.2406687364 0.4364879091	102 112,39 112,48638 0,008/172138 0,008/0752738	85,82588 0.0154654624 0.1646422339	85,44039 0,2697023521 0,5929144852	102 85,96 85,93846 6,030178323316	154-34946 0.0056922916 0.00447974068	154.41411 0.0269320921 0.1063922204	162.75 154.75 154,172 0,00064 0,05056728094	1,75483059 0,123433430	1067.9463 232.9606377 1,422247277	1073.21 1073.21 1071.4113 3.23532169 0.1676000037	172,22891 0,000965081 0,01804629008	169,82165 5,946039402 1,416663683	172.26 171.59675 0.4372516625 0.3538574997
113,34545 6 0,0012567025 0,00128586397	113,02813 0,0794508988 0,2487800388	103 113,31 113,36881 0 0,003,048921 0 0,048960,08594	88,194048 0,02432165012 0,1806068027	86,12797 0,0492973209 0,2571279576	103 86.35 86.325485 60.000600985225	1573138 1.38450522 0.7424127705	154,51831 13,48130746 2,316689821	158.49 158.99038 158.99038 2,370429744	1067.7165 14.46292289 0.348294122	1067,8517 560,1884249 2.168379874	1091.52 1097.0623 1087.0623 19.87108929	172,24156 0,0901334025 1 0,006706146755	0.3597240529 0.3482378215	172.23 172.26517 0.0011000449
0.1295968279	112,84854 4,859,75972 1,91621947	104 115,05 114,70445 0,086584449 0,2247457627	86,86262 0,0517016644 0,2610262886	85,01554 1,197642692 1,256411434	104 87.11 86.69079 6 0.0480530241 0.2598473424	158.95458 7.482522576 1.591683575	156,90803 22,10005914 2,907278912	161.7 161.7 168.27824 113.7064416 2.116116285	1091,4094 18,92772036 0,3970394977	1081,0887 210,8342967 1,340740673	104,76 1091,6017 15.43154089 0.3885000365	173,05623 0,000680129 5 0,01516022084	172,21526 0,6636012676 0,4706663236	173,02304 173,02304 0,0000484416
0,2775614556 0,4529012263	114,57485 3,06061523 1,50876816	116,33 115,8713 0,43386660 0,5662358985	87,725385 0.215857805 0.526522789	86.68722 2,256347726 1,7040254	105 88.19 87.743904 0.1990016412 0.2006051287	160.4819 3,262.2561 1,1127454	160,40668 3,961373702 1,224067685	160,49 180,04155 5,94407403 1,90883119	77,39971249 0,7920723415	1090,4824 390,1337216 1,785803709	1110,29 1110,2418 93,08778324 0,0088801764	0.0007745089 0.01586285714	172,79088 4,890211174 1,382354286	175 176,7453 0.05487205 7 0.1455428571
114,85475 0,0029675625 0,04769163763	114,51029 0.07(0805241 0.2453919961	114,8 114,78777 0,0001400729	87.88702 0.1544332904 0.4451817898	87,23169 1,099853860 1,187489059	106 88.28 87,8841 0.147904681 0.4359479384	161.80142 2,784159216 1,020728516	160,37054 9,606652292 1,896042087	163.47 161.51704 3,814052762 1,154680167	1106.0856 66.08714436 0.7269111148	1104.8823 87,00580729 0.8371681850	100 1114,21 1105,300 79,34034289 0,7983247987	174.42697 0.0059243809 0.04414083109	173,36076 0,9765957778 0,5673674391	174,25 174,22528 174,22528 0,0155650784 0,07153427015
114,192314 0,001044194398 0,02830589648	114,401218 0,66818467622 0,2112868601	107 114.18 114.24967 0.0080798198	87 88121 0.1147789541 0.384028565	87.80803 0,1696963246 0,4860340354	107 88.22 87,87128 0.121508841 0.3852731607	162.51398 1.79188232 0.833277788	181.51112 5,377204464 1,416418422	107 163,83 162,9213 2,27617599 0,9208036092	1107.2665 30.50505225 0,4963649625	1104.0006 17.26531026 0.7898815192	107 1112,79 1107,3018 30,12383726 0,460219746	174.3558 0.0008564 0.01479051616	173,246 1,175056 0,6218002124	174.03 174.041 0.05109081 0.1290818276
112.92966 5 0,000599155 0.06285726942	114,17843 1,706750345 1,18746434	108 112.87 113.10084 0.09671104626 0.2755240542	87,81884 0,0404653456 0,2290594366	87,815704 0,000018465816 0,004891824186	108 87,62 87,701935 0,01393034422 0,1344397632	163.09028 1.176617478 0.0806894871	162,72487 2,117403317 0,6863016202	104.16 164.16 162.97785 1,445645522 0,7323546	1106,4325 13,7455925 0,3339688871	1107,8919 5,99319381 0,520521736	106 1110.14 1507,4154 7.42344516 0.2464235045	174,2857 0,00001849 0,002487152447	173,40111 0,7901254321 0,5100063113	174.29 174.19547 0,0090259209 0,05423719089
110,51289 0,761,9878961 0,7966595038	113,71631 16,61930322 3,71790405	109,04 109,04 110,70629 1,134842784 0,8716253182	87,87028 0.0439024784 0.2381017257	85,09888 0,0003564544 0,02143505904	97,880,91 0.0096368281 0.2266031617	163.3738 0.66263644 0.5031668687	163,20337 0,9902713569 0,6066810231	104.2 104.2 183.02629 1,424943564 0,7266853837	1108-8323 10.38389029 0.2883889005	1108.861 10,169721 0.2867876813	1112.05 1112.05 1100.6233 5.88887289 0.2182186053	175,24309 0,0013023481 0,02105773619	173,9064 1,66977696 0,783660429	175,28 174,88899 0,168629201 0,2344876789
110,38686 0,9722856996 0,2441298649	110,74081 0.3730886561 0.5546263507	110, 110,13 110,51073 0,1449653239 0,3457991159	89,0215 0,33466225 0,6456473214	88,19758 1,900781880 1,565200803	110 89,8 86,927886 0,4616028162	162.65073 0.2393851329 0.299308544	163,01129 0,0163662641 0,07889642724	100,14 162,37334 0,587787856 0,4609399289	1123.9948 44.29168704 0,5888171871	1112,8902 322,2912208 1,58765312	1130,66 1123,3888 1123,3888 52,72503544 0,6422146553	176,74872 0.1154720394 0.1927155684	174,4105 7,179,720,25 1,5130,72449	177.00 177.00 178.40152 0.4740047104 0.3887740897
110,87402 0,0378437804 0,1752981568	110,55401 0.0158734801 0.1138326708	111 110,68 111,0317 0,12360280 0,3177629201	86,2212 0,01858914 0,1407638144	89.22771 0.7703748441 0.9934465195	111 88.05 88.16809 0.0327284281	163.33478 0.2854722576 0.314458348	162,05453 3,223,712521 1,036,691038	111 163.85 163.0212 0,88690844 0,6059285017	1125.5754 26.28152516 0.4532238688	1121,9515 76,53625225 0,7737242416	111 1130,7 1120,6123 25,88469129 0,4499602016	176,07712 0,0127418944 0,06408720018	175.82582 0.1328270724 0.2056973154	175.18 175.94276 0.0611276176 0.1400257847
111,31784 0,0102919896 0,1147944969	110,786376 0,1613016374 0,3612561443	111.79 111.79 111.34716 0.0248900000 0.141343646	89,699505 0,281424945 0,588523408	88,595116 2,300000573 1,713871755	112 90,14 89,02839 0,3740867901 0,6786112046	167-20833 2,078412389 0,8548295286	165,08232 12,72834068 2,715434331	112 108,05 168,274,48 5,940,9527 1,408560252	1132,9973 37,24294729 0,63574792	1124,3176 216,5193496 1,297726275	112 1130,1 1131,9717 50,81286880 0,8257838133	178,36363 0,5572174609 0,4167809001	175,81625 10,76301406 1,83347292	112 179.1 178.0411 1.12126921 0.5912339475
111,73263 0,0039100003 0,05590534342	111.751826 0.008437524736 0.08225664806	111.5 111.7648 111.7648 0.013151502 0.1026954411	89,65956 0,1999873036 0,4886345282	89,33282 0,5895661024 0,8514781378	91,1 90,1 98,04483 0,0070887049 0,0060721421	167-94714 1.37580058 0,8935075088	167.58025 2,370830062 0,9104482025	113 160.12 187.45995 2.755796003 0.9815811258	1131.3202 21,82054004 0,4093241889	1131.301 21,79561 0,4110143754	113 1135,07 1130,092 1130,092 28,922884 0,4734279865	178,70703 0.3067788209 0.3064737263	177,79146 2,156609732 0,8192234743	179,28 179,5866 0,637787666 0,4080828076
113,21449 0,0002099801 0,01280035338	111,5084 1,50855406 1,116254417	113.2 113.074815 113.074815 10.1672136821	89,63128 0,1356544384 0,409688888	89,13614 0,7462540986 0,9505444444	114 90 89,70002 0,293911111	162,43852 0,0046949904 0,04215991378	165,09079 7,402698224 1,675672846	114 162.37 162.10115 0.0722600225 0.1655785167	1137.8635 31,54507225 0,4811874663	1129,8653 191,3589241 1,206855708	1143,5 1143,5 1130,1566 28,5205306 0,4071097550	178.22801 0.0847813301 0.1311182183	177,26693 1,425903165 0,5900967164	178,46 178,16574 0,085889476 0,1048894904
112,78325 0,0000105625 0,002681716616	113,50616 0,8625420458 0,7325412307	115 112.78 112.65612 0.0033770344 0.05153385662	90,689084 0.5184230403 0.7889678649	80,8381 3,13602901 1,034177402	115 91,51 90,84214 0,5898089796 0,5381836044	162.42466 0.2504721156 0.1361672817	162,2436 0,12702096 0,2191681919	118 162,0 162,9926 0,0430356025 0,1275830268	1154-5448 92,46014801 0,8259258666	1138,0638 788,3804544 2,413431143	1164,16 1164,16 1154,159 92,563641 0,4264327527	178.22801 177.01687 0.0547813201 0.0002455469 0.1311192165 0.008653107345 0	177.57503 0.455855009 0.3813728814	178 178,42008 0,7010,697177
114.22544 0,0019855996 0,00899536186	113,19027 1,165616873 0,9448636729	114.27 114.27 113.945595 0.105238904 0.2538604103	91,21601 0,4637473801 0,7410119695	90,34641 2,404326348 1,687257880	91.0 91.2406 91.2406 0.43460636 0.47175180424	166-26589 0.003897681 0.01178345865	162,54912 13,59651277 2,729093233	196,25 165,72088 0,2801690781 0,3183818549	1185.5133 113,5652549 0,909050846	1154.3204 477,4050202 1.85789084	116 1176,17 1160,1445 121,5616502 0,9374070075	177.0602 0.000411084 .0114608688	177.42224 0.1461074176 0.2168060089	175.04 175.04 176.06008 0.0035545444 0.00357600542
114.37273 0,000074529 0,002385989695	114,19214 0,0325498198 0,1572615196	114.37 114.37 114.28951 0.0084788401 0.07037884708	91,27865 0,2900598225 0,5805774341	91.254616 0.3190590075 0.8157525594	91,82 91,82 91,30046 0,2396206304 0,533130037	185.4025 0.0010400825 0.01960172341	163,3196 4.22878096 1,243514543	165.37 165.37 165.10754 0,040852516 0,1587107688	1164-2566 49.46871666 0,6004632279	7152,4054 78,91034896 0,7584458182	1171,29 1184,7771 42,41788841 0,8580450444	174,29787 0,000037309 0,04489536002	175,44771 4,962691844 1,278676386	174,22 174,48914 0,0724283396 0,1544828378
112,92812 0,0074166544 0,07832045974	114,21853 1,385,35225 1,230249911	116 112.07 113.0719 0.066259099 0.2102002899	91,7751 0,30791401 0,6003664289	91,75923 0,3057753929 0,815164772	92.33 92.33 91,879234 0,2031806866 0,4882118486	103-54834 0,0061371556 0,04734382941	\$ 4,762127773 5 4,762127773 5 1,318807035	165,47 164,8878 0,3078166864 0,3812540037	5 1174.0376 55,6489876 0,8854010623	1175,0717 49,96086485 0.5979241037	1182.14 1182.14 1173.672 71.707024 0.7163289153	7 171,42995 9 0,102369025 2 0,1863645604	175,79476 21,94668256 2,737858687	171.11 171.11 171.98279 0.7617623641 0.5100753801
114,62147 0,008(37380) 0,0711279022	113,21719 1,749626286 1,154889122	114,54 114,57 114,37137 0,0264360760 0,1472236773	93,12812 0.8665375744 0.9696651705	2,61226038 2,718328726	94.26 94.26 93.09074 0.3094640476 1,000468913	167.25943 0.2064015249 0.04814748418	163,75284 12,80771687 5 2,143635712	118 167.34 166.59888 0.1625218590	1179.8605 62.40210025 0,6651674432	1188.5736 1,02738496 0,08535147996	1187.56 1187.56 1180.0019 56,77251781 0.8286008576	0.0059133316 0.137868512	6,361301952 1,470573144	171,51 171,3064 0.16713296 0,2311235496
7 113,61259 9 0,0052693061 2 0,56393341554	9 11329397 6 0.0605307639 2 0.2166901632	9 120 4 113,64 7 113,526,6 9 0,00013,00736 3 0,01007574423	2 03,2667 4 0,50679689 5 0,7594761409	4 93,703064 8 0,0470612281 6 0,2328795571	120 93.92 93.286376 0.3969488774 0.6726127768	3 165,58546 9 0,0698105516	4 164,81330 7 4,781263292 2 1,310917296	9 120 4 160.8 6 180,7004 6 0,00894916 7 0,0867146283	5 1180,3457 5 37,67872689 2 0,5173838534	6 1188,6165 6 4,56463225 8 0,1830704605	9 120 6 1186.48 9 1181.6423 1 23.40334129 6 0.40773544665	6 168,95026 6 0,9936149676 2 0,58906908528	8 172,00665 2 16,37537622 4 2,406293579	1 167.56 1 167.56 4 169.5491 6 2.5505681 6 1,000656109
0 110,99141 1 0,0329095681 4 0,1637126813	7 112,10583 9 1,679434565 2 1,169506362	0 121 4 110,81 6 111,36626 6 0,320756676 3 0,5200433174	7 92,36589 9 0,1253938921 9 0,3616306252	4 92,95885 1 0,0523725225 1 0,2487651499	0 121 2 02.74 6 92.443184 4 0.08000073786 8 0.3200617576	4 163.53636 6 0.0969966496 6 0.05111844518	9 165,87857 2 4,237710445 6 1,258140014	0 16,462 4 163,52042 6 0,0000161784 3 0,0000805305	7 1173,4907 9 15,0496849 4 0,32940808	5 1182,0621 5 22,01690241 6 0,7966238285	0 127 8 1177,37 3 1176,7465 9 2,626,02625 6 0,1376372763	6 168,04195 6 1,149076803 6 0,6420015872	5 165,84624 2 3,520276538 0 1,12369886	121 105.07 1 165.7626 1 2.560135436 9 0.9520171286
113,3244 0,0055413136 0,0657306843	5 111,29461 5 4.163620252 2 1,806083885	12 113.2 113.0923 0.024847216 0.13918763	9 94,29794 11 0,5070294436 2 0,7494579018	5 93,20641 5 3,250340068 6 1,800368468	11 122 4 95.01 4 94.20128 6 0.6640903976 6 0.6012158616	6 163.68567 6 0.0002053489 8 0.00875381786	7 164,76045 5 1,124564203 4 0,5478006562	127 163,1 163,85216 0,0014318868 0,0223115456	7 1178.1348 9 16.68885904 6 0.3465632811	1 1181,7227 1 0,24730729 5 0,04205402860	11 122 17 1162,22 15 1179,0021 15 7,589,6641 10 0,2298979085	5 168,18488 3 0,5688438144 2 0,4608839472	4 166,88269 8 0,2965487361 8 0,3265888491	1 127 17 167.43 8 168.795(6) 9 1,865846722 8 0,8158594553
4 111,620674 6 0,00060014278 3 0,0262660985	1 112,08942 2 0,201978394 6 0,4028287801	2 123 5 111.86 7 111.7393.6 8 0.00793072225	6 0.2305754841 8 0.2005754841	1 94,42536 8 0,0272448036 8 0,175111394	2 123 11 94.26 6 93,048335 6 0,488738031	7 162-5136 9 0.04562496 6 0.1315270836	5 164.42118 0 4.085087746 2 1.24455885	2 123 7 102.4 6 162.37445 6 0,000658025	8 1177.557 4 16,255089 1 0,3413497471	7 1183.2943 9 2,90463849 9 0,144237849	2 123 2 1181,50 2 1179,5350 1 1,2164,5156 5 0,17379,27842	8 168,27487 4 0,2448963169 2 0,2049617225	9 168,36217 1 0,327376589 1 0,3410238699	2 123 3 197.78 6 168.8838 2 1.217447424 3 0.6576349832

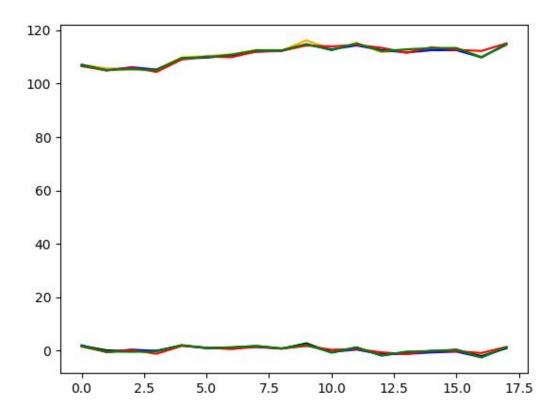
Appendix C - 7 day forecast preliminary results











<u>Appenaix D</u>	- Links to	Source Co	<u>ode, Marke</u>	et Data and	<u>a Kesuits</u>
	=				
			_		
		-			