Forecasting Commodities Prices using LSTM Model
ML Singapore! Group 59, National University of Singapore

Teo Yue Yang (A0171771A), Goh Yin Hao (A0155167B), Ng Tze Xuan Shaun (A0155911H),
Lim Ming En (A0155160N), Ong You Sheng Aaron (A0168284U), Chew Wei Ting Fionna (A0172148E)

Introduction

From January 2003 to July 2008, the prices of most major commodities surged at an unprecedented rate: wheat by 120%, copper by 363%, aluminum by 100% and nickel by 138% (Alquist et al., 2019). Many attributed the rapid growth to the increase in global demand for commodities, which in itself reflected a shift in the global economy. The driving forces behind the demand were the emerging economies in Asia, largely China.

On hindsight, the simultaneous rise in prices across the commodities was telling of the changes to come in the global economy. Furthermore, the relationship between commodities and economic growth can be drawn since the 17th century (Harvey et al., 2017). However, effects of changes within the commodities markets are not limited to the economy. Fluctuations in the commodities pricing can also indicate global disasters, consumer trends and technological disruptions etc (Onour and Sergi, 2011; Marvasti and Lamberte, 2016; King et al., 2015). This highlights the necessity to understand price trends in the commodities market and indices.

However, only understanding historical trends is equivalent to working in retrospect. This does not enable the consumers, companies or even countries to better prepare for the effects of the changes in the commodities market. Thus, this suggests the need to be able to predict upcoming financial trends in the commodities market. Such predictive capabilities can consequently empower individuals and organisations to anticipate and appreciate impending financial and social events.

Predicting the stock market has proven to be an onerous task due the nature of the time-series data and the interwoven tapestry of human behaviours, environmental factors and socio-politics (Miao et al., 2007). These elements makes the stock market, including the commodities market, highly unpredictable and volatile. In spite of the enhanced complexity, data scientists have come up with different approaches to surmount the task of predicting the stock market. One common approach is using machine learning to identify patterns from historical price data and formulate potential future prices. This has been proven to be potent in predicting time-series data including stock markets, weather forecast and radioactive decay (Heinermann and Kramer, 2016; Rasouli et al., 2012; Abdel-Aal and Al-Haddad, 1997; Patel et al., 2015). Hence, this paper proposes to use neural networks, an architecture for machine learning, to predict future changes in commodities prices.

Neural Network Models

Neural network is a type of machine learning modeled after the human brain. It comprises of an assemblage of interconnected nodes or units, known as artificial neurons that are organized into layers. Each neuron consists of an input, output and activation function. The input layer represents the input interface of the network whereas the output layer serve as the overall output platform of the network. Each connection between neurons are weighted, allowing different inputs to have distinct effects on the final output. The main bulk of training a neural network is to refine the weights and parameters in the activation function such that the final output of neural network resembles the data set. This is usually achieved by optimization algorithms like backpropagation.

Multilayer Perceptron Model

One of the most basic models of neural networks is the Multilayer Perceptron Model (MLP). MLP is a type of feed-forward neural networking model, with information strictly moving in one direction, from input to output. As its name indicates, it has multiple perceptrons that are arranged into layers. It will consist of at least 3 layers - the input layer to receive a signal, the output layer to produce a prediction, and hidden layer(s) in between utilizing activation functions (Skymind, 2014). The activation functions presents non-linear properties into the model that will be able to learn intricate mappings and predictions, with power beyond a simple Linear Regression Model due to the activation function. (Walia, 2017)

Long Short-Term Memory Model

LSTM Networks are a type of Recurrent Neural Networks (RNN). An RNN can be thought of as multiple copies of the same network, or modules, each passing a message to the next. This feature allows information to travel through time in the whole network, allowing data from a previous point in time to be used as input for the next time point. Hence we can connect past information to the present task, something that traditional neural networks are unable to achieve (Pascanu et al., 2012). This makes it extremely robust for analysis of time-series data such as stock market prediction (Nelson et al., 2017).

However, the usage of RNNs face a significant roadblock called the Vanishing Gradient problem. Information flowing through an RNN passes through many stages of multiplication, which can easily 'vanish' if the weights connecting the modules are at a value along the activation function where the gradient is close to zero (Hochreiter, 1998). During back-propagation, vanishingly small gradients will prevent the weights from being updated, resulting in the network not being trained properly.

An LSTM network can be viewed as a chain of repeating modules, similar to a traditional RNN. The difference is that each module of an LSTM network consists of four neural network layers, as compared to a single layer in a RNN module. Moreover, the chained modules in an LSTM network are connected by a pipeline, or a cell state, which helps to solve the Vanishing Gradient problem. Information can freely flow through time in this pipeline without modification. Although information can be added or removed to the cell state, it is regulated by structures called gates with the intention to forget or replace information to better suit the current prediction (Olah, 2015). Most importantly, these information will not be lost during back-propagation as they will not be excessively multiplied by any weights. The LSTM network can now be trained properly regardless of the number of modules or the size of its weights.

For this paper, we decided to compare the most basic neural network model, the MLP network against a more sophisticated model, the LSTM network for predicting the commodities market.

Data Extraction

Without loss of generality, we decided to use a single commodity to test the efficacies of MLP and LSTM models. To this end, we selected gold arbitrarily. We obtained the gold prices from the World Gold Council (WGC) via Quandl database. The extraction was achieved using Python and the Quandl module. The data format is shown in Table 1.

Table 1: Data Format for Gold Prices from WGC

Date	Price
1/1/1979	226
2/1/1979	226.8
⋮	⋮

Subsequently, the input examples were generated by taking the data, which consisted of 10,510 data points, and splitting it such that every ten consecutive price points are used as the input and the subsequent price after these ten points is used as the target output for this particular example. For instance, data from 1/1/1979 to 10/1/1979 and 11/1/1979 are used as the input and target output respectively for the first input instance. The next input instance will use data from 2/1/1979 to 11/1/1979 and 12/1/1979 as the input and target output respectively. These input instances are then divided such that 80% of the data was used for training while the rest was used as the validation set.

Data Normalization

Due to the volatile nature of the gold price data, we decided to normalize the data. Data normalization reduces the data range, enabling better speed and accuracy for machine learning (Shah-Neha-K, 2011). Conventional normalization utilizes either the Z-score, Min-max or decimal point (Shalabi et al., 2006). However, these traditional methods fall short when it comes to commodity prices market prediction. Since the training requires live tracking of the market, it will be arduous to formulate statistical normalization. Hence, we decide to approximate a normalization constant, N, to an arbitrary value. We then use $Price$ / N as the input data and target output to our model. The prediction of the model and the target output are then multiplied by this same constant before comparing them.

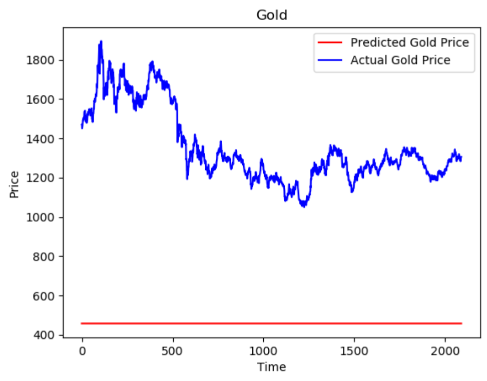

Figure 1: LSTM Model, N = 1

We varied a range of 1 to 1000 for N to observe the effects of the different normalizations on the training, while keeping all other parameters constant. From Figure 1 where N = 1, it can be observed that the model outputs a constant. This is likely due to the fact that gold prices range from values above \$1000. Without any form of normalization, these extreme inputs result in the vanishing gradient problem, where neurons venture into saturable regions of the activation function, where gradient is close to zero (Jansma, 2018). Thus, these neurons are unable to update their own weights. When this happens to all of the neurons, the resulting output of the model will always be a constant.

From Figure 2 where N = 500, it can be observed that the model's predictions are very accurate, with the exception of the first 500 sets of the validation data, where there is a spike in the actual price of gold. This 150% spike in price can be attributed due to the global financial crisis in 2009 (Roubini, 2013). As expected, our model is still able to predict the direction of price movement correctly but is unable to predict the extent of the spike, which is caused by many external unforeseen factors.

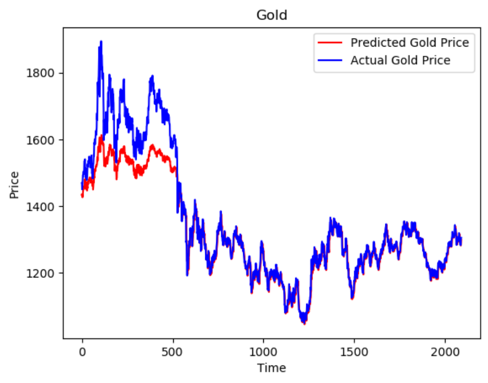

Figure 2: LSTM Model, N = 500

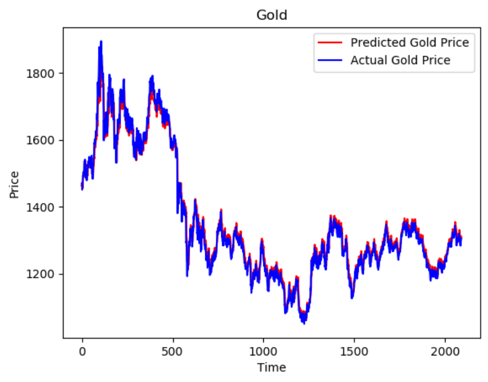

Figure 3: LSTM Model, N = 1000

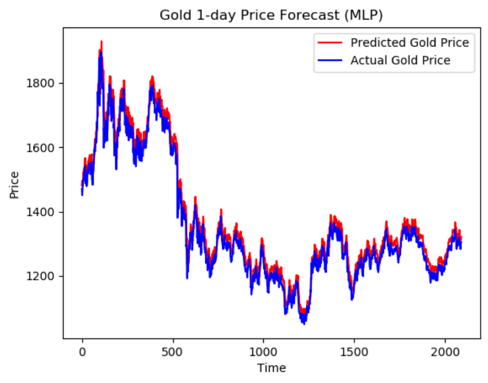

Figure 4: MLP Model

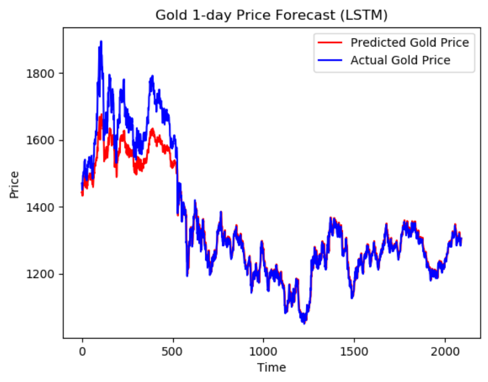

Figure 5: LSTM Model

From Figure 3 where $N = 1000$, it can be observed that the model is able to predict the price the gold accurately, including the spike in the price of gold. One possible conclusion to be drawn is that with a very large value of N, the input data and target output is shrunk to a much smaller magnitude. As such, the magnitude of the spike in price will be shrunk as well, making it easier for the model to predict. However, this will also mean that any inaccuracies in the prediction will be greatly magnified as well, which can be observed in the latter half of the graph in Figure 3.

Ultimately, $N = 500$ was chosen for its consistency and accuracy in predicting day-to-day prices. However, it also means that the model will lose out when it comes to predicting unforeseeable external events.

Model Implementation

MLP vs LSTM

We have attempted using both the MLP and LSTM models, training both models with the same parameters, where each had 2 hidden layers with 20 nodes each and $N = 500$.

From Figure 4 and Figure 5, we are able to conclude that the MLP model is able to better predict unforeseen events (price spike in 2011 due to financial crisis) but falls short behind the LSTM model when it comes to overall accuracy. This result is expected as the LSTM model takes into consideration previous inputs, which is essential when it comes to time series prediction problem. This is akin to the model considering the momentum of the price changes, which explains why it is more resistant to sharp increases and thus, being less accurate when it comes to predicting prices during unforeseen events.

Activation Function

In neural networks, the activation function of a node defines the output when given the inputs. The output produced is then used as input for the next node and this cycle repeats until a desired solution is obtained.

For initial considerations, several activation functions such as sigmoid, tanh and Rectified Linear Unit (ReLU) activation function were considered. A model was chosen based on two criteria: ability to learn fast and good convergence performance. For the first model, the activation function that was utilised is the ReLU. It is a commonly used activation function and is linear for positive values and zero for negative values. ReLU is cheaper and faster to compute due to the simple math in the function while its linearity allows it to converges faster. The slope does not plateau or saturate when x is large and does not face the Vanishing Gradient issue, as mentioned earlier in the report, which is a common problem for other activation functions such as sigmoid or tanh. Simply put, the graph for sigmoid and tanh is an S-curve - when the absolute value of x gets larger, the derivative becomes close to zero (towards the ends of the S-curve).

As mentioned earlier regarding the Vanishing Gradient Problem, with an activation like sigmoid or tanh, if z small derivatives are multiplied together, the gradient exponentially decreases when we propagate down each layer. Ineffective updating of the weights and biases of initial layers during the training phase will cause the model to have an output of zero (Figures 6 and 7).

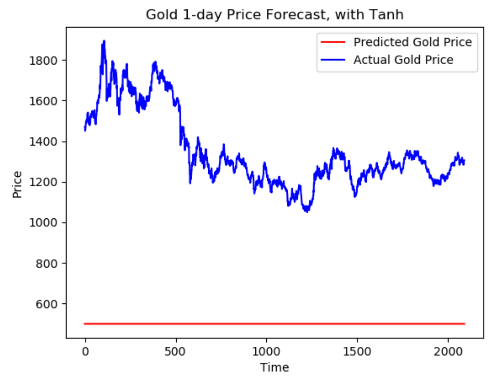

Figure 6: Tanh Activation Function

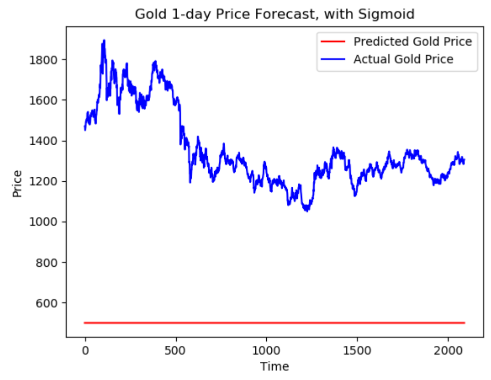

Figure 7: Sigmoid Activation Function

However, there was an issue when implementing ReLU-some of the predicted models had an output of zero in its graph (Figure 8). We realised that this was due to an issue known as a ”dying” ReLU. A ReLU neuron is considered dead when it is stuck in the negative side, constantly giving an output of zero. Because the gradient in the negative range is zero, once a neuron is negative, it is highly unlikely that the neuron will recover. As such, over time, a model using the ReLU activation function may end up with a large portion of their network doing nothing.

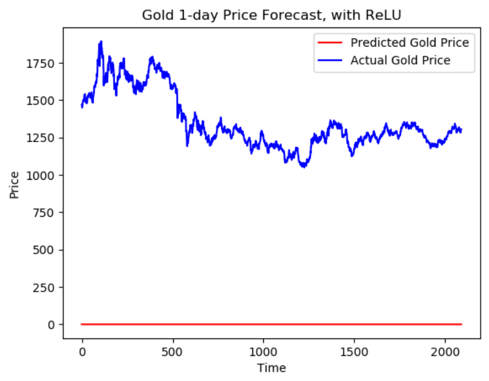

Figure 8: ReLU Activation Function

To combat this, Leaky ReLU was utilised. Instead of the function being zero at negative values, Leaky ReLU will have a small,negative slope, allowing the gradient to flow. With ReLU, the neural network may never learn if the neurons are not activated but it is extremely unlikely with Leaky ReLU. (Figure 9).

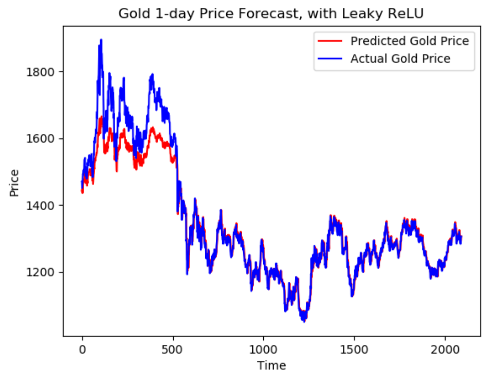

Figure 9: Leaky ReLU Activation Function

Choice of Optimizer

A popular choice for optimizing the weights of the neural networks with backpropagation is stochastic gradient descent (SGD). It minimizes the error between the output of the network and data while using random samples from the data as the input. This is attained by minimizing the gradient of the error function. Despite its popularity, we decided to use a more efficient optimizer known as Adaptive Moment Estimation (ADAM) optimizer. It computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the gradients (Kingma et al., 2014). In short, the ADAM optimizer dominates SGD algorithm in efficiency as it has a lower space and time complexity while having comparable results as shown in Figure 10 and Figure 11. Hence, we opted for ADAM optimizer in our neural networks.

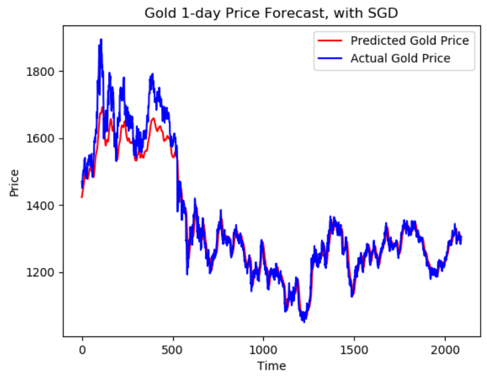

Figure 10: LSTM Model with SGD Optimizer

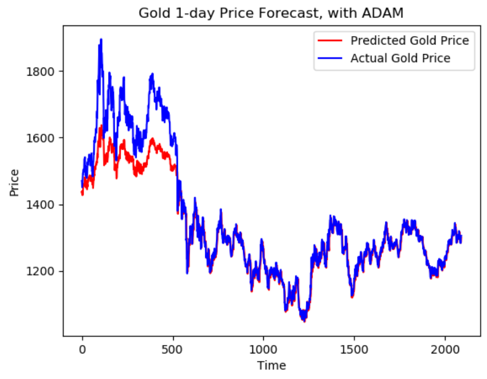

Figure 11: LSTM Model with ADAM Optimizer

Overfitting

Overfitting refers to the function modeling the training data too well. This simply means that the model has high variance and low bias. An indicator of overfitting would be a high accuracy on training data but a low accuracy on test data (Koehrsen, 2018). On the surface, the model may seem to be performing well due to a small error margin on training data but in reality, unable to generalise and replicate the same accuracy output for new, unforseen data. There are numerous solutions to this issue such as utilizing more data, data augmentation and early termination.

As a preventive measure, we utilized the early termination technique. We implemented a patience value that will cause the training to stop when the chosen performance ceases to improve after a set amount of epochs. The patience value indicates the number of training epochs before triggering a change in the learning rate. Often, this value is set between 10 and 100, but it is important to acknowledge that each model and problem will require its own unique patience value (Mahsereci et al., 2017). After testing various patience values, the optimum value for this project was found to be 10% of the total training epochs. As such, the patience value was set to be 50 for all the graphs in this report.

Results

Accuracy of the models is calculated as such:

$$Accuracy = 1 - \frac{1}{X}\sum_{x=1}^{X}\frac{|p_x - o_x|}{o_x} \qquad (1)$$

where X is the total number of examples, p_x and o_x are the xth predicted and observed price respectively.

Using this method of calculation, for the 1-day price forecast, the model is able to achieve an accuracy of 97.99%. If the validation examples from huge spike in gold price are excluded, the model accuracy would improve to 98.95%.

There were attempts to create models with varying n-day price forecasts as well. For a 7-day price forecast, the model is able to achieve an accuracy of 95.23%. If the validation examples from huge spike in gold price are excluded, the model accuracy would improve to 97.41%. This result is expected as a longer forecast would mean more uncertainty.

Similarly, for a 30-day price forecast, the accuracy for the model fell further to 89.00%. If the validation examples from huge spike in gold price are excluded, the model accuracy would improve to 95.18%. Again, this result is expected due to increased uncertainty.

Limitations

Despite significant accuracy, there are still many limitations in the predictive capabilities for the financial market. The price of commodities is affected by many different factors and by only tracking the opening prices daily, the model effectively uses a combination of fundamental analysis to predict future commodity prices. However, this does not take into account unforeseen events such as news and rumours that can have significant impact on the market as well. Furthermore, the opening prices are not the only factors affecting trade. Trade volume, price-to-earning ratio and earnings per share are other quantifiable factors that play a role. However, it is difficult to combine multiple data sets of varying type and scale to conclude on a final, more accurate prediction.

Conclusion

The LSTM model is a reliable method to model the prediction of future stock prices when there is sufficient data available. Even though there are many factors that affect future prices, the LSTM model innately covers fundamental analysis of the commodity prices chosen and the predictions produced are significantly close to the actual prices. However, for manufacturers to reap the maximum benefits from this model, this should not be the only indicator of price trends but rather, couple it with other analysis such as news and sentiments in order to supplement the method that we modelled. In turn, a more robust system can be created to accurately predict the future prices of commodities and allow manufacturers to make more informed choices during procurement.

Miscellaneous

We effectively exploited our manpower by taking advantage of each of our team member's multidisplinary specialities. Being Year 3 Engineering Science (Robotics) majors, Shaun, Ming En and Yin Hao have had brief experiences in using machine learning tools and thus, led our team to construct the required models and optimize them. Yue Yang, a Computational Biology major, extracted data from Quandl before and hence guided us on how to integrate the necessary data into our models. Fionna, a Data Science major, utilized her skills by cleaning and preprocessing data while Aaron, a Computer Science major, built the MLP model. Fionna and Aaron both did not have any prior experience with building machine learning models and learnt from the rest of the team on adapting the necessary code for our project. Despite their brief experiences, Shuan and Yin Hao had never done such a project and were unfamiliar with the in-depth workings of different models (MLP, LSTM) and their applications. After extensive research for this project, they mastered the different model concepts and applications. Yue Yang and Ming En's biggest takeaway was how to adjust the activation functions and parameters accordingly for the desired result. It required thorough experimentation before finally understanding their effects and how to modify it in the future. The most painful experience for us was attempting to live extract and normalize data for real world usage as we were unfamiliar with the process.

Yue Yang: Data extraction and integration of live data with models
Yin Hao: Building of model for LSTM model
Shaun: Optimization of models using various activation functions
Ming En: Optimization of models using various optimizers
Aaron: Building of model for MLP model
Fionna: Data cleaning and data preprocessing

References

Abdel-Aal, R. and Al-Haddad, M. (1997). Determination of radioisotopes in gamma-ray spectroscopy using abductive machine learning. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 391(2):275–288.

Alquist, R., Bhattarai, S., and Coibion, O. (2019). Commodity-price comovement and global economic activity. *Journal of Monetary Economics*.

Harvey, D., Kellard, N., Madsen, J., and Wohar, M. (2017). Long-Run Commodity Prices, Economic Growth, and Interest Rates: 17th Century to the Present Day. *World Development*, 89:57–70.

Heinermann and Kramer, O. (2016). Machine learning ensembles for wind power prediction. *Renewable Energy*, 89:671–679.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116.

Jansma, H. (2018). Intuit and Implement: Batch Normalization. *Towards Data Science*.

King, C., Maxwell, J., and Donovan, A. (2015). Comparing World Economic and Net Energy Metrics, Part 1: Single Technology and Commodity Perspective. *Energies*, 8(11):12949–12974.

Kingma, P, P. D., and Ba, J. (2014). Adam: A method for stochastic optimization. *preprint arXiv*.

Koehrsen, W. (2018). Overfitting vs underfitting: A conceptual explanation.

Mahsereci, M., Balles, L., Lassner, C., and Hennig, P. (2017). Early stopping without a validation set. *arXiv preprint arXiv:1703.09580*.

Marvasti, A. and Lamberte, A. (2016). Commodity price volatility under regulatory changes and disaster. *Journal of Empirical Finance*, 38:355–361.

Miao, K., Chen, F., and Zhao, Z. (2007). Stock price forecast based on bacterial colony RBF neural network. *Journal of Qingdao University (Natural Science Edition)*, 2(11).

Nelson, D., Pereira, A., and de Oliveira, R. (2017). Stock markets price movement prediction with LSTM neural networks. *Proceedings of International Joint Conference on Neural Networks*, pages 1419–1426.

Olah, C. (2015). Understanding lstm networks. *Recurrent Neural Networks*.

Onour and Sergi, B. (2011). Modeling and forecasting volatility in global food commodity prices. *Agricultural Economics (Zemdlsk ekonomika)*, 57(3):132–139.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). Understanding the exploding gradient problem. *Computing Research Repository*, abs/1211.5063.

Patel, Shah, S., Thakkar, P., and Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4):2162–2172.

Rasouli, K., Hsieh, W., and Cannon, A. (2012). Daily streamflow forecasting by machine learning methods with weather and climate inputs. *Journal of Hydrology*, 414–415:284–293.

Roubini, N. (2013). Paranoid investors pushed gold to $1,900 an ounce in 2011, but the bubble has burst. *Slate Magazine*.

Shah-Neha-K, S. (2011). Introduction of Data mining and an Analysis of Data mining Techniques. *Indian Journal of Applied Research*, 3(5):137–139.

Shalabi, L., Shaaban, Z., and Kasasbeh, B. (2006). Data Mining: A Preprocessing Engine. *Journal of Computer Science*, 2(9):735–739.

Skymind (2014). A Beginner's Guide to Multilayer Perceptrons (MLP). *Proceedings of International Joint Conference on Neural Networks*.

Walia, A. S. (2017). Activation functions and its types-which is better? go to the profile of anish singh walia. *Towards Data Science*.