A MODEL TOWARDS PRICE PREDICTION FOR COMMODITIES USING DEEP LEARNING: CASE OF ETHIOPIAN COMMODITY EXCHANGE

SOLEN GOBENA

HAWASSA UNIVERSITY, HAWASSA, ETHIOPIA

June, 2022

A MODEL TOWARDS PRICE PREDICTION FOR COMMODITIES USING DEEP LEARNING: CASE OF ETHIOPIAN COMMODITY EXCHANGE

SOLEN GOBENA

MAJOR ADVISOR: DEGIF TEKA (PHD)

CO-ADVISOR: EFREM YOHANNES (Ph.D. Candidate)

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

HAWASSA, ETHIOPIA

June, 2022

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY EXAMINERS' APPROVAL

SHEET-1

(Submission Sheet-2)

We, the undersigned, members of the Board of Examiners of the final open defense by Solen Gobena have read and evaluated his/her thesis entitled "A MODEL TOWARDS PRICE PREDICTION FOR COMMODITIES USING DEEP LEARNING: CASE OF ETHIOPIAN COMMODITY EXCHANGE", and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirements for the degree.

_____	_____	_____
Name of Major Advisor	Signature	Date
_____ _	_____	_____
Name of Internal Examiner-I	Signature	Date
_____	_____	_____
Name of Internal Examiner-II	Signature	Date
Michael Melese (PhD)		August 22, 2022
Name of External examiner	Signature	Date
_____	_____	_____
SGS Approval	Signature	Date

Acknowledgment

First and foremost, I would like to praise the almighty GOD for giving me this opportunity; without him, I would not have been successful in life and my study. Many thanks to Ethiopian Commodity Exchange for their invaluable support and goodwill for letting me use the data for this study.

It is a great pleasure for me to express my heartfelt gratitude to my advisor Dr. Degif Teka and co-advisor Mr. Efrem Yohannes for their generous support, guidance and valuable feedback on my work. My gratitude also goes to all the community of the Department of Computer Science of Hawassa University Institute of Technology. Finally, I would like to thank all my families, friends for the endless support and encouragement they have given me in both good and bad times during my master's study.

STATEMENT OF THE AUTHOUR

I hereby declare that this MSc thesis is my original work and has not been presented for a degree in any other university, and all sources of material used for this thesis / have been duly acknowledged.

Name: ……...……. Signature: …….……

Place: Institute of Technology, Hawassa University, Hawassa

Date of Submission: …….

Contents

List of Figure

LIST OF TABLE

LIST OF ABBREVIATIONS

ANN: Artificial Neuron Network

ARIMA: Autoregressive Integrated Moving Average

BPTT: Back Propagation through Time

CNN: Convolutional Neural Network

DA: Directional Accuracy

DNN: Deep Neural Network

ECX: Ethiopian Commodity Exchange

ELM: Extreme Learning Machine

FAO: Food and Agriculture Organization of the United Nations

GPU: Graphics Processing Unit

GRU: Gated Recurring Unit

LSTM: Long Short-Term Memory

MAE: Mean Absolute Error

MAPE: Mean Absolute Percentage Error

MLT: Multiple Linear Regression

MLP: Multilayer Perceptron

NEPSE: Nepal Stock Exchange

NSE: National Stock Exchange

NYSE: New York Stock Exchange

RMSE: Root Mean Squared Error

RNN: Recurrent Neural Networks

SLR: Simple Linear Regression

VRNN: Variational Recurrent Neural Network

Abstract

The development of information technology makes it possible to collect and store large amounts of data every second. Market Enterprises are generating large amounts of data, and it is difficult to use traditional data analysis methods to analyze and predict their future market price. Price predictions are an integral component of trade and policy analysis. The prices of agricultural commodities directly influence the real income of farmers and it also affects the national foreign currency. Haricot bean is produced in many areas of Ethiopia and it is rich in starch, protein, and dietary fiber, and is an excellent source of minerals and vitamins. Haricot bean is also the main agricultural commodity traded on the Ethiopian commodity exchange (ECX) market for the past 10 years. Though there are price prediction works for various crops in Ethiopia and abroad using machine learning and deep learning approaches, price prediction for Haricot bean has not been studied using machine learning as to the best of our knowledge,. The main objective of this study is to develop a price prediction model that can predict future prices of Haricot Bean traded at the ECX market based on time series data. Past 10 years, data has been obtained from the Ethiopian commodity exchange (ECX) with sample dataset size of 12272. Simple linear regression (SLR), multiple linear regression (MLR), and long short term memory (LSTM) were evaluated as predictive models. The results showed that LSTM outperformed other predictive models in all measures of model performance for predicting the Haricot Bean prices by achieving a coefficient of determination (R^2) of 0.97, mean absolute percentage error (MAPE) of 0.015, and mean absolute error (MAE) of 0.032.

Keywords: long short-term memory (LSTM), multiple linear regression (MLR), simple linear regression (SLR), Haricot Bean, Time series

Chapter One

1 Introduction

1.1 Background

The development of information technology makes it possible to collect and store large amounts of data every second. Market Enterprises are generating large amounts of data, and it is difficult to use traditional data analysis methods to analyze and predict their future market price. Commodity trading is a very complex market system with a highly developed future commodity electronic trading system. Broadly speaking, a commodity market is any organized market in which tangible or intangible commodities are traded through a single mechanism and promote the most effective competition between buyers and sellers (Exchange, 2017) (Hernandez et al., 2013). The existence of a single market mechanism that combines numerous buyers and sellers at a given point in time has led to the greatest concentration of trade in the product and price bidding system or auction system leads to so-called pricing, that is, due to the greatest degree of concentration and competition between buyers and sellers, the actual market price for compensation of goods is determined at a certain point in time.

The Ethiopian Commodity Exchange (ECX) was established in April 2008 to create an efficient, transparent and orderly trading system to meet the needs of buyers, sellers and intermediaries and increase the market share of small Ethiopian producers (Exchange, 2017). ECX distributes real-time market information on coffee, sesame, Haricot Beans, and provides additional supply contracts. To analyze the market situation in this sector, it is important to have trade data, with this in mind, the exchange generates and stores large amounts of data for each commodity, such as Symbol, year of production, opening price, closing price, maximum, minimum and quantity (lot). ECX data needs to be carefully analyzed for certain parameters to predict future market prices.

Prediction may be defined as the development and use of a model to determine the class of unlabeled sample or the value or range of values of an attribute that a given sample is likely to possess. The purpose of prediction is to determine the numerical value of the target characteristic for unknown objects; this issue type is also known as regression, and if the prediction involves time series data, it is commonly referred to as forecasting (Saud & Shakya, 2020).

Hence, it is important to apply prediction method to the ECX market, which aid farmers in planning and deciding their production portfolio and marketing for increased farm profit, consumers in budgeting, and traders in understanding market trends, and the government in planning economic development in the country. A significant amount of pricing information would reinforce the country's steady connection between production and marketing. Several studies have been carried out elsewhere for market price prediction, yield prediction, and similar areas using machine learning, statistical methods and the more advanced deep learning approach. Various techniques have been used to forecast financial time series, including traditional forecasting methods. In fact, the Autoregressive Integrated Moving Average (ARIMA) and its variants are most commonly used in the literature to predict stock prices (Adebiyi et al., 2014). These have shown that the ARIMA model has strong short-term predictive potential and can compete favorably with existing stock price prediction tools. Unfortunately, these models do not take into account the latent dynamics in the data and are based on assumptions of linearity. Because financial data is not linear most of the time, the researchers looked for other tools that might work better than traditional statistical tools. Classic machine learning began to develop: Numerous studies were developed on machine learning models for predicting financial time series. Among them, the artificial neural network (ANN) model is very popular because of its ability to learn models from data and derive solutions (Aamodt, 2015). However, many articles have reported that the ANN model trained by the back propagation algorithm has some prediction limitations and can easily converge to the local minimum due to the enormous noise and complex dimensionality of the market data. Given these limitations, deep learning has been proposed to overcome the local convergence problem for nonlinear optimization problems and improve the accuracy of the ANN model.

Deep learning neural networks can automatically learn arbitrary complex mappings from inputs to outputs and support multiple inputs and outputs (Hu et al., 2021). There are different deep learning neural network models which can be used for market price prediction. Conventional Neural Network (CNN) is a deep learning model used in the prediction of prices which consists of multiples neurons connected by a hierarchical structure and weight among the neurons can be shared, thus CNN has the advantage to reduce the weight of the network. Long Short-Term Memory (LSTM) is also a deep learning model used in the prediction of prices of a-nonlinear time-series market data. The advantage of LSTM is the design of self-loop which help to flow for an extended period. Recurrent

Neural Network (RNN) is another deep learning neural network model that uses the previous state to calculate the current state and it has good capability in dealing with sequential data.

Generally, deep learning models work well in the prediction of market prices. Therefore, this research uses deep learning techniques to analyze and predict market prices based on previous ECX data. The deep learning model experimented is compared for their efficiency and performance with traditional machine learning models.

1.2 Statement of the Problem

From the beginning, the challenge faced by traders was to predict market prices. Farmers and investors need the information to produce a certain product at any given time to get the most benefit from their investment; this information will make them more profitable. Technical analysts predict the future value of market prices by looking at market charts and identifying patterns and trends. It is difficult for financial analysts to analyze and predict the market based on the large amount of past data generated by the market. Traders start a company without knowing the future market information which might be risky.

The analysis performed so far is based on small data and conventional statistical model and the accuracy of this model is relatively low as compared to state-of-the-art deep learning models. Policy makers, governmental or non-governmental institutes are also using an old prediction method, which is time-consuming and prone to human error. This kind of forecasting approach has an impact on the country's economy. Although various market forecasting studies have been conducted in other countries, it is impractical to directly apply the results to the local context. Market characteristics that affect one country may not have a similar impact on another country; this is due to the economic power of the country, the economic power of farmers, growers and producers and the amount of commodities produced and traded in the market of that country and also price range for commodities can vary from one country to another. To the best of our knowledge, no attempt has been made using a current emerging technology called: deep learning technique to predict the price of commodities at the ECX. Furthermore, up to the knowledge of the researcher no machine learning model has been found for prediction of price for Haricot Beans. Therefore, it was required to study the domestic market carefully to analyze and predict market prices.

1.2.1 Research Questions

With this in mind, this research attempts to answer the following research questions:

1. How to model market price prediction for ECX using deep learning model?
2. What is the performance of deep learning model as compared to the traditional machine learning algorithms for prediction of market price of Haricot Beans?
3. How feature selection improves performance?

1.3 Objective of the Study

1.3.1 General Objective

The general objective of this study is to develop a price prediction deep learning model that can predict future prices of Haricot Bean traded at the ECX market based on time series data.

1.3.2 Specific Objective

To achieve the general objective, the following specific objectives are set.

- To prepare the dataset for experimentation and analysis
- To determine the method of predicting market prices
- To develop a prediction model for predicting market prices for Haricot Beans
- To evaluate market price predictions model accuracy
- To compare the performance of deep learning model with traditional machine learning
- To analyze the research results and make recommendations based on the results obtained

1.4 Scope of the Study

This research focuses on deep learning methods used for forecasting the market price at ECX, making the best recommendations, and providing these details and their results and compare the model with traditional machine learning. This research aims to analyze the dynamic behavior of prices by selecting the main commodity exchanged within the ECX market (i.e. Haricot Bean). The study focuses only on Haricot Bean commodity; since the production and demand for Haricot Bean is increasing in the market.

1.5 Limitation of the Study

The time given to undertake this research was one of the limitations. Due to this, the study was unable to include the international market price of the commodity. So, the attribute was not incorporated, If the attribute was incorporated the performance of the model may increase and may be more robust. The study focused only on Haricot Bean commodity; this was due to the fact that the production and demand for Haricot Bean increasing market.

1.6 Significant of the Study

Since the price of agricultural products directly affects the actual income of farmers and traders, it also affects investors in the field. For growers, price forecasts help plan farms based on expected profits, therefore, it also helps to eliminate price risk. Market price forecasts can also help researchers and policy makers to formulate appropriate economic construction strategies. This research used integrated and updated real-time market price information in order to predict the market price for farmers, traders, investors and decision-makers and it ultimately improve the economic level of the country and farmers.

1.7 Thesis Organization

The remaining chapters of this work are structured as follows; Chapter two describes the overview concept of machine learning, deep learning, price prediction based on deep learning concept and also deals with related work in the area of price prediction. Chapter three describes the methodology, proposed model design, data collection and the prediction algorithms are briefly described. Chapter four the experimental part which briefly shows the result of models and comparison of predictive model using percentage split validation. The last chapter shows the conclusion and future work or recommendation part.

Chapter Two

2 Literature Review and Related Work

Background

This chapter describes models and research relevant to the study in this thesis. Sections 2.1 provide introductory material to Haricot bean markets and 2.2 and 2.3 describe about prediction and machine learning in general, while section 2.4 describes what deep neural networks are and how they work. Sections 2.5 and 2.6 detail about time series data and application of deep learning in ECX and 2.7 provide related wrks, respectively.

2.1 Haricot Beans Market

Haricot Beans are one of the most important beans produced by smallholder farmers for food and income, mainly in the lowlands and Rift Valley in Ethiopia. Figure 2.1 demonstrates haricot bean producing areas in Ethiopia. They are rich in starch, protein and fiber and are an excellent source of minerals and vitamins (FAO, 2015). Its short ripening time (less than three months), high nutritional value, relatively long shelf life and low resource requirements demonstrate its importance even for the poorest farmers in production and consumption(FAO, 2015). Due to its important role in improving food security, export earnings and creating jobs for the national economy, the pulses industry is gaining increasing attention in Ethiopia. Since 2010, white beans have been exclusively exported and are legally required to be listed on the Ethiopian Commodity Exchange (ECX) (FAO, 2015). The introduction of trade in white beans at ECX since October 2010 has significantly changed the value chain. Haricot Beans are delivered to different ECX branches, depending on their distance, for quality control and evaluation. Deliveries are made directly by the seller or through his agent at the ECX quality control center. Upon arrival, a quality assurance officer will take a sample based on visual inspection, evaluate the product, and issue a Receipt Printout (GRN) to the wholesaler. An electronic copy of the GRN number is sent to the ECX exchange in Addis Ababa. The wholesalers then unload their products at the regional ECX warehouse and ship to the ECX exchange in Addis Ababa, where they meet with exporters. They can transact by using their seat bought from the ECX exchange; otherwise, they operate through agents. Exporters who agree to purchase from a supplier will have to sign an agreement and then ECX will transfer funds from the exporter to the wholesaler's account within two days. They will then push the beans from the ECX repository to their own repository. In the ECX system, brokers do not play any role.

However, official local merchants appoint agents on their behalf to exchange and transfer money to their account.

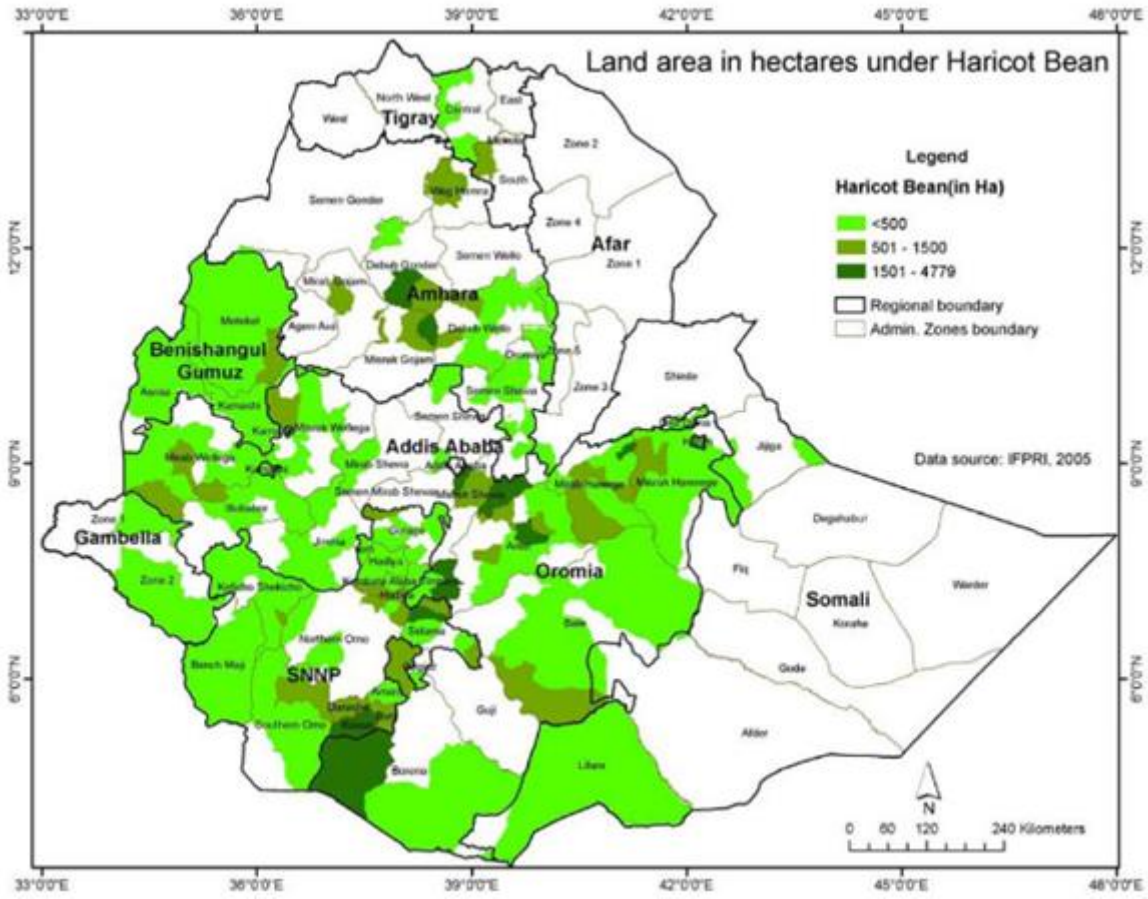


Figure 2. 1 Haricot Bean producing areas in Ethiopia (Source: FAO, 2015)

2.2 Machine learning

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine Learning (ML) is the term used to encompass a variety of techniques used to discover patterns and relationships in data sets (W.Apt, 2014). The fundamental goal of any machine learning algorithm is to discover significant or non-trivial relationships in a "training set" data and produce a generalization of these relationships that can be used to interpret new and unseen data. The knowledge learned by machine learning schemes the structural descriptions of the data can be represented in different ways (W.Apt, 2014) (Abaimov & Martellini, 2022). Schemes such as genetic algorithms or neural networks generate implicit internal models of the data which are not easily understood by human beings or other machines. Other schemes generate more useful descriptions which can be interpreted by human users these include:

Rules

These are a popular form of knowledge representation as they resemble the way human experts tend to describe their knowledge of a domain(Witten et al., 2011) (Abaimov & Martellini, 2022). Rule representations range from simple "if-then" production rules to more complex systems such as "ripple-down" rules and exception based schemes.

Decision trees and graphs

The most basic form of decision graph is a simple binary tree, which is exactly equivalent to a set of if-then rules (W.Apt, 2014). Tree representations are not considered to be as comprehensible to humans as rules. However, newer schemes such as "exception dags" (directed acyclic graphs) are claimed to be more natural and understandable representations

Concept hierarchies

Here the data is classified into a tree of categories and sub-categories. The topmost level of the tree represents the broadest classification of the examples, this is, the most general description. Lower levels of the tree refine the initial classification, with the examples at the leaves of the tree having the most specific description (W.Apt, 2014) (Witten et al., 2011) . A concept hierarchy differs from a decision tree in that examples may be placed at internal nodes of the concept tree. These examples are representative of more general relationships in the data anomalous cases tend to be pushed down towards the leaves. In a decision tree all of the examples are classified at the leaves.

Supervised vs. Unsupervised learning

This is one of the most fundamental distinctions between learning methods. Supervised learning involves developing descriptions from a pre-classified set of training examples, where the classifications are assigned by an expert in the problem domain (Khadka, 2018)(Zou, 2020). The aim is to produce descriptions that will accurately classify unseen test examples. In unsupervised learning, no prior classification is provided, and it is up to the learning scheme itself to generate one based on its analysis of the training data (Khadka, 2018)(Zou, 2020). Unsupervised schemes are often referred to as "clustering" schemes and are often closely related to statistical clustering methods.

Building supervised learning models has three stages:

a. Training: The algorithm will be provided with historical input data with the mapped output. The algorithm will learn the patterns within the input data for each output and represent that as a statistical equation, which is also commonly known as a model.

b. Testing or validation: In this phase the performance of the trained model is evaluated, usually by applying it on a dataset (that was not used as part of the training) to predict the class or event.

c. Prediction: Here we apply the trained model to a data set that was not part of either the training or testing. The prediction will be used to drive business decisions.

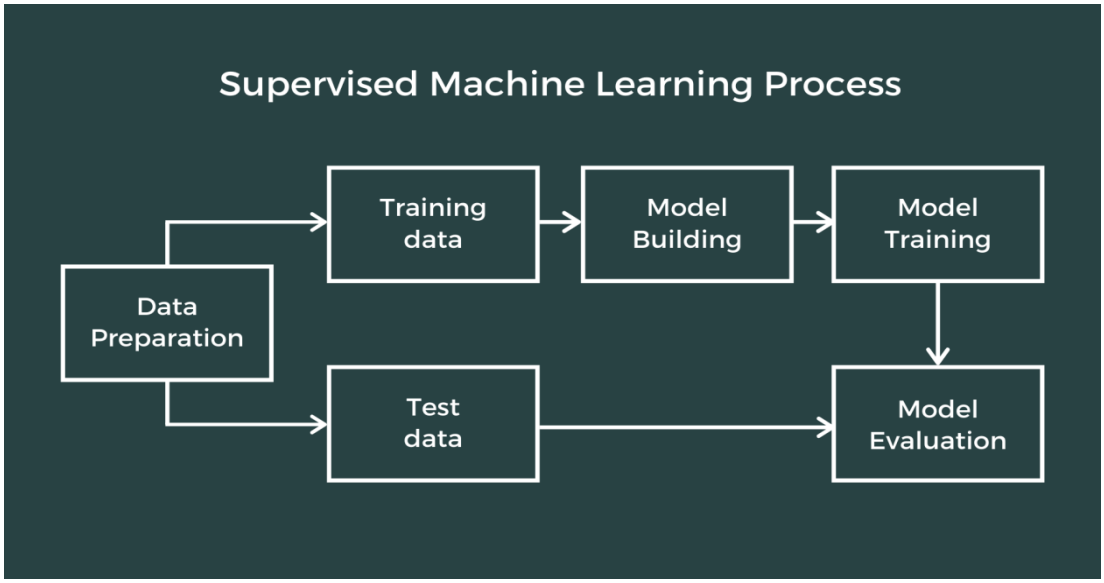


Figure 2. 2 Supervised ML Process

Similarity-based vs. Knowledge-based

Similarity-based schemes form generalizations based solely on the similarities and differences found between the training examples (Witten et al., 2011). Knowledge-based learning makes use of a "domain theory" or background knowledge of the problem domain, usually supplied by an expert (W.Apt, 2014) (Witten et al., 2011). If the domain theory is complete and correct, we have "explanation-based learning," where the scheme attempts to learn new and more efficient ways of interpreting new examples by analyzing how existing examples are explained by the theory. In the case of incomplete background knowledge the learner tries to improve the domain theory by correcting mistakes or adding new parts to the theory.

Top-down vs. Bottom-up

A top-down scheme begins by examining a set of training examples, looking for generalizations that best describe the differences between the cases. In essence the scheme searches the space of possible concept descriptions for one which best matches the data. Bottom-up or "case-based" schemes look at the individual examples and build up descriptive structures from them.

Exact vs. Noise-tolerant

Some schemes are able to cope with noisy input, such as missing or incorrect data. Similarity-based methods often fall into this category, since that look at large numbers of cases simultaneously and can average out some of the effects of noise. Knowledge-based schemes which analyze a single example extensively are more prone to failure when faced with inconsistent data.

2.3 Deep Learning

Deep learning is an artificial intelligence method that trains machines to do what our human brain naturally does: model-based learning. Deep learning is a major innovation that has enabled driverless vehicles to enable them to perceive traffic signs or recognize a person walking or even distinguish whether a driver is conscious or not in order to safely park the car and get in a traffic jam to avoid. It is also behind voice-controlled devices such as smartphones, tablets, televisions, wireless speakers and given the current circumstances we get results that were previously considered unrealistic. Figure 2.3 is an illustration that explains the difference between artificial intelligence, machine learning, and deep learning.

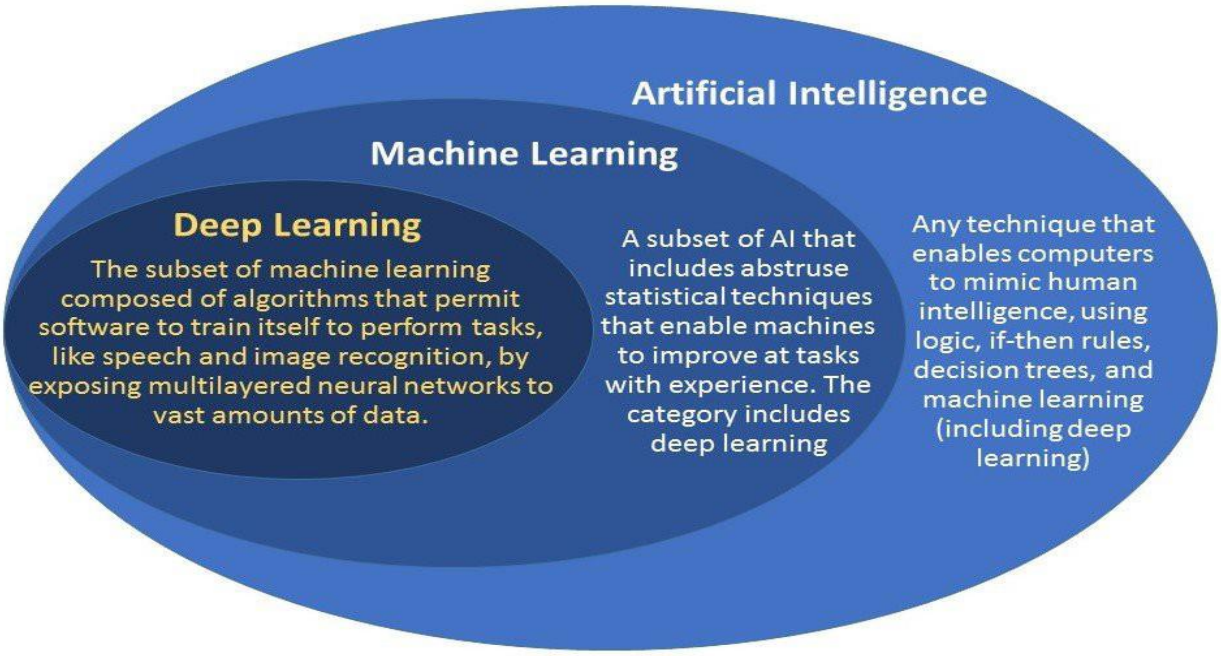


Figure 2. 3 A comparison between Deep learning, Machine learning and Artificial Intelligence (source https://qph.fs.quoracdn.net)

Deep learning achieves higher popularity accuracy at better rates; this enables the consumer electronics buyer to meet consumer demands and is critical to unique security programs that include self-sufficient vehicles. Figure 2.3 is a determination that illustrates a simple neural network and a deep learning neural network.

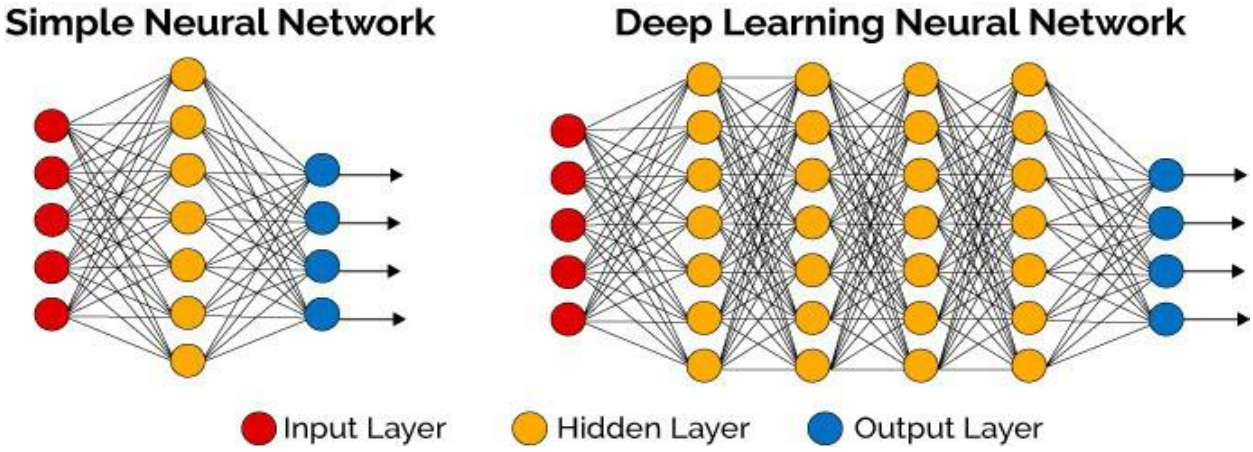


Figure 2. 4 Conceptual illustrations of a simple neural network and a Deep Learning Neural Network

As shown above, a deep learning neural network is made up of multiple hidden layers, much like stacking multiple simple neural networks. Since the first theoretical deep learning methodology in the 1980s, deep learning has only just begun to approach its true potential

for two fundamental reasons: Deep learning requires a large amount of tagged information or data that was only available in the last decade. The implementation of the autonomous car takes millions of photos and thousands of hours of footage. Deep learning requires considerable computing power. Highly efficient GPUs have a highly effective parallel architecture that enables development teams to reduce learning time from weeks to hours or minutes when used in conjunction with clusters or cloud computing. Recurrent neural network, Convolutional neural network and long-short term memory are among the deep neural networks.

2.3.1 Recurrent Neuron Network (RNN)

Recurrent neural networks (RNNs) have a long history and were developed in the 1980s. The Hopfield network, introduced in 1982 by J. Hopfield, can be considered one of the first networks with periodic connectivity, a fully connected neural network was mentioned in 1989 and the famous Elman's network was introduced in 1990. Elman's web was inspired by the architecture used by Jordan; therefore they are often collectively referred to as the Elman and Jordan networks (Sherstinsky, 2020)(Widegren, 2017). Figure 2.5 shows Jordan's network.

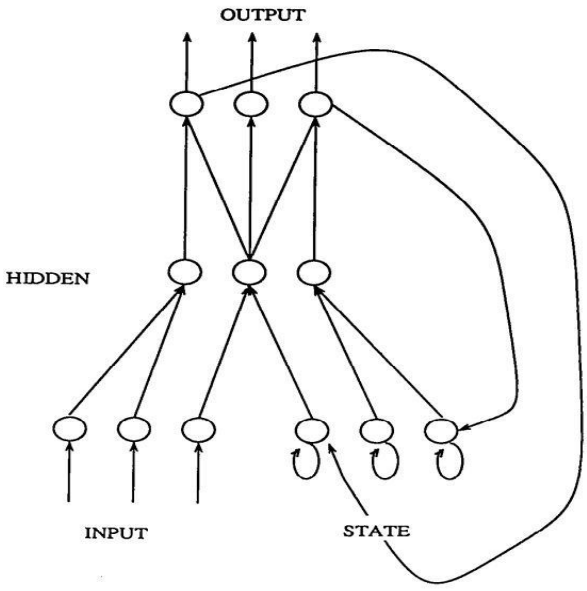


Figure 2. 5: Jordan network (1986).

The basic difference between a feed forward neuron and a recurrent neuron is feed forward neuron has only connections from the input to the output. The recurrent neuron instead has

also a connection from the output again to the input and therefore it has three weights. This third extra connection is called feed-back connection and with that the activation can flow round in a loop (Widegren, 2017) as shown in figure 2.6. The recurrent network can use the feedback connection to store information over time in form of activations

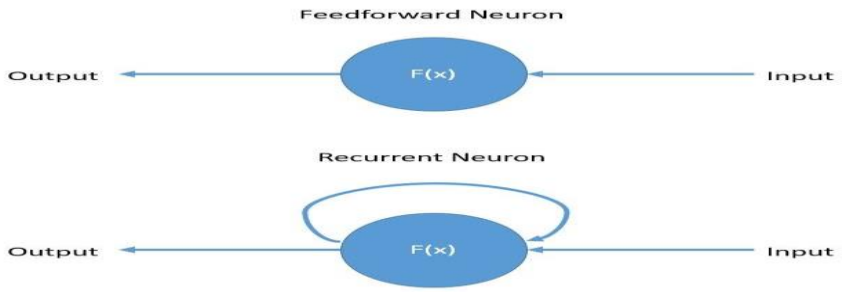


Figure 2. 6 Feedforward vs. Recurrent neuron

2.3.1.1 Architecture of Recurrent Neural Network

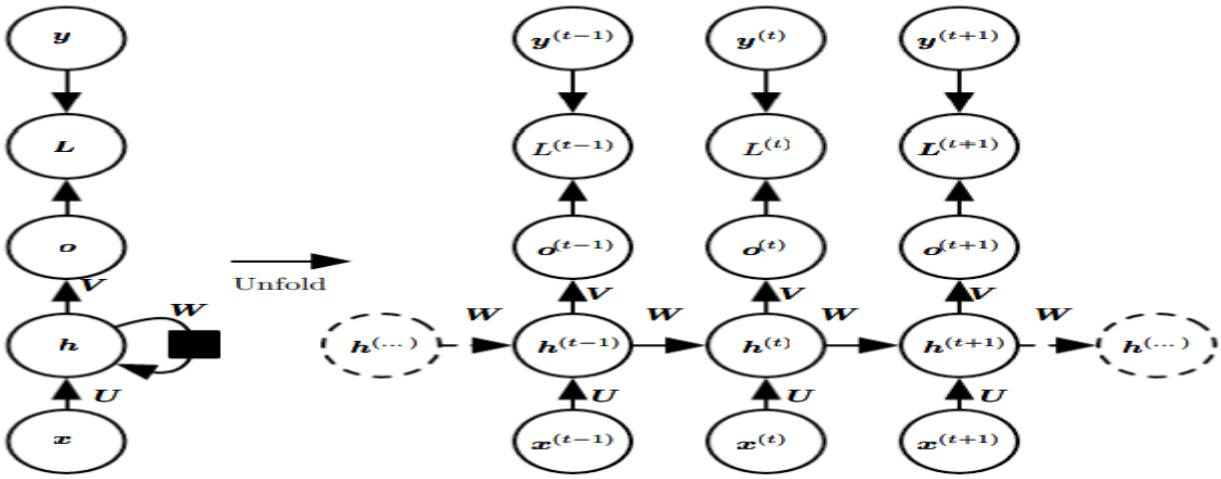


Figure 2. 7 Architecture of recurrent neural network

Recurrent Neural Network maps an input sequence x values to a corresponding sequence of output o values loss L measure the difference between the actual output y and the predicted output o (Sherstinsky, 2020)(Aamodt, 2015). The RNN has also input to hidden connection parameterized by a weight matrix U, hidden to hidden connections parameterized by a weight matrix W, and hidden-to-output connections parameterized by a weight matrix V (Sherstinsky, 2020) (Widegren, 2017). Then from time step t = 1 to t = n we apply the following equation:

$$\alpha^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \qquad (1)$$

$$h^{(t)} = \tanh(\alpha^{(t)}) \quad (2)$$

$$o^{(t)} = c + Vh^{(t)} \quad (3)$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)}) \quad (4)$$

Propagation equations of the recurrent neural network where U, V, W are the weight matrix that is shared among each time step.

The above equations are also known as forwarding propagation of RNN where the b and c are the bias vectors and tanh and softmax are the activation functions. To update the weight matrix U, V, W we calculate the gradient of the loss function for each weight matrix i.e. ∂L/∂U, ∂L/∂V, ∂L/∂W, and update each weight matrix with the help of a back-propagation algorithm (Sherstinsky, 2020). When a back-propagation algorithm is applied to RNN, it is sometimes also known as BPTT i.e. back propagation through time (Aamodt, 2015) (Sherstinsky, 2020). Gradient calculation requires a forward propagation and backward propagation of the network which implies that the runtime of both propagations is O(n) i.e. the length of the input.

Depending on the objective we can choose any loss function. Total loss for a given sequence of x values is the sum of all the losses at an individual time step. Another variation that can be done in recurrent neural network architecture is that we can change the recurrent connection from hidden to hidden state and make it from output to hidden state (Sherstinsky,2020) as shown in the figure 2.8

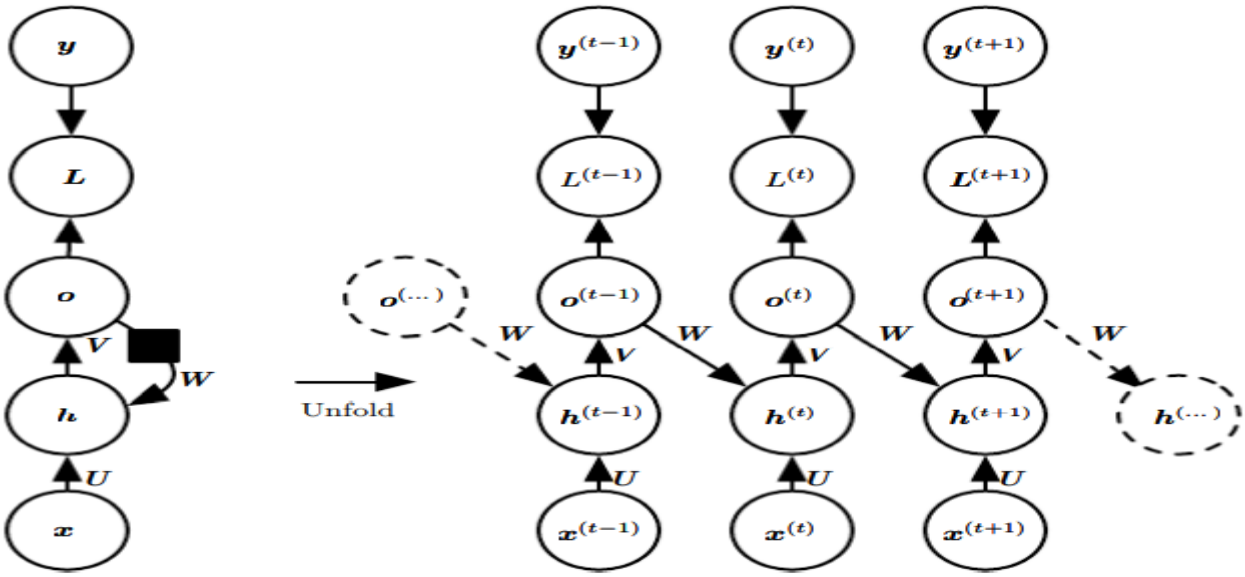


Figure 2. 8 Recurrent neural network connections

2.3.2 Long Short Term Memory model (LSTM)

LSTM, which stands for Long Short Term Memory, is a type of neural network that is particularly useful in time series prediction. According to an article by Srivastava on LSTM and the basics of deep learning, an LSTM network is the most effective solution for the analysis of time series and thus for the prediction of the stock market. Srivastava says, "Sequence prediction problems have been around for a long time, they are considered to be one of the most difficult problems to solve in the data science industry". This includes a variety of problems; from predicting sales to finding patterns in stock market data, from understanding movie plots to recognizing the way you speak, from language translations to predicting your next word on the phone keypad with the recent advances in data science, it has been found that for almost all of these sequence prediction problems, long and short term memory networks have been considered to be the most effective solution (Yang et al., 2020).

LSTMs have an advantage over conventional feed forward neural networks and recurrent neural networks in many ways, since they selectively remember patterns over long periods of time. Figure 2.9 shows a simplified LSTM cell.

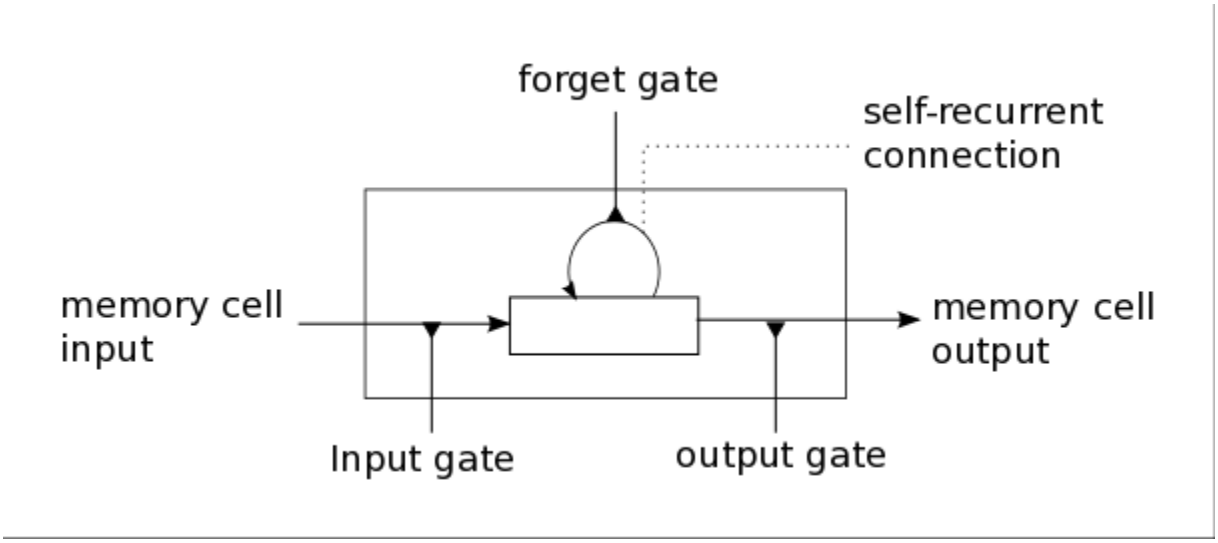


Figure 2. 9 Simplified look at an LSTM cell.

In the case of a basic neural network, adding new information completely transforms the existing information by applying a sigmoid function so that all information is modified as a whole. In LSTM, information flows through a mechanism known as cell states Forget things (Widegren, 2017).

2.3.2.1 LSTM Cell Architecture

A typical LSTM network consists of several blocks of memory called cells. There are two states that are carried forward to the next cell; the cell state and the hidden state. Blocks of memory are responsible for remembering things, and manipulation of this memory occurs through three main mechanisms called forget gate, input gate and output gate. Figure 2.10 shows LSTM cell architecture.

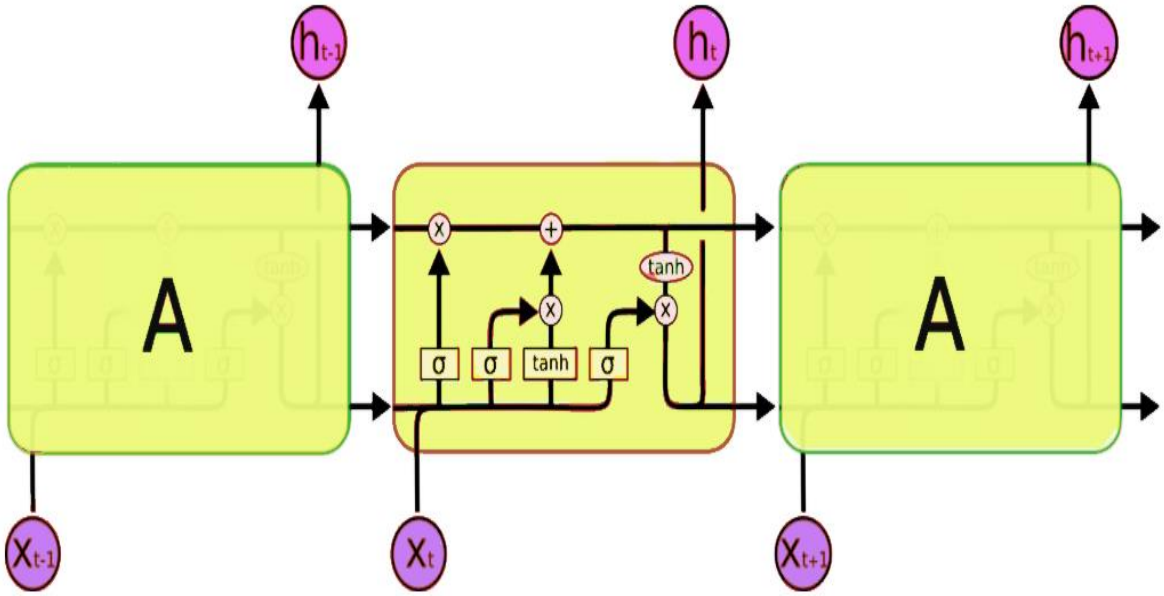


Figure 2. 10 LSTM Cell Architecture

2.3.2.1.1 Forget Gate

Srivastava describes it this way: “A forget gate is responsible for removing information from the state of the cell. Information that is no longer necessary for the LSTM to understand things or information of lesser importance is removed. This gate takes in two inputs; h_t-1 and x_t. h_t-1 is the hidden state from the previous cell or the output of the previous cell and x_t is the input at that particular time step. The given inputs are multiplied by the weight matrices and a bias is added, then the sigmoid function is applied to that value, the sigmoid function gives a vector with values in the range 0 to 1, corresponding to each number in the state cell. Basically, the sigmoid function is responsible for deciding which values to keep and which to discard. If a '0' is output for a certain value in the cell status, this means that the forgetting gate wants the cell status to forget this part of the cell. Similarly, a '1' means that the gate to oblivion wants to remember all information. This output vector of the sigmoid

function is multiplied by the state of the cell. Figure 2.11 shows LSTM cell architecture of forget gate.

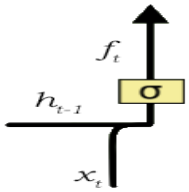


Figure 2. 11 Forget Gate, Internal Architecture

2.3.2.1.2 Input Gates

Srivastava explains, "The gateway is responsible for adding information about the cell state. This information addition is basically a three-step process, as can be seen in the diagram above.

1. Regulating what values need to be added to the cell state by involving a sigmoid function. This is basically very similar to the forget gate and acts as a filter for all the information from h_t-1 and x_t.
2. Creating a vector containing all possible values that can be added (as perceived from h_t-1 and x_t) to the cell state. This is done using the tanh function, which outputs values from -1 to +1.
3. Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

Once this three-step process is done with, we ensure that only that information is added to the cell state that is important and is not redundant."

Figure 2.12 shows the Internal Architecture of Input Gate

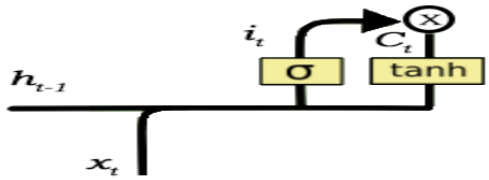


Figure 2. 12 Input Gates, Internal Architecture

2.3.2.1.3 Output Gate

To quote Srivastava: “The output gate is responsible for selecting useful information from the current state of the cell and displaying it as an output”. The operation of an output gate can in turn be divided into three steps:

1. Creating a vector after applying tanh function to the cell state, thereby scaling the values to the range -1 to +1.

2. Making a filter using the values of h_t-1 and x_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.

3. Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.”

Figure 2.13 shows the Internal Architecture of output Gate

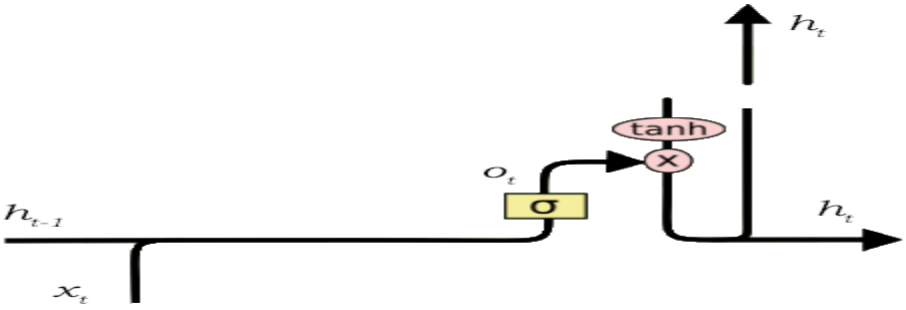


Figure 2. 13 Output Gate, Internal Architecture

2.4 Time Series Analysis

Before we talk about time series analysis, let's first define what a time series is: it is a sequence of data points or values arranged in a list or plotted against the flow of time; in most cases a sequence of time series will have equidistant continuums in time data points. Two main purposes of time series analysis are: (1) to identify the nature of the observation sequence phenomenon, and (2) to predict (accurately predict) the time series variable. Both goals require the identification of patterns of observed time series as well as a formal description. We can interpret the pattern and embed it in other data sets once the pattern has been identified. Regardless of our complexity and correctness of understanding the concept, we can infer the recognized pattern to logically predict the results (Khadka, 2018)(Zou, 2020). Time series analysis includes techniques for disaggregating information from a sequence of data points in order to extract key measurements and various attributes from the information. Time series prediction is the use of a model to predict future value by examining patterns in pre-existing data sets or sets of values. In addition, a random model for time series prediction is always restricted by the fact that data points that are closer in time are more closely correlated than data points with a greater time interval. Maintain a long-term correlation without neglecting short-term dependencies between data points is exactly the challenge in time series forecasting (Saud & Shakya, 2020). As such, many data mining techniques can be used to solve the problem with varying degrees of success.

2.5 Applications of Deep Learning in ECX

In the literature, we note that haricot beans are one of the most important beans produced by smallholder farmers for a living and livelihood in Ethiopia. We base our analysis on the bean industry as it plays an important role in improving food security, export earnings and job creation for the national economy of Ethiopia, due to the fact that these industries must be supported by different research methods, we tried to predict the prices of beans using deep learning applications in different aspects or attributes to help exporters and farmers do more and benefit in terms of costs and human resources.

2.6 Related Work

(Hiransha et al., 2018) Study NSE Stock Market Prediction Using Deep-Learning Models. In the paper, they used four types of deep learning architectures i.e. Multilayer Perceptron (MLP), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) for predicting the stock price by using day-wise

closing price of two different stock markets, National Stock Exchange (NSE) of India and New York Stock Exchange (NYSE). They trained the network with stock price from NSE and predicated for five different companies from both NSE and NYSE. The study reported that CNN outperformed all the other models.

(Shen & Shafiq, 2020) investigates Short-term stock market price trend prediction using a comprehensive deep learning system. In order to solve the problem, they used different data mining techniques to evaluate the Chinese stock market dataset. Stock price trend prediction model based on the long short-term memory (LSTM) were used. They compare the most frequently used machine learning models with their prediction model LSTM. It is observed that long short-term memory (LSTM) outperforms the machine learning model.

The paper (Saud & Shakya, 2020) investigates on predicting the stock price of NIB and NABIL commercial banks which is listed on Nepal Stock Exchange (NEPSE). Three deep learning methods VRNN, LSTM and GRU were applied in predicting the stock price and their performance was analyzed. Based on the amount of data they used for training the prediction model it showed that GRU and LSTM are able to outperform VRNN in predicting stock prices. In predicting the stock price of NIB and NABIL for GRU it was 4.74 and 4.71 respectively but it was 5.58 and 5.06 respectively for LSTM.

(Balaji et al., 2018) investigates the application of different types of deep learning approaches in predicting stock price. Root Mean Squared Error (RMSE), Directional Accuracy (DA) and the Median Absolute Percentage Error (MdAPE) were used to measure the performance of prediction model based on Long-short term Memory (LSTM), Gated Recurring Unit (GRU), Convolutional Neural Networks (CNN) and Extreme Learning Machines (ELM). Their result showed that all models are capable of forecasting a market. GRU based models provide better DA for shorter forecast horizon and ELM based models for longer forecast horizon. Similarly, GRU based models give the lowest MdAPE values for shorter forecast horizon and ELM based models give the least MdAPE values for longer. Generally they have showed that deep learning models stated in their study are capable of generating highly accurate stock price forecasts.

(Gebisa, 2019) used seven years stored data from the ECX database and applied different machine learning method to predict the market price of ECX. Based on the data given

different result was obtained from the machine learning model applied. Decision tree Regression (CART): 0.909470, SVR: 0.910699, KNN: -0.103426 and LSTM 0.91523. The result has showed that LSTM outperformed all the experimented models.

The study (Worku,T.2015) is based on data mining classifiers: Gaussian Processes, Linear Regression, Multilayer Perceptron, SMOreg, Decision Table, M5Rules, M5P and REPTree. The author used data stored from the ECX data base from 2010 to 2015 and it was observed that Decision Table Classifier has the highest optimal accuracy score with maximum optimal Correlation Coefficient Percentage (CCP) of 97.8%, minimum optimal Root Mean Square Percentage Error (RMSPE) of 3.9% and an optimal Time of 0.02 seconds to build the Price Predictive model.

One of the main weaknesses found in the works involved was the limited data preprocessing mechanisms that are created and used. Most of the work mostly focused on building predictive models. Gebisa (Gebisa, 2019) and Worku (Worku,2015) predicted the local market price of Ethiopia commodities traded in Ethiopian Commodity Exchange. (Worku,2015) indicated that the Decision Table model has a strong predictive potential in the short term and has the ability to compete favorably with existing stock price prediction tools. Unfortunately, these models do not consider the latent dynamics existing in the data and they are based on linearity assumptions. (Gebisa, 2019) showed the performance of the machine learning model in predicting the market price of coffee. He concludes that Machine learning is a useful tool for predicting stock prices in emerging markets. However, many papers have reported that the Machine learning model has some limitations in forecasting (Shen & Shafiq, 2020), and it can easily converge to the local minimum because of the tremendous noise and complex dimensionality of the stock market data. In view of these limitations, Deep Learning has been proposed to overcome the local convergence issue for non-linear optimization problems and enhance the accuracy of the Machine learning model. We believe that by using a deep learning model based on data stored in ECX database, we can benefit producers and investors by implementing predictive models that have been well tested.

Chapter Three

3 Research Methodology

3.1 Research Design

3.1.1 Research Approach

The proposed method use experimental research to create a specific deep learning tool based on a model and testing its performance on a practical data. It attempt to identify new insights into the price predictor for haricot bean forecasting.

3.1.2 Research Data

The study uses daily haricot bean historical data for model development and evaluation purposes.

3.1.2.1 Data Sources

In this study, data was taken from ECX historical market data retrieved from the ECX database. The data required for this study was requested from ECX in an official collaboration letter written by the Institute of Technology of Hawassa University. After the data access request was approved by the department and responsible division, the request data was collected form ECX. Past 10 years, data has been obtained from the Ethiopian commodity exchange (ECX) with sample dataset size of 12272.

3.1.3 Data preparation

This stage includes all the steps that will be involved in creating the final dataset (data imported into the model) from the original source data. This step also includes choosing the best method for extracting data for the model and splitting data for the model. Splitting data involves segmenting a sample of data into two sets for training and testing the model. The training set forms the bulk of the sample data and is used by the deep learning model to learn the patterns present in the data, 80% of the data was used as a training set and the remaining 20% were used as a test set.

Thus, the sample data was partitioned as follows:

1. Training set = 80% of total sample size.

2. Testing set = 20% of total sample size.

The volume of data implied a period of ten years for the daily haricot bean price data.

3.1.4 Data evaluation

Data evaluation first involves transforming the input and output variables to reduce noise. Data mapping is the next step towards understanding the underlying relationships. Data transformation is very important in deep learning to achieve good predictive performance by eliminating bias and correlations between inputs and make them statistically independent (Oancea and Ciuncu, 2014). Research (Nayak, Misra & Behera, 2014) concluded that transforming data through normalization speeds up training times. This is very valuable when modeling an application where the inputs are at vastly different scales. MinMax scaling is one of normalization technique that involves moving the decimal point in attribute values. Min-Max scaling is a normalization technique that enables us to scale data in a dataset to a specific range using each feature's minimum and maximum value.

The transformation is given by:

$$X\_std = (X - X.min) / (X.max - X.min) \quad (5)$$

$$X\_scaled = X\_std * (max - min) + min \quad (6)$$

3.2 Feature Selection

This step is to take time to understand what kind of data is available and which data is the most appropriate and has the most predictive power that can address the objective. The dataset consists of 14 attributes. These attributes are trade date, commodity, commodity type, commodity name, description, symbol, origin, grade, opening price, closing price, max price, min price, warehouse and production year. In this step, only the features which are to be fed to the neural network are chosen. This work used closing price as a target attributes to predict future price with combination of other selected attributes.

3.3 Model

A deep neural network model called Long Short-Term Memory networks (LSTMs) was trained and tested and compared with machine learning model called Simple Linear regression and Multiple Linear Regression.

3.3.1 Simple Linear regression

Linear regression is utilized, to identify the relationships between a dependent variable and independent variables and it is good for working with continuous and numerical data. Simple linear regression is defined by using a feature to predict an outcome.

Simple regression model allows for a linear relationship between the forecast variable y and a single predictor variable x:

$$yt = \beta 0 + \beta 1xt + \varepsilon t \quad (7)$$

The coefficients β0 and denote the intercept and the slope of the line respectively. The intercept β0 represents the predicted value of y when x=0. The slope β1 represents the average predicted change in y resulting from a one unit increase in x.

3.3.2 Multiple Linear Regressions

Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable. It is an important regression algorithm that models the linear relationship between a single dependent continuous variable and more than one independent variable. It uses two or more independent variables to predict a dependent variable by fitting a best linear relationship.

It has two or more independent variables (X) and one dependent variable (Y), where Y is the value to be predicted. Thus, it is an approach for predicting a quantitative response using multiple features.

$$Y = \beta 0 + \beta 1X1 + \beta 2X2 + \beta 3X3 + \cdots + \beta nXne + e \quad (8)$$

Y = Dependent variable / Target variable

β0 = Intercept of the regression line

β1, β2, β3… βn = Slope of the regression line which tells whether the line is increasing or decreasing

X1, X2, X3, ….Xn = Independent variable / Predictor variable

e = Error

3.3.3 Long short-term memory (LSTMs)

LSTM is a type of Recurrent Neural Network (RNN). Recurrent neural network is designed to avoid long-term dependence problems and is well suited for processing and predicting time

series. Developed by Sepp Hochreiter and Jurgen Schmidhuber in 1997, the LSTM model is based on a unique set of memory cells that replace neurons in the hidden layer of RNN, and its key is the state of the memory cells. As a matter of fact, with its ability to remember both long term and short term values, the LSTM models have proved very rewarding for the treatment of financial time series, thereby becoming the preferred Deep Learning tool for time series analysis (Sherstinsky, 2020). Furthermore, their ability to handle very noisy data and their independence from the linearity assumption makes them go to models for analysis of commodity prices.

As shown in figure 3.1 LSTMs take inputs from the previous time step into account when modifying the model's memory and input weights. The input gate makes decisions about which values are important and should be let through the model (Olah, 2015). A sigmoid function is used in the input gate, which makes determinations about which values to pass on through the recurrent network. Zero drops the value, while 1 preserves it. A TanH function is used here as well, which decides how important to the model the input values are, ranging from -1 to 1 (Sherstinsky, 2020) (Olah, 2015).

After the current inputs and memory state are accounted for, the output gate decides which values to push to the next time step. In the output gate, the values are analyzed and assigned an importance ranging from -1 to 1. This regulates the data before it is carried on to the next time-step calculation. Finally, the job of the forget gate is to drop information that the model deems unnecessary to make a decision about the nature of the input values. The forget gate uses a sigmoid function on the values, outputting numbers between 0 (forget this) and 1 keep this (Sherstinsky, 2020) (Olah, 2015).

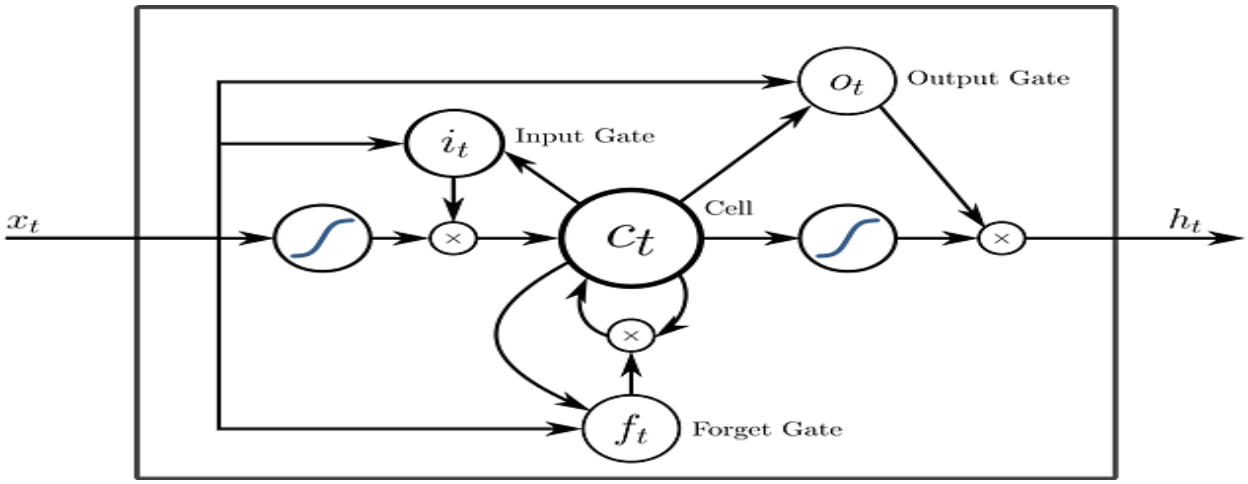


Figure 3. 1 LSTM model

An LSTM neural network is made out of both special LSTM layers that can interpret sequential word data and the densely connected like those described above. Once the data moves through the LSTM layers, it proceeds into the densely connected layers

3.4 Performance Evaluation

The performances of the models developed in this study have been assessed using various standard statistical performance evaluation criteria. The researcher used three evaluations metric such as (R^2), mean absolute error (MAE) and mean absolute percentage error (MAPE) for evaluating and comparing the models performance. And also select the best model according to evaluations techniques.

3.4.1 MAE (Mean Absolute Error)

The simplest measure of forecast accuracy is called Mean Absolute Error (MAE). Mean Absolute Error is simply, as the name suggests, the mean of the absolute errors. The absolute error is the absolute value of the difference between the forecasted value and the actual value. Mean Absolute Error measures accuracy for continuous variables. Mean Absolute Error tells us how big of an error we can expect from the forecast on average. The Mean Absolute Error measures the average magnitude of the errors in a set of predictions, without considering their direction. The Mean Absolute Error is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight. Both Mean Absolute Error and Root Mean Square Error express average model prediction error in units of the variable of interest. The Mean Absolute Error and the Root Mean Square Error can range from 0 to ∞ and are indifferent to the direction of errors.

$$\mathrm{MAE} = |(1/\mathrm{n})|\mathrm{a1} - \mathrm{c1}| + \cdots . . + |\mathrm{an} - \mathrm{cn}| \quad (9)$$

Where:-

a is the observation data and c computed data and n is the number of data. Assuming that the actual output is **a**, expected output is **c**. To make regression more robust minimize absolute error, not squared error.

3.4.2 MAPE (Mean Absolute Percentage Error)

One of the most common metrics used to measure the forecasting accuracy of a model is MAPE. The mean absolute percentage error (MAPE) is the mean or average of the absolute percentage errors of forecasts. Error is defined as actual or observed value minus the

forecasted value. MAPE can be considered as a loss function to define the error termed by the model evaluation.

$$\text{MAPE} = \left(\frac{1}{n}\right) * \Sigma \left(\frac{|\text{actual} - \text{forecast}|}{|\text{actual}|}\right) * 100 \quad (10)$$

Where:

n = sample size

Actual = the actual data value

Forecast = the forecasted data value

3.4.3 R-squared (R^2)

R-squared is the "percent of variance explained" by the model. That is, R-squared is the fraction by which the variance of the errors is less than the variance of the dependent variable. It is called R-squared because in a simple regression model it is just the square of the correlation between the dependent and independent variables, which is commonly denoted by "r". In a multiple regression model R-squared is determined by pairwise correlations among all the variables, including correlations of the independent variables with each other as well as with the dependent variable.

3.5 Proposed Architecture

The high-level architecture of our proposed solution can be divided into three parts: the first is the choice of features, which ensures high efficiency of the selected functions; secondly, we check the data and do the dimensionality reduction; and the final part, the main contribution of our work, is to build a predictive model by training and testing the model with specific data.

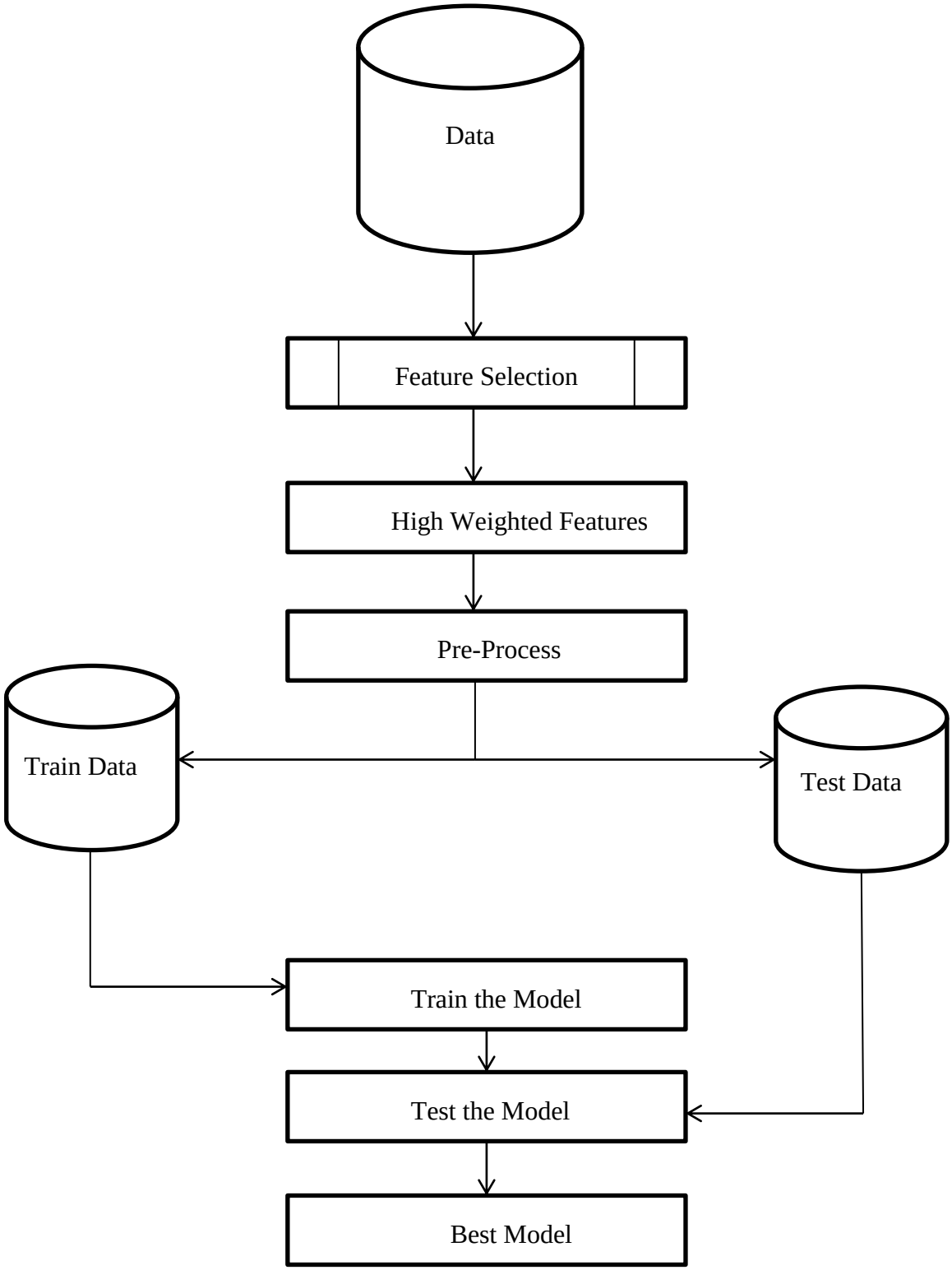


Figure 3. 2 the proposed work frame

The proposed solution is divided into three parts: the first is the choice of features, which ensures high efficiency of the selected functions three machine learning methods that have their own built-in feature selection function: Information gain, Random Forest and Pearson correlation, all three methods were employed to select the subset of features having the highest coefficient of determination (R^2).; secondly, we check the data and do the dimensionality reduction MinMaxScaler from the sci-kit learn library was used to scale the input data to a range between 0 and 1.; and the final part, the main contribution of our work, is to build a predictive model by training and testing the model with specific data.

3.6 Tools and Libraries

Python scripts are written to convert raw stock prices (.csv files) into function vectors for training, forecasting, and testing, respectively. An interactive environment called a Colab notebook is used to write and execute code. Numpy provides efficient multidimensional array data structures as well as functions for manipulating arrays. Since we need to convert the dataframe into matrix hence numpy is used. It converts data into array. For reading raw input files such as .csv and converting the time series data to the correct format and to visualize the data as well for manipulation of data panda library was used. Matplotlib is imported which performs such act plt used to plot both the predicted as well as actual values and it is a very useful library for implanting plots into submissions developing totally beneficial. A machine learning python library called Scikit-Learn/SkLearn which chiefly includes arrangement, relapse, and bunching calculation is used for splitting the dataset into training and testing phase and to normalize the dataset. Sequential model were imported from keras layers, output was predicted using dense layer and hence this layer were also imported from keras and also LSTM is imported from keras layers because keras supports deep neural network as well as activation layers. Math library is used to compute mathematical functions such as the mean square error.

Chapter Four

4 Result and Discussion

This chapter presents the experimental result and evaluates the performance of each model. In this study the researcher aims at addressing three research questions as documented in chapter one: how to model market price prediction for ECX using deep learning model? The second was what is the performance of deep learning model as compared to the traditional machine learning algorithms for prediction of market price of Haricot Beans? And the last was how feature selection improves performance?

4.1 Experimental Result of Predictive Algorithms

The three predictive Simple Linear Regression, Multiple Linear Regression and LSTM are studied on the experiment. All these models were trained with the collected data-set from ECX. The motive behind this was to predict future market price of commodity and to compare the performance of deep learning model with the traditional machine learning models.

4.1.1 Feature Selection

Real world data sets have a very large number of attributes arising from a diverse array of measurements. Alternatively, we may come up with unimportant attributes when we aren't really sure what features to be fitted in our model. The collected dataset has 14 attributes. The researcher used three machine learning methods that have their own built-in feature selection function: Information gain, Random Forest and Pearson correlation, all three methods were employed to select the subset of features having the highest coefficient of determination (R^2). Information gain method achieved the highest R^2 value (0.948). The feature subset selected by Information gain were Opening price, Max-price, Min-price and production year as presented in table 4.1.

Table 4. 1 Attributes selection

No	Method used	Selected Attribute	Total Attribute Selected	Coefficient of Determination (R^2)
1	**Information gain**	**Opening price, Max-price, Min-price Production year**	**4**	**0.948**
2	Pearson correlation	Opening price, Production year	2	0.909
3	Random forest	Opening price, Max-price, Min-price	3	0.926

Attributes resulting from Information gain (Opening price, Max-price, Min-price and Production year) are fitted to the neural network and trained using the LSTM model and these attributes are also fitted to the multiple linear regression model.

4.1.2 Predicting of Haricot Bean Price using LSTM

As it has been said LSTM was the deep learning models used in order to obtain the predicting Haricot bean price.

4.1.2.1 Transforming the Data

MinMaxScaler from the sci-kit learn library was used to scale the input data to a range between 0 and 1. An additional scaler that works on a single feature column (scaler_pred) was created and reused the scaler later when we want to unscale our model predictions (1 dimensional). The model was trained on a three-dimensional data structure. The first dimension is the sequences, the second dimension is the time steps (mini-batches), and the third dimension is the features. An essential step in time series prediction is to slice the data into multiple input data sequences with associated target values. For this process, the researcher uses a sliding windows algorithm. This algorithm moves a window step by step

through the time series data, adding a sequence of multiple data points to the input data with each step. In addition, the algorithm stores the target value (Closing Price) following this sequence in a separate target data set. Then the algorithm pushes the window one step further and repeats these activities. In this way, the algorithm creates a data set with many input sequences (mini-batches), each of which has a corresponding target value in the target record. The process was applied on both the creation of the training and the test data. Applying a sliding window approach to our data, the result was a training set (x_train) that contains 9818 input sequences, and each has 30 time-steps and four features. The corresponding target dataset (y_train) contains 9818 target values.

4.1.2.2 Hyperparameters Selection

Random forest regressor was used to obtain the number of layers, units, activation function, epoch, and batch size directly from the algorithm and re-fit to the LSTM model using only the most preferable values before fitting. The input layers had four input attributes, namely min-price, opening price, max-price, and production year which are connected to the hidden layer. The second layer is the hidden or activation layer with a different number of neurons. A sigmoid activation function is used. The hidden layer computes each input attribute and is connected to the outputs layer. The third layer is the output layer. The important parameters which have been used throughout the LSTM modeling process are shown in Table 2.

Table 4. 2 LSTM parameters

No	Parameter	Data/Technique Used
1	Number of input neuron(s)	100
2	Number of output neuron(s)	1
3	Transfer function(s)	Sigmoid transfer function ()
4	Error function(s)	Mean squared error(MSE) function

5	Neuron(s)	50, 50, 50, 50
6	Epoch	46
7	Batch size	164

As shown in table 4.1, the result of applying different attribute selection method, we have obtained four choices of dataset‘s attributes (min-price, opening price, max-price and production year), out of all the available attributes. Only the most four influencing variables (min-price, opening price, max-price and production year) were used that are highly important on the haricot bean price prediction. These attributes resulting from Information gain in feature selection experimentation are fitted to the neural network and trained using the LSTM model. Figures 4.1 show the experimental result for LSTM. The figures from the result show how the predicted value is closer to actual values.

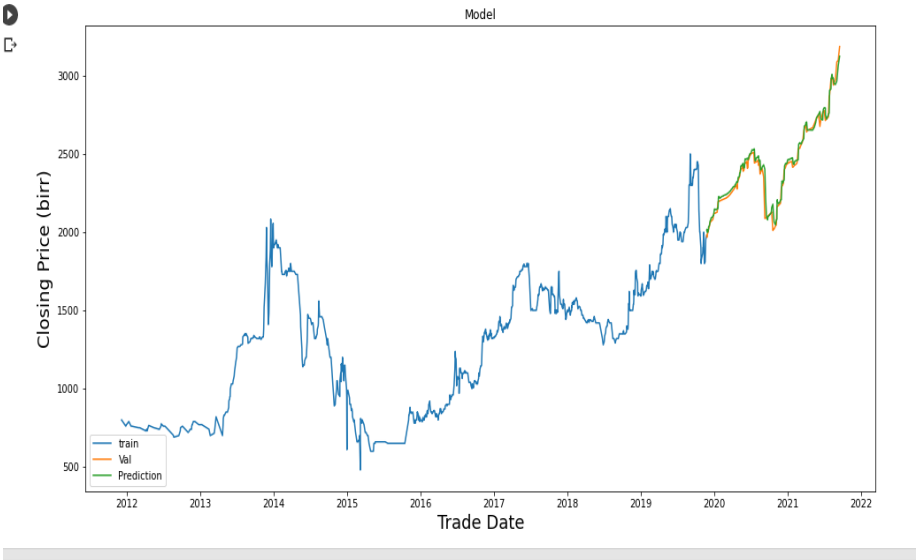


Figure4. 1 Trains, actual and predicted values LSTM model

Table 4. 3 Actual and predicted value (LSTM)

NO	Date	Actual Price of Haricot Bean	Predicted Price of Haricot Bean
1	1-Jul-21	2775	2783.32
2	2-Jul-21	2777	2783.14

3	2-Jul-21	2777	2781.54
4	8-Jul-21	2714	2712.63
5	8-Jul-21	2714	2724.72
6	9-Jul-21	2717	2725.33
7	9-Jul-21	2717	2721.91
8	14-Jul-21	2723	2729.4
9	15-Jul-21	2725	2731.53
10	16-Jul-21	2727	2737.43
11	16-Jul-21	2727	2739.72
12	22-Jul-21	2734	2736.12
13	23-Jul-21	2737	2731.36
14	27-Jul-21	2869	2877.25
15	28-Jul-21	2871	2879.88
16	29-Jul-21	2874	2880.28
17	29-Jul-21	2874	2876.67
18	3-Aug-21	2971	2981.44
19	5-Aug-21	2979	2983.98
20	6-Aug-21	2984	2986.11
21	10-Aug-21	2977	2983.32

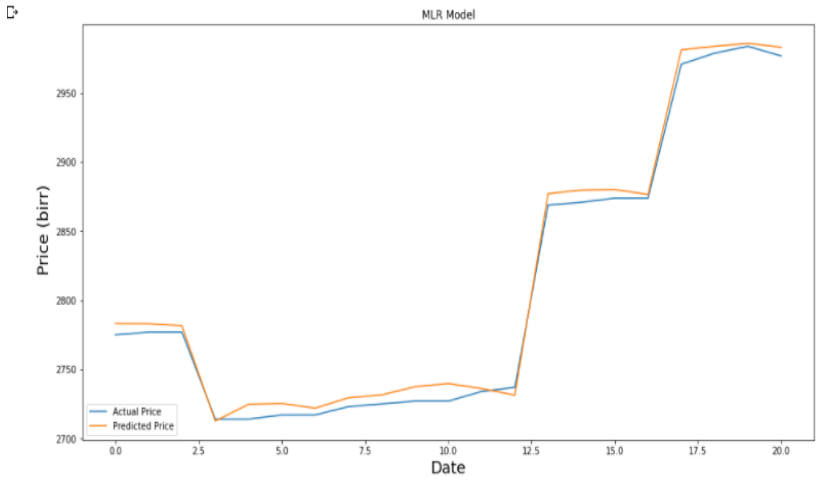


Figure 4. 2 Predicting future 20 days (LSTM)

4.1.3 Multiple Linear Regression

The second model used by the researcher was multiple linear regression. To perform Multiple Linear Regression the researcher imported LinearRegression from sklearn.linear_model. Figure 7, shows the actual and predicting values of Haricot bean closing price using multiple linear regression. The predicted values are close to the actual values.

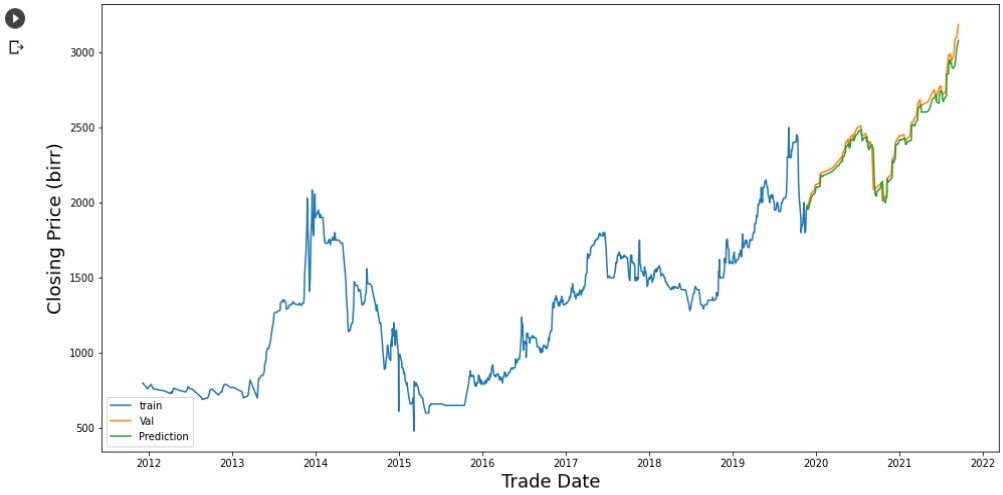


Figure 4. 3 Trains, actual and predicted values MLR model

Table 4.4 actual and predicted price (MLR)

No	Date	Actual Price of Haricot Bean	Predicted Price of Haricot Bean
1	1-Jul-21	2775	2783.32
2	2-Jul-21	2777	2788.77

3	2-Jul-21	2777	2784.11
4	8-Jul-21	2714	2723.45
5	8-Jul-21	2714	2722.32
6	9-Jul-21	2717	2727.23
7	9-Jul-21	2717	2724.85
8	14-Jul-21	2723	2729.23
9	15-Jul-21	2725	2734.89
10	16-Jul-21	2727	2737.57
11	16-Jul-21	2727	2734.99
12	22-Jul-21	2734	2743.65
13	23-Jul-21	2737	2747.55
14	27-Jul-21	2869	2874.67
15	28-Jul-21	2871	2874.51
16	29-Jul-21	2874	2882.98
17	29-Jul-21	2874	2883.02
18	3-Aug-21	2971	2975.71
19	5-Aug-21	2979	2989.09
20	6-Aug-21	2984	2990.66
21	10-Aug-21	2977	2985.72

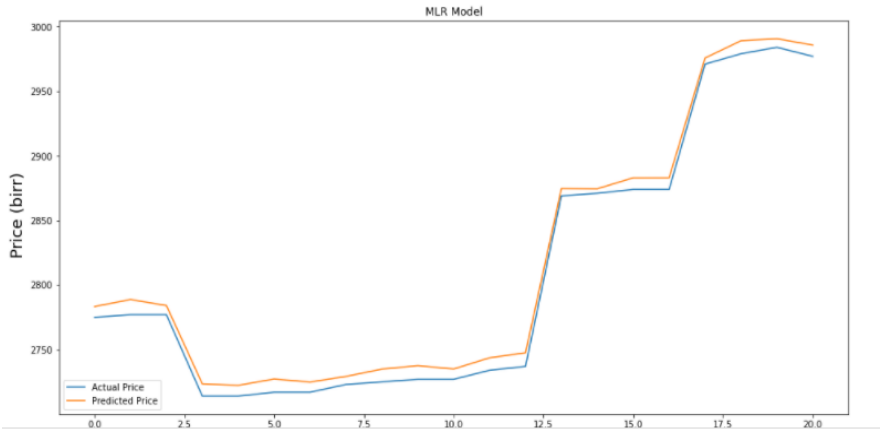


Figure4. 4 Predicting futures 20 days (MLR)

4.1.4 Simple Linear regression

The third model used by researcher was Simple Linear regression in order to obtain the predicting Haricot bean price. Linear regression is utilized, science, it helps identify the relationships between a dependent variable and independent variables. Simple linear regression is defined by using a feature to predict an outcome. Figure 4.5, shows the actual and predicting values of haricot bean closing price using simple linear regression

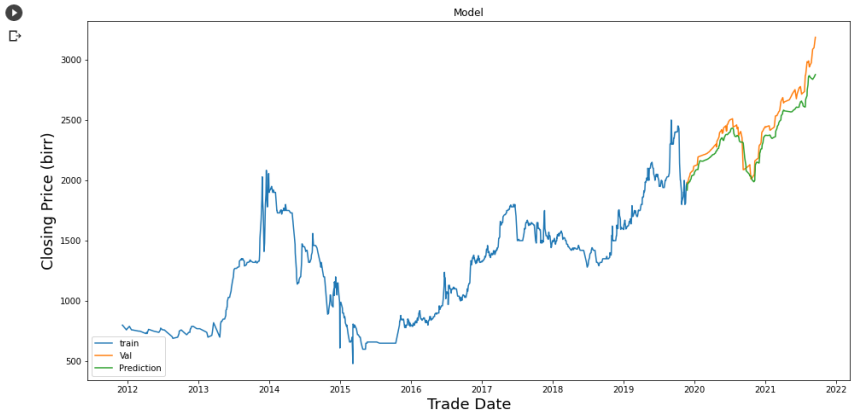


Figure 4. 5 Trains, actual and predicted values SLR model

Table4. 5 Actual and predicted values (SLR)

No	Date	Actual Price of Haricot Bean	Predicted Price of Haricot Bean
1	1-Jul-21	2775	2790.34
2	2-Jul-21	2777	2789.32
3	2-Jul-21	2777	2794.48

4	8-Jul-21	2714	2728.28
5	8-Jul-21	2714	2720.08
6	9-Jul-21	2717	2727.99
7	9-Jul-21	2717	2730.11
8	14-Jul-21	2723	2731.77
9	15-Jul-21	2725	2732.34
10	16-Jul-21	2727	2739.36
11	16-Jul-21	2727	2741.55
12	22-Jul-21	2734	2745.55
13	23-Jul-21	2737	2747.68
14	27-Jul-21	2869	2881.63
15	28-Jul-21	2871	2884.01
16	29-Jul-21	2874	2888.53
17	29-Jul-21	2874	2889.84
18	3-Aug-21	2971	2986.66
19	5-Aug-21	2979	2996.23
20	6-Aug-21	2984	3002.67
21	10-Aug-21	2977	2994.67

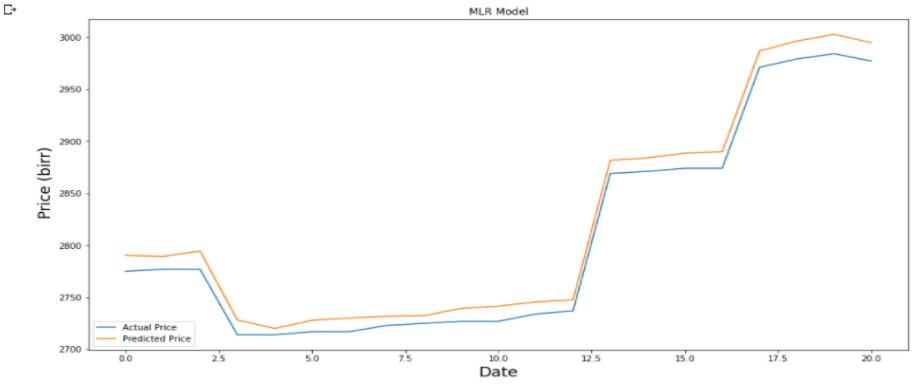


Figure 4. 6 Predicting futures 20 days (SLR)

4.2 Performance Evaluation of the Predictive Algorithm

The researcher used, percentage split validation performance evaluation techniques for comparison between the predicting models.

4.2.1 Percentage Split Validation (80%Training and 20% Testing)

The dataset we have obtained splits in to two, training set and testing set. The training set had 80% of the data and the remaining 20% of the data were testing purpose. The underlined table had shown the comparison of the model with percentage split validation.

Table 4. 6 Comparison of predictive algorithms

Models	**Metrics**		
	MAE	**MAPE**	**R squared**
LSTM	**0.0320**	**0.0150**	**0.97**
MLR	0.0711	0.0321	0.95
SLR	0.1280	0.0540	0.92

Table 4.5 presented the comparison of three predictive algorithms using percentage split validation metric to select the best predictive models. The evaluated result shows, LSTM algorithms is the best model compared to linear regression and multiple linear regression algorithms. The model found the maximum R squared 0.97, the lower MAPE 0.015, and the minimum MAE 0.032. Multiple linear regression is the second best algorithm. In multiple linear regression, the experimental performance evaluation founds the second maximum R squared 0.95, the second lower MAPE 0.0321and the minimum MAE 0.0711. Simlelinear regression algorithms comes the third in the experimental evaluation results compared to the two algorithms. The algorithm found the least R squared 0.92, the higher MAPE 5.74 4.31 and the MAE 0.182

Based on the experiment result in MAE, MAPE and R squared LSTM algorithms is the best and Multiple linear regression the second and simple linear regression is the third or the least algorithms compared between them.

CHAPTER 5

5 Conclusion and Future work

5.1 Conclusion

Globalization and the interaction between the various world markets have led to the need for an adequate decision-making process. Therefore, an effort is needed towards the design of intelligent systems in order to apply emerging technologies such as deep learning to provide reliable forecasting information that can be used by farmers, retailers and policy makers so that they can make decisions about production, marketing and policies progress.

Regulatory market conditions greatly affect the price of haricot bean. Considering the price of haricot bean in the future, agricultural authorities can reduce price variations and, consequently, reduce the high risk present in the market for product and ultimately increase the interests of producers and consumers. In fact, this can help to make the right decision by identifying and showing the future condition of prices in this sector at different times.

The main advantage of time series forecasting is that it only needs data from the time series being analyzed; this feature is useful for forecasting a large number of price data. The performance of model estimation largely depends on the data, and if that data has some missing values, then there are constraints on which the model can be affected. This study compared the SLR, MLR and LSTM models for forecasting using ECX haricot bean price data.

The researcher followed a sequence of steps for building a prediction model. Such as data collection, feature selection, data preprocess and data transformation. Regression predictor was used to obtain feature importance directly from the algorithm and fit selected features to the model and randomforest Regressor is used to obtain number of layer, units, activation function, epoch and batch size directly from the algorithm and then re-fit using only the most preferable values and the most important features selected by feature selection model are opening price, max price min price and closing price. Three models were applied namely, SLR, MLR and LSTM. The performances of the models are evaluated by percentage split validation performance metrics. The results illustrate that R squared value of 0.97 for LSTM, 0.95 for MLR and 0.92 for SLR. The results showed that LSTM outperformed other predictive models in all measures of model performance for predicting the Haricot Bean prices by achieving a coefficient of determination (R^2) of 0.97, mean absolute percentage error (MAPE) of 0.015, and mean absolute error (MAE) of 0.032.

5.2 Recommendations and Future work

Price prediction is a key factor, so it is important to have up-to-date information by looking at market conditions in future research and also to use different prediction models such as the deep learning model to reduce the risk. Future research should explore the possibility of using a hybrid model to predict haricot bean prices and to increase forecast accuracy research should explore more attributes such as global market demand, weather conditions and global bean price information.

Finally, further research is needed to determine how long a trained deep learning system remains valid and effective in predicting before it is found to need retraining.

6 Reference

Bulti, A. G., & Ray, A. (2019). Commodity Market Price Analysis and Prediction using Machine Learning Framework. 8, 2822–2828.

Exchange, C. (2017). Creating Agricultural Markets : How the Ethiopia Commodity Exchange Connects Farmers and Buyers through Partnership and Technology.

Hernandez, B. M. A., Lemma, S., & Rashid, S. (2013). The Ethiopian Commodity Exchange and the coffee market :

Hiransha, M., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2018). NSE Stock Market Prediction Using Deep-Learning Models. Procedia CoShen, J., & Shafiq, M. O. (2020). Short-Term Stock Market Price Trend Prediction Using a Comprehensive Deep Learning System. Journal of Big Data, 7(1).

Jayanth Balaji, A., Harish Ram, D. S., & Nair, B. B. (2018). Applicability of deep learning models for stock price forecasting an empirical study on bankex data. Procedia Computer Science, 143, 947–953.

Ji, X., Wang, J., & Yan, Z. (2021). A stock price prediction method based on deep learning technology. 5(1), 55–72.

Lachiheb, O., & Gouider, M. S. (2018). A hierarchical Deep neural network design for stock returns prediction. Procedia Computer Science, 126, 264–272.

Worku, T (2015). Predicting the market status of coffee, pea beans and sesame : a case of ethiopian commodity exchange (ecx).

Gebisa, A. Y. (2019). Coffee price pridiction using machine-learning techniques: a case of ethiopian commodity exchange (ecx).

Shen, J., & Shafiq, M. O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. Journal of Big Data,

Hochreiter S, Schmidhuber J . LSTM can solve hard long time lag problems. In: Advances in neural information processing systems; 1997.

FAO. (2015). Analysis of Price Incentives for Haricot Beans in Ethiopia for the Time Period Analysis of Price Incentives for Haricot Beans in Ethiopia for the Time Period. February, 58p.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014).

Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Re- search 15(1): 1929-1958.

Saud, A. S., & Shakya, S. (2020). Analysis of look back period for stock price prediction with RNN variants: A case study on banking sector of NEPSE. Procedia Computer Science, 167(2019)

Hu, Z., Zhao, Y., & Khushi, M. (2021). A Survey of Forex and Stock Price Prediction Using Deep Learnin

Han, J. & Kamber, M. (2006). Data Mining: Concepts and Techniques, Second Edition. Elsevier Inc

Widegren, P. (2017). Deep learning-based forecasting of financial assets.

Yang, C., Zhai, J., & Tao, G. (2020). Deep Learning for Price Movement Prediction Using Convolutional Neural Network and Long Short-Term Memory. 2020.

Aamodt, T. (2015). Predicting Stock Markets with Neural Networks.

Khadka, S. (2018). STOCK PRICE PREDICTION OF NEPAL USING. I, 61–65.

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock price prediction using the ARIMA model. Proceedings - UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, UKSim 2014, June, 106–112.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.

Olah, C. (2015). Understanding LSTM Networks [Blog]. Web Page, 1–13. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 404(March), 1–43.

Apt, W. (2014). Introduction. *Demographic Research Monographs*, 1–13.

Abaimov, S., & Martellini, M. (2022). Understanding Machine Learning. In *Advanced Sciences and Technologies for Security Applications*.

Witten, I. H., Frank, E., & Hall, M. a. (2011). Data Mining: Practical Machine Learning Tools and Techniques (Google eBook).

7 APPENDIX:

Codes Used in Google co-laboratory

1 Basic imported libraries

```
import math # Mathematical functions

import numpy as np # Fundamental package for scientific computing with Python

import pandas as pd # Additional functions for analysing and manipulating data

from datetime import date, timedelta, datetime # Date Functions

import matplotlib.pyplot as plt # Important package for visualization we use this to plot the data

import matplotlib.dates as mdates # Formatting dates

from sklearn.metrics import mean_absolute_error# Packages for measuring model performance /
 errors

from keras.models import Sequential # Deep learning library, used for neural networks

from keras.layers import LSTM, Dense, Dropout # Deep learning classes for recurrent and regula
r densely-connected layers

from keras.callbacks import EarlyStopping # EarlyStopping during model training

from sklearn.preprocessing import RobustScaler, MinMaxScaler # This Scaler removes the medi
an and scales the data according to the quantile range to normalize the price data

import seaborn as sns

from keras.layers.recurrent_v2 import lstm_with_backend_selection# Configure the neural netw
ork model

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import RandomForestRegressor
```

```
import math # Mathematical functions
import numpy as np
import pandas as pd
from datetime import date, timedelta, datetime
from pandas.plotting import register_matplotlib_converters #
import matplotlib.pyplot as plt #
import matplotlib.dates as mdates
from sklearn.metrics import
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import  MinMaxScaler
```

2 Importing data-set and pre-processing the data

importing data

from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():

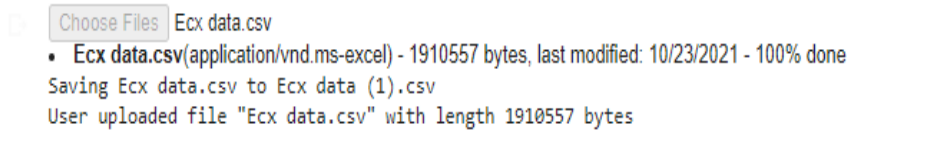
  print('User uploaded file "{name}" with length {length} bytes'.format(

      name=fn, length=len(uploaded[fn])))

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```



3 pre-processing

Convert the data to numpy values

np_data_unscaled = np.array(data_filtered)

np_data = np.reshape(np_data_unscaled, (nrows, -1))

print(np_data.shape)

4 Transform the data by scaling each feature to a range between 0 and 1

scaler = MinMaxScaler()

np_data_scaled = scaler.fit_transform(np_data_unscaled)

5 Creating a separate scaler that works on a single column for scaling predictions

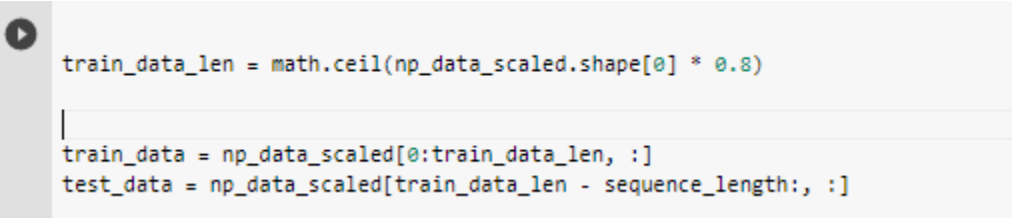
scaler_pred = MinMaxScaler()

df_Close = pd.DataFrame(data_filtered_ext['Closing Price'])

np_Close_scaled = scaler_pred.fit_transform(df_Close)

6 Split the training data into train and test data sets

train_data_len = math.ceil(np_data_scaled.shape[0] * 0.8)



7 Create the training and test data

```
train_data = np_data_scaled[0:train_data_len, :]

test_data = np_data_scaled[train_data_len - sequence_length:, :]
```

8 setting the timeframe

```
def partition_dataset(sequence_length, data):
    x, y = [], []
    data_len = data.shape[0]
    for i in range(sequence_length, data_len):
        x.append(data[i-sequence_length:i,:]) #contains sequence_length values 0-
sequence_length * columsn
        y.append(data[i, index_Close])
```

9 Generating batch_size, epochs and num_layers

```
def build_model(hp):
    model = tf.keras.Sequential()
    batch_size=hp.Int('batch_size',
                      min_value=10,
                      max_value=100)
    epochs=hp.Int('epochs',
                      min_value=50,
                      max_value=100)
    for i in range(hp.Int('num_layers', 2, 8)):
        model.add(tf.keras.layers.Dense(units=hp.Int('units_' + str(i), 50,100, step=10),
```

```
            activation=hp.Choice('act_' + str(   ['relu', 'sigmoid'])))
    model.add(tf.keras.layers.Dense(1, activation='softmax'))
    model.compile(
        'adam',
        loss="mean_squared_error",
        metrics=[keras.metrics.MeanAbsoluteError()],
        )
    return model



tuner = kt.RandomSearch(
    build_model,
    objective=kt.Objective("val_mean_absolute_error", direction="min"),
    max_trials=5,
    executions_per_trial=3,
     overwrite=True,
    directory='Solen',
    project_name='Closing Price')
```

10 Configure the neural network model

model = Sequential()

model.add(LSTM(100, return_sequences=False, input_shape=(x_train.shape[1, x_train.shape[2])))

#model.add(LSTM(50, return_sequences=True))

#model.add(LSTM(50, return_sequences=True))

#model.add(LSTM(50, return_sequences=True))

#model.add(LSTM(50, return_sequences=False))

#model.add(Dense(5))

model.add(Dense(1))

```
from keras.layers.recurrent_v2 import lstm_with_backend_selection
# Configure the neural network model
model = Sequential()

# Model with n_neurons = inputshape Timestamps, each with x_train.shape[2] variables
n_neurons = x_train.shape[1] * x_train.shape[2]
print(n_neurons, x_train.shape[1], x_train.shape[2])
model.add(LSTM(80, return_sequences=True,  input_shape=(x_train.shape[1], x_train.shape[2])))
model.add(LSTM(60, return_sequences=True))
model.add(LSTM(90, return_sequences=True))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(1))

# Compile the model
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
model.summary()
```

```
Epoch 1/70
42/42 [==============================] - 10s 81ms/step - loss: 0.0088 - val_loss: 0.0016
Epoch 2/70
42/42 [==============================] - 2s 42ms/step - loss: 9.3840e-04 - val_loss: 0.0022
Epoch 3/70
42/42 [==============================] - 2s 42ms/step - loss: 8.7435e-04 - val_loss: 0.0019
Epoch 4/70
42/42 [==============================] - 2s 42ms/step - loss: 8.4598e-04 - val_loss: 0.0021
Epoch 5/70
42/42 [==============================] - 2s 42ms/step - loss: 7.9289e-04 - val_loss: 0.0017
Epoch 6/70
42/42 [==============================] - 2s 42ms/step - loss: 7.4242e-04 - val_loss: 0.0011
Epoch 7/70
42/42 [==============================] - 2s 42ms/step - loss: 7.4297e-04 - val_loss: 0.0012
Epoch 8/70
42/42 [==============================] - 2s 42ms/step - loss: 6.3534e-04 - val_loss: 0.0013
Epoch 9/70
42/42 [==============================] - 2s 42ms/step - loss: 6.3574e-04 - val_loss: 0.0014
Epoch 10/70
42/42 [==============================] - 2s 42ms/step - loss: 5.8321e-04 - val_loss: 9.0375e-0
Epoch 11/70
42/42 [==============================] - 2s 42ms/step - loss: 5.2803e-04 - val_loss: 8.8050e-0
Epoch 12/70
42/42 [==============================] - 2s 42ms/step - loss: 5.2874e-04 - val_loss: 9.9510e-0
Epoch 13/70
42/42 [==============================] - 2s 42ms/step - loss: 5.0712e-04 - val_loss: 9.4164e-0
Epoch 14/70
42/42 [==============================] - 2s 42ms/step - loss: 4.4718e-04 - val_loss: 0.0013
Epoch 15/70
42/42 [==============================] - 2s 42ms/step - loss: 5.2339e-04 - val_loss: 0.0011
```

```
Epoch 16/70
42/42 [==============================] - 2s 42ms/step - loss: 4.0850e-04 - val_loss: 0.0011
Epoch 17/70
42/42 [==============================] - 2s 42ms/step - loss: 4.0302e-04 - val_loss: 0.0011
Epoch 18/70
42/42 [==============================] - 2s 42ms/step - loss: 3.8445e-04 - val_loss: 0.0011
Epoch 19/70
42/42 [==============================] - 2s 42ms/step - loss: 4.5336e-04 - val_loss: 8.7861e-04
Epoch 20/70
42/42 [==============================] - 2s 42ms/step - loss: 3.8974e-04 - val_loss: 9.6061e-04
Epoch 21/70
42/42 [==============================] - 2s 42ms/step - loss: 3.8329e-04 - val_loss: 0.0014
Epoch 22/70
42/42 [==============================] - 2s 42ms/step - loss: 3.7503e-04 - val_loss: 9.3033e-04
Epoch 23/70
42/42 [==============================] - 2s 42ms/step - loss: 3.5319e-04 - val_loss: 9.6394e-04
Epoch 24/70
42/42 [==============================] - 2s 42ms/step - loss: 3.5202e-04 - val_loss: 0.0011
Epoch 25/70
42/42 [==============================] - 2s 42ms/step - loss: 3.8465e-04 - val_loss: 0.0012
Epoch 26/70
42/42 [==============================] - 2s 42ms/step - loss: 3.6559e-04 - val_loss: 0.0010
Epoch 27/70
42/42 [==============================] - 2s 42ms/step - loss: 3.2970e-04 - val_loss: 0.0010
Epoch 28/70
42/42 [==============================] - 2s 42ms/step - loss: 3.4902e-04 - val_loss: 9.3056e-04
Epoch 29/70
42/42 [==============================] - 2s 42ms/step - loss: 3.1904e-04 - val_loss: 0.0011
```

```
Epoch 30/70
42/42 [==============================] - 2s 42ms/step - loss: 3.2751e-04 - val_loss: 9.8779e
Epoch 31/70
42/42 [==============================] - 2s 42ms/step - loss: 3.2953e-04 - val_loss: 9.9123e
Epoch 32/70
42/42 [==============================] - 2s 42ms/step - loss: 3.2212e-04 - val_loss: 9.8577e
Epoch 33/70
42/42 [==============================] - 2s 42ms/step - loss: 3.4283e-04 - val_loss: 0.0012
Epoch 34/70
42/42 [==============================] - 2s 42ms/step - loss: 3.4423e-04 - val_loss: 0.0010
Epoch 35/70
42/42 [==============================] - 2s 42ms/step - loss: 3.3562e-04 - val_loss: 0.0011
Epoch 36/70
42/42 [==============================] - 2s 42ms/step - loss: 3.1780e-04 - val_loss: 0.0010
Epoch 37/70
42/42 [==============================] - 2s 42ms/step - loss: 3.3046e-04 - val_loss: 0.0010
Epoch 38/70
42/42 [==============================] - 2s 42ms/step - loss: 3.1324e-04 - val_loss: 0.0013
Epoch 39/70
42/42 [==============================] - 2s 42ms/step - loss: 3.0037e-04 - val_loss: 0.0012
Epoch 40/70
42/42 [==============================] - 2s 42ms/step - loss: 2.9676e-04 - val_loss: 0.0011
Epoch 41/70
42/42 [==============================] - 2s 47ms/step - loss: 3.0012e-04 - val_loss: 0.0012
Epoch 42/70
42/42 [==============================] - 2s 47ms/step - loss: 3.2615e-04 - val_loss: 0.0011
Epoch 43/70
42/42 [==============================] - 2s 44ms/step - loss: 3.2127e-04 - val_loss: 0.0013
Epoch 44/70
42/42 [==============================] - 2s 42ms/step - loss: 3.0368e-04 - val_loss: 0.0011
Epoch 45/70
42/42 [==============================] - 2s 42ms/step - loss: 3.0344e-04 - val_loss: 0.0011
Epoch 46/70
42/42 [==============================] - 2s 42ms/step - loss: 3.1141e-04 - val_loss: 0.0011
Epoch 47/70
42/42 [==============================] - 2s 42ms/step - loss: 3.3249e-04 - val_loss: 0.0011
Epoch 48/70
42/42 [==============================] - 2s 42ms/step - loss: 3.1042e-04 - val_loss: 0.0013
Epoch 49/70
42/42 [==============================] - 2s 42ms/step - loss: 3.0251e-04 - val_loss: 0.0013
Epoch 50/70
42/42 [==============================] - 2s 42ms/step - loss: 2.9239e-04 - val_loss: 0.0011
Epoch 51/70
42/42 [==============================] - 2s 42ms/step - loss: 3.0096e-04 - val_loss: 0.0011
Epoch 52/70
42/42 [==============================] - 2s 42ms/step - loss: 2.9846e-04 - val_loss: 0.0012
Epoch 53/70
42/42 [==============================] - 2s 42ms/step - loss: 3.0208e-04 - val_loss: 0.0017
Epoch 54/70
42/42 [==============================] - 2s 42ms/step - loss: 3.1754e-04 - val_loss: 0.0011
Epoch 55/70
42/42 [==============================] - 2s 42ms/step - loss: 2.9878e-04 - val_loss: 0.0014
Epoch 56/70
42/42 [==============================] - 2s 42ms/step - loss: 3.1700e-04 - val_loss: 0.0011
Epoch 57/70
```

```
Epoch 58/70
42/42 [==============================] - 2s 42ms/step - loss: 3.0138e-04 - val_loss: 0.0012
Epoch 59/70
42/42 [==============================] - 2s 42ms/step - loss: 3.0566e-04 - val_loss: 0.0021
Epoch 60/70
42/42 [==============================] - 2s 42ms/step - loss: 2.9806e-04 - val_loss: 0.0013
Epoch 61/70
42/42 [==============================] - 2s 42ms/step - loss: 3.2749e-04 - val_loss: 0.0015
Epoch 62/70
42/42 [==============================] - 2s 42ms/step - loss: 3.3092e-04 - val_loss: 0.0013
Epoch 63/70
42/42 [==============================] - 2s 42ms/step - loss: 2.9400e-04 - val_loss: 0.0013
Epoch 64/70
42/42 [==============================] - 2s 42ms/step - loss: 2.9510e-04 - val_loss: 0.0015
Epoch 65/70
42/42 [==============================] - 2s 42ms/step - loss: 3.2078e-04 - val_loss: 0.0015
Epoch 66/70
42/42 [==============================] - 2s 42ms/step - loss: 2.9284e-04 - val_loss: 0.0012
Epoch 67/70
42/42 [==============================] - 2s 42ms/step - loss: 2.8957e-04 - val_loss: 0.0012
Epoch 68/70
42/42 [==============================] - 2s 42ms/step - loss: 3.0848e-04 - val_loss: 0.0015
Epoch 69/70
42/42 [==============================] - 2s 42ms/step - loss: 2.9607e-04 - val_loss: 0.0012
Epoch 70/70
42/42 [==============================] - 2s 42ms/step - loss: 2.9820e-04 - val_loss: 0.0011
```

11 Compile the model

```
model.compile(optimizer = 'adam', loss = 'mean_squared_error')

model.summary()
```