Crypto Algorithmic Trading Bot with a Comparative study of Deep Learning Models

Sidthaarth Gnanasekar
Student Number 200903651
MSc. FT Data Science and Artificial Intelligence, QMUL
s.gnanasekar@se20.qmul.ac.uk
Project Supervisor: Professor Jiadong Yu
jiadong.yu@qmul.ac.uk

***Abstract*—Our research focuses on the challenges of short-term cryptocurrency time series forecasting using a Deep Neural Network (DNN) and Machine Learning (ML) algorithms. Numerous traders participate in intraday trading, which entails trading Bitcoin against the dollar bill (USD) on a relatively lower timeframe to squeeze their profit even from slight market swings. High-frequency trading, which takes advantage of spreads and arbitrage possibilities to create modest profits, has traditionally been the domain of algorithmic trading. It is hard to forecast the future price of any crypto pair due to the tremendous volatility of the crypto market. This research shows that introducing domain knowledge into the process of training deep learning and machine learning models improves crypto price prediction significantly. Only a few studies have been published on a deep neural network to create a price prediction model. Using deep learning techniques, this project aims at building different powerful and sophisticated models that forecast price fluctuations in the cryptocurrency market. Furthermore, the accuracy of the metric of these models is compared, and the best model is fed to an algo bot in the binance testnet account via API, which makes trades on its own on a lower timeframe, buying and selling bitcoin pair based on the prediction of the best-trained model for the 1-minute timeframe. The tested models include Autoregressive Integrated Moving Average (ARIMA), Convolutional Neural Network (CNN), Bi-Directional Long Short-Term Memory Network (LSTM), and CNN-LSTM Neural Networks.**

I. Introduction

When it comes to stock purchases and investments, our emotions frequently play a role. [1] examined the relationship between shopping environment and buying behavior and discovered that ”pleasantness felt within a retail store environment can have a significant impact on purchase.” This is also true in the case of stock markets. According to [2], ”asset prices and sentiment impacts and correlate positively with investor sentiment.” Investor overconfidence leads to increased trading aggression and overpricing, and individual investors prefer to build their investing methods rather than indexing in the hopes of outperforming the market, says [3].

People have been attempting to anticipate future prices of stocks and assets since the first stock market opened in Amsterdam in 1653 [4]. The stock market has survived numerous bursting booms and financial disasters. Because of the recent development of cryptocurrencies, a new form of market where these cryptocurrencies can be traded has emerged. Apart from a few exceptions, these exchanges operate very much like traditional stock exchanges. Normal stock exchanges are only open for a few limited hours during working days, whereas cryptocurrency exchanges are open around the clock, seven days a week. Unlike other stock exchanges, cryptocurrencies provide an additional feature that they can be purchased by anybody in the world, hence called ”decentralized money.”

For the past few months, bitcoin and cryptocurrencies have been a hot topic in the news. As cryptocurrency markets get extensive media attention and experience exponential growth, creating a model that can predict price changes becomes both intriguing and potentially profitable. In contrast to equities, the cryptocurrency market is one of the most complex because of its high volatility, nonlinearity, and irregularity.

Even, as humans we can’t predict cryptocurrency market price for every minute and based on that prediction and strategy humans can’t handle multiple trades at once, such as performing 1000 trades with buying and sell signals on lower timeframe. Humans find it challenging to execute several trades on the lower timeframe at different entry and exit points since the spreads and arbitrage are highly volatile in the cryptocurrency market. This is a kind of advantage for our bot to make a profit by taking many trades during that slight market fluctuations. Constructing a model with high accuracy as possible is to feed the bot that can make us money and outperform humans on the lower timeframe is the goal.

BTCUSD was chosen as the cryptocurrency pair to be analyzed in this study because reliable data availability is one of the main reasons. BTCUSD, basically all cryptocurrency pairs are also helpful in algorithmic trading bot implementation in Binance Testnet account via API.

Binance Chain is a decentralized exchange (DEX) for digital assets that were created by Binance and its community. The testnet is a developer-only test environment for the Binance Chain network, managed by the Binance Chain development community.

Recent advance in machine learning, specifically deep learning, and the use of recurrent neural networks (RNNs), which are based on the Artificial Neural Network paradigm, have created a new type of algorithmic trading where Artificial Intelligence algorithms use deep learning models to predict price movements. Four different models are trained and tested with various hyperparameters. These models aren’t just for analyzing BTCUSD data, they can also be used to analyze any data from a time series and used for the forecast.

II. Problem Statement

In light of these challenges, predicting stock prices for the upcoming years, months, or even the day's closing stock price is becoming increasingly popular and readily available since the time series forecast is solved. However, in the one-minute timeframe, the ability to make accurate forecasts of one of the world's highly volatile cryptocurrency pairs (BTCUSD) could feasibly accelerate the pace of this research. This research builds on prior work [5] to make furthermore accurate model especially on the lower timeframe by providing a BTCUSD price dataset containing different variables, such as univariate, multivariate, and PCA computed variables, to the different neural network models and comparing their accuracy.

III. Contribution

We illustrate a novel approach; a new deep learning algorithm Bidirectional LSTM model of a univariate dataset is fed to an Algorithmic Trading Bot and performs the trades in binance testnet account via API based on the prediction of the Bidirectional LSTM model. If the predicted price is high than the current price of BTCUSD it buys the crypto pair and sells for some minimum profitable threshold price since our bot takes numerous trades, the minimum spread profit could be a considerable amount of profit on the wholescale. When the evaluation metrics of CNN, Bidirectional LSTM, ARIMA, and CNN-LSTM with the combination of univariate, multivariate, and PCA performed feature datasets are compared, it is evident that univariate Bidirectional LSTM has a higher forecasting accuracy and is probably more suitable for BTCUSD price forecasting on the lower timeframes. In this method, to analyze and predict the historical data, bidirectional LSTM uses a combination of forwarding and backward LSTM, which has the complete benefit of time sequence for the BTCUSD price data to provide more accurate forecasts.

IV. Related Work

Technology has changed the way financial markets and financial assets are exchanged. Two significant associated technology improvements are investors using computers to automate their trading operations and markets reconfiguring themselves so that virtually all markets are now electronic limit order books [6]. The speed and accessibility of such trading markets support the use of algorithmic trading to make trading decisions, place orders, and even handle those orders after they have been submitted [7].

In 2009, high-frequency algorithmic trading accounted for 60% of all US equities transactions, and it is a major driver of computing and analytics innovation [8], particularly in machine learning, deep learning, and grid/GPU computing. However, as the 6 May 2010 Flash Crash demonstrated, algorithmic trading is a major source of concern for algorithmic policymakers and traders. In this scenario, the Dow Jones Industrial Average dropped 600 points in 5 minutes, resulting in a 600 billion dollar fall in the market value of United States corporate equities [8]. This incident demonstrated is a lack of understanding of high-frequency algorithmic trading and its potential vulnerability. Defending against such occurrences needs a detailed understanding of the algorithmic trading process.

Consider the many types of trading, investigate how trade is completed on an exchange, and analyze the aims and obstacles to interpret algorithmic trading [9]. Cryptocurrency machine learning applications are a new field with limited research efforts. [10] created a Bayesian regression model for purchasing and selling Bitcoins. Another approach created by [11] predicted the price change of Bitcoin using random forests. Since Bitcoin data is time-series data, these approaches fail to take advantage of time sequences, but the recurrent neural network has a benefit in this scenario.

[12] illustrated two prediction models based on recurrent neural networks (RNNs) and long short-term memory (LSTM) for bitcoin prediction and compared them to a classic time series forecasting model, the autoregressive integrated moving average (ARIMA) model [13]. They created classification models based on Bitcoin price data that forecast whether the next Bitcoin price would rise or fall based on previous values. RNN and LSTM models were effective than the ARIMA model in the work of [12].

In 2018 [14] researched Bitcoin blockchain data, such as the number of Bitcoin wallets and unique addresses, block mining difficulty, hash rate, and so on, and built prediction models based on those features that are significantly related to the Bitcoin price.

Jang and Lee [15] examined macroeconomic factors such as the S&P 500, Dow 30, Euro Stoxx 50, and NASDAQ, as well as exchange rates between major fiat currencies, in addition to blockchain data. They focused on the three different prediction models: Bayesian neural networks (BNNs), linear regression, and support vector regression, and found that the BNN model excelled the other two. As a follow-up research, Jang et al. [16] developed a rolling window LSTM model and demonstrated that it outperformed prediction models based on linear regression, SVR, neural networks, and LSTM. Unfortunately, none of the previous studies considered any CNN-based bitcoin prediction on the lower timeframe and different novel deep learning model combinations. In addition, unlike earlier studies, we present detailed comparisons of several deep learning models, particularly in the lower timeframe.

V. Background

A. ARIMA (Autoregressive integrated moving average)

In 1970, Box and Jenkins introduced the ARIMA model. It's also known as the Box-Jenkins methodology, which consists of a set of techniques for detecting, estimating, and evaluating ARIMA models with time series data. The model is one of the most often used approaches in financial forecasting [17], [18]. The future value of a variable in an ARIMA model is a linear combination of previous values and past errors, written as:

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q} \quad (1)$$

Where, Y_t is the true value, Et is the random error at time t, ϕ_i and θ_j are the coefficients, and p and q are integers that are commonly referred to as autoregressive and moving average, respectively.

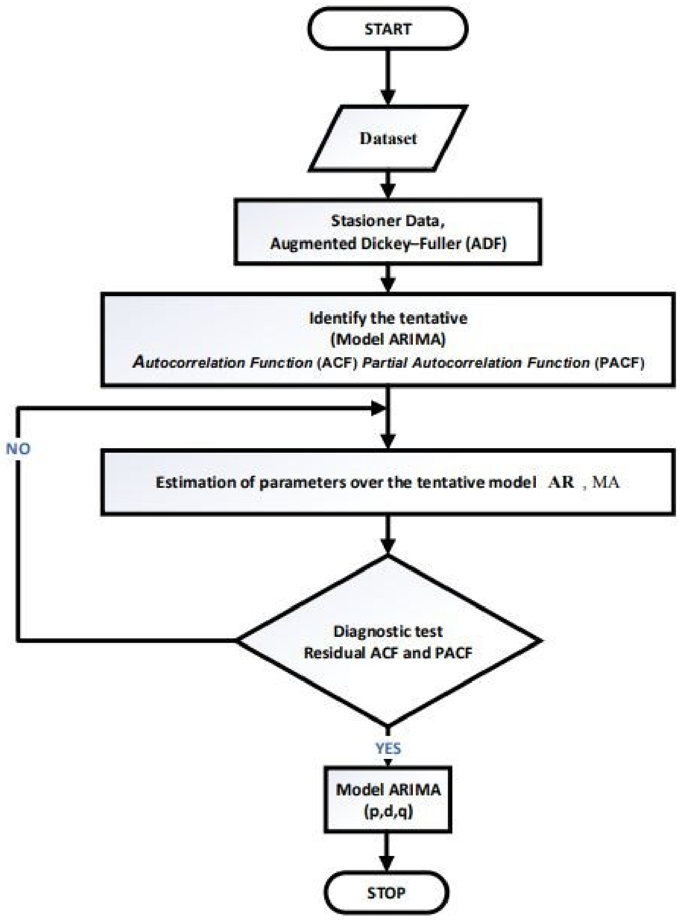

START
Dataset
Stasioner Data, Augmented Dickey–Fuller (ADF)
Identify the tentative (Model ARIMA) Autocorrelation Function (ACF) Partial Autocorrelation Function (PACF)
NO
Estimation of parameters over the tentative model AR, MA
Diagnostic test Residual ACF and PACF
YES
Model ARIMA (p,d,q)
STOP

Fig. 1. Representing flow of ARIMA model architecture. Adapted from [19].

ARIMA model performs better only if the time series data is stationary so, the first step in developing an ARIMA model is to make the data stationary if it's not stationary. Fig. 1 describes the Augmented Dickey-Fuller (ADF) test is used to determine whether data is stationary by analyzing the ADF test's output value, such as the t-test result, p-value, and so on. Then, ACF and PACF plots are generated that represent the value for the parameter of the ARIMA model that is (p, d, q). The most minimal AR (p) and moving average MA (q) parameter values are used to select the best model. In the case of AR (1) and AR (2), the optimal model according to the parsimony principle is AR (1). Model identification, parameter estimation, and diagnostic checking are the phases in creating an ARIMA predictive model [19].

B. CNN

Convolutional Neural Network (CNN) is the most extensively used deep learning approach, which is a sort of machine learning in which models directly recognize images, text, video, or sound. CNN uses data to learn, classifies images based on patterns, and eliminates the need for humans to extract features [20]. It can be used to forecast time series with great success. CNN's local perception and weight sharing can considerably minimize the number of parameters, enhancing model learning efficiency [21].

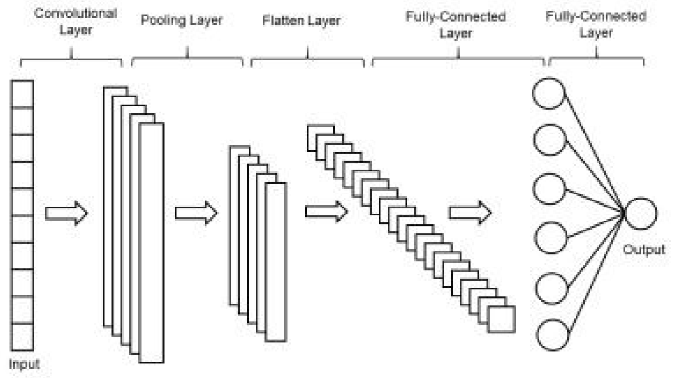


Fig. 2. Structure of one dimensional(1D) CNN. Adapted from [22].

Fig. 2 describes a typical CNN consists of multiple convolutional layers, a pooling layer, and a flattened layer. Each convolution layer has several convolution kernels, and the formula for calculating them is shown in formula (2)

$$l_t = \tanh(x_t * k_t + b_t) \quad (2)$$

where l_t is the convolution output value, tanh is the activation function, x_t is the input vector, k_t is the convolution kernel weight, and b_t is the convolution kernel bias.

The features of the data are retrieved are quite large after the convolution operation of the convolution layer, thus a pooling layer is added after the convolution layer to minimise the feature dimension and reduce the cost of training the network. Finally, as an output, one or more completely connected layers are added. A completely connected CNN layer is also called a multilayer perceptron with weights w_{ij} connecting all input units i to all output units j. No weights are added to the pooling layer.

C. Bidirectional LSTM

Traditional recurrent neural network (RNNs) attempts to tackle the problem of "lack of memory" in feed-forward neural networks, which is responsible for poor performance on sequences and time-series problems. Long Short-Term Memory (LSTM) neural networks [22] are advanced type of (RNN) that can learn long-term relationships through feedback connections. It interprets patterns in a text or a time series by processing data through a network of LSTM or hidden cells, also known as hidden or LSTM units, with an architecture like that is shown in the Fig. 1.

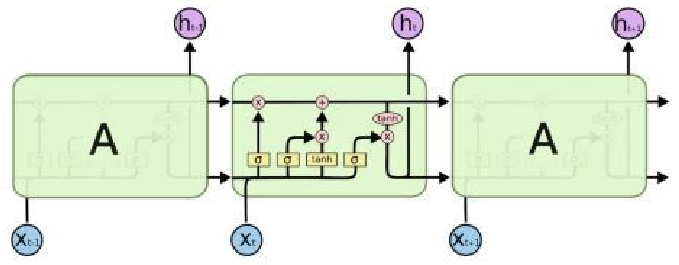


Fig. 3. LSTM unit network architecture. Adapted from [23].

Each LSTM cell in Fig. 3 is made up of different gates that use sigmoid and tanh as activation functions to select whether to forget or remember current and historical information. The output of the last cell h_{t-1} is combined with the information from the input x_t, which is generally a tensor of dimensions such as number of sequences, length of each sequence, number of variables in each sequence. As a result, the new tensor is passed to the forget gate, which is the first layer. Here it is decided if the output of the previous cell is still relevant to the present input that makes the impact on the current output layer h_t.

The input gate layer uses a tanh function with the input tensor as input and generates a candidate C_{it} for the current cell state. Finally, the current cell state is equal to the sum of the old and new candidates, weighted by the gate's sigmoid function output. To put it mathematically:

$$C_t = f_{forget} * C_{t-1} + f_{input} * C_{it} \tag{3}$$

The output gate ct is the cell's final component. These output gate values are calculated by a tanh function of the current cell state that is already multiplied by a sigmoid function with the input of the previous gate, in the same way, it is determined for the other gates also. A dense layer must be connected to the output layer to fully understand the LSTM network's output. The type of dense layer to use is determined by the application [24].

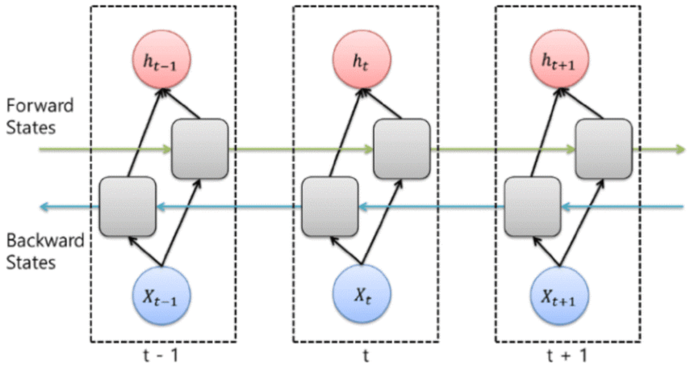


Fig. 4. Represents the model architecture of Bidirectional LSTM. Adapted from [25].

The results of RNN-based neural network models tend to reflect only the previous state because they are processed in chronological order. A Bi-directional Recurrent Neural Network (BRNN) is a technique for adding hidden layers in the reverse direction to both past data as well as the data to be processed in the future [26].

It has two different hidden layers that are not connected in any way, either forward or backward. Fig. 4 confirms When all input values are passed to the two hidden layers, both hidden layer output values are calculated, resulting in the final output. The forward direction is identical to that of a general RNN [25]. Input values are entered in the opposite way in the forward and backward hidden layers, and output layer values are determined after all inputs have been applied to hidden layers in both directions. Back Propagation Through Time is utilized in the same way as normal RNN when updating the weights in a reverse manner. Since it uses both previous and future data, the data learned in this manner is superior to the present unidirectional LSTM approach. [27] introduced a new deep learning model for the stock market prediction that integrated CNN with a BiLSTM artificial neural network model based on multivariate time series data.

D. *CNN-LSTM*

In an encoder-decoder design, a convolutional neural network, or CNN, can be deployed as the encoder. CNN is capable of reading across sequence input and automatically learning the prominent features, rather than directly supporting sequence input. These can then be decoded using an LSTM decoder as usual. CNN-LSTM models are hybrid models that combine a CNN and an LSTM, and we're putting them together in this case in an encoder-decoder architecture. Although numerous features are read as separate channels that eventually have the same impact, the CNN needs the input data to have the same 3D structure as the LSTM model [28].

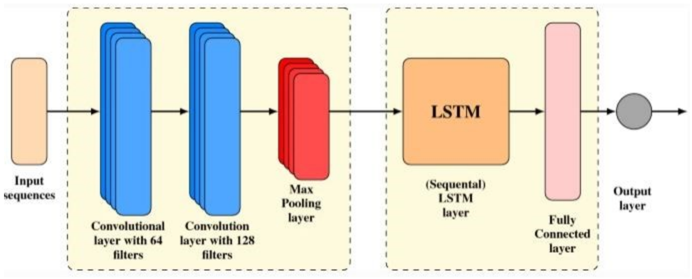


Fig. 5. Representing CNN–LSTM model architecture; Two convolutional layers, a pooling layer, an LSTM layer, a fully connected layer, and an output layer. Adapted from [29].

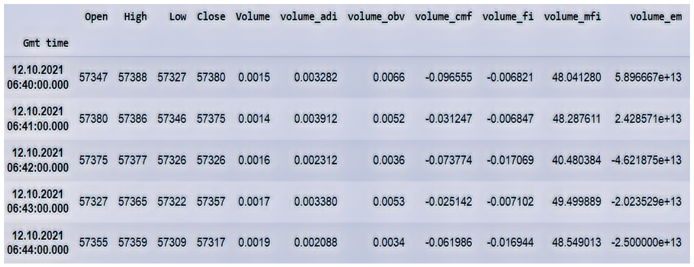
The convolutional layers perform a convolution between the raw input data and convolution kernels, resulting in new feature values. Because this technique was designed to extract features from image datasets, the input data must be in structured matrix form [30]. Fig. 5 confirms Multiple convolved features can be formed by applying different convolution kernels to the input data, which are usually more helpful than the original input data's initial features, hence improving the model's performance. Convolutional layers are typically followed by pooling layers. A pooling layer is a subsampling technique that removes specific values from convolved features and generates a reduced dimension matrix. Now this matrix is considered as the convolved feature that the convolutional layer produced. The slight changes in the input do not affect the pooled output values, the pooling procedure can help the system be more robust [29].

The LSTM layer receives the values from the previous convolutional layer as input. Multiple LSTM layers are layered together, each layer is acting as input to the next layer. Every LSTM layer has an LSTM unit, which is made up of three basic gates: input, output, and forget. LSTM unit uses this structure to establish a controlled information flow by selecting which information should be ”forgotten” and which should be ”remembered,” hence, allowing it to learn long-term dependencies.

VI. Methodology

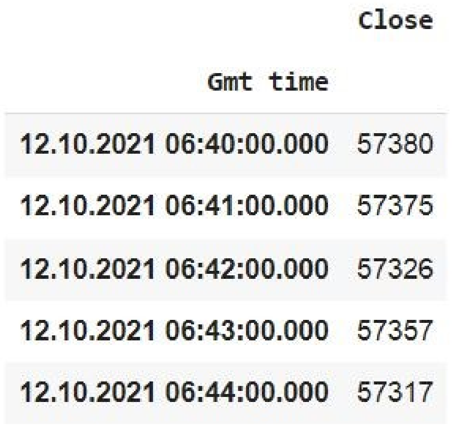
For the crypto Algorithmic Trading Bot, the best effective model must be chosen. So, we compare ARIMA, CNN, Bidirectional LSTM, and CNN-LSTM models, by providing three different datasets with the same number of rows but with different number of variables or columns. Following are three datasets consisting of univariate, multivariate, and PCA variables.

Figs. 6–8. are the sample data imported from the python notebook (Supporting material of the project).



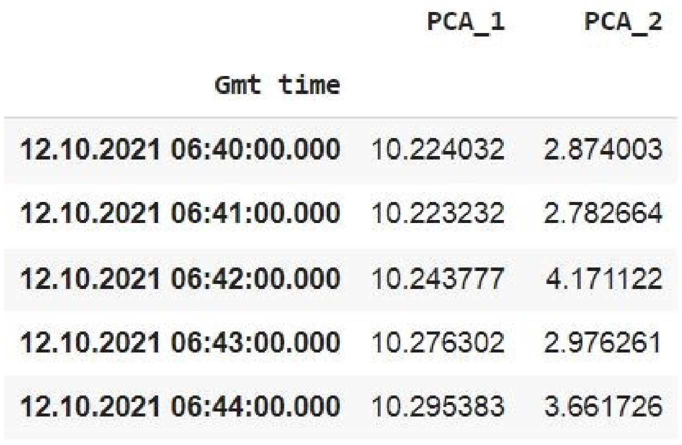
	Open	High	Low	Close	Volume	volume_adi	volume_obv	volume_cmf	volume_fi	volume_mfi	volume_em
Gmt time											
12.10.2021 06:40:00.000	57347	57388	57327	57380	0.0015	0.003282	0.0066	-0.096555	-0.006821	48.041280	5.896667e+13
12.10.2021 06:41:00.000	57380	57386	57346	57375	0.0014	0.003912	0.0052	-0.031247	-0.006847	48.287611	2.428571e+13
12.10.2021 06:42:00.000	57375	57377	57326	57326	0.0016	0.002312	0.0036	-0.073774	-0.017069	40.480384	-4.621875e+13
12.10.2021 06:43:00.000	57327	57365	57322	57357	0.0017	0.003380	0.0053	-0.025142	-0.007102	49.499889	-2.023529e+13
12.10.2021 06:44:00.000	57355	57359	57309	57317	0.0019	0.002088	0.0034	-0.061986	-0.016944	48.549013	-2.500000e+13

Fig. 6. Represents a multivariate data set around 90 columns or features.



	Close
Gmt time	
12.10.2021 06:40:00.000	57380
12.10.2021 06:41:00.000	57375
12.10.2021 06:42:00.000	57326
12.10.2021 06:43:00.000	57357
12.10.2021 06:44:00.000	57317

Fig. 7. Represents the univariate dataset contains only close price feature.



	PCA_1	PCA_2
Gmt time		
12.10.2021 06:40:00.000	10.224032	2.874003
12.10.2021 06:41:00.000	10.223232	2.782664
12.10.2021 06:42:00.000	10.243777	4.171122
12.10.2021 06:43:00.000	10.276302	2.976261
12.10.2021 06:44:00.000	10.295383	3.661726

Fig. 8. Represents the PCA performed variable.

a) Univariate time series: A univariate time series contains only one observation recorded sequentially over an equal period. In our case, the data collected for BTCUSD bitcoin, for example (In Fig. 7), we have only one variable, which is the close price of bitcoin for every one minute.

b) Multivariate time series: There are multiple time-dependent variables in a multivariate time series. The independent variable is dependent on not just its previous values, but also on other variables. This relationship is used to predict future values. For example (In Fig. 6), we have roughly 90 variables in our data for BTCUSD bitcoin, such as open price, close price, low, high, volume, Bollinger band indicator value, RSI value, and so on for every one minute.

c) PCA Performed Variable: When dealing with multidimensional data, there is a problem called the Curse of Dimensionality. High dimensional data is a dataset containing many attributes, usually on the order of a hundred or more. Deep learning models suffer a lot while training such high dimensional data and may lead to overfitting is referred to as Curse of Dimensionality. To deal with this challenge, Principal Component Analysis (PCA) is a dimensionality-reduction approach that is performed for decreasing the dimensionality of large data sets by transforming a massive number of features into a smaller number of variables that retain most of the information in the large dataset. For example, the multidimensional data collected for BTCUSD bitcoin (In Fig. 8). We performed PCA technique and transformed a massive number of variables into two PCA variables.

A. API Dataset Extraction

The BTCUSD historical data are extracted by the binance server (a cryptocurrency exchange platform) spanned from 2021 October to the end of November 2021. The time gap between each data point is one minute. However, the data extracted dates may be less but, we collected these data in a one-minute timeframe. As a result, we have around 50000 data points with variables which is a significant number to train our models. Historical data of BTCUSD bitcoin are extracted with the features like open price, close price, low, high, and volume.

Feature Engineering techniques like slicing the variables are performed based on the needs of univariate and multivariate datasets.

B. Data Pre-processing

There is a large gap in the magnitude of the input data. As a result, the features are normalized by the Z-Score method. To reduce the training time and improve the model efficiency and perform better for the test data as well. To put it mathematically:

$$\frac{x-\mu}{s} \tag{4}$$

Where x represents the input data, μ represents the mean of the input data, and s denotes the standard deviation of the input data. This calculation is done for all the datapoint.

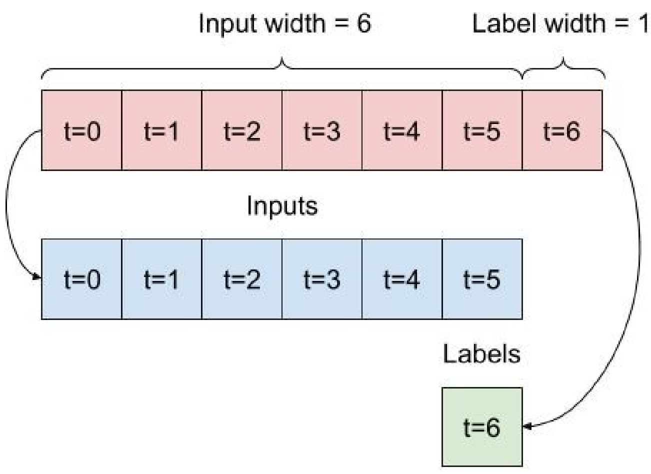


Fig. 9. Data Pre-processing Time steps architecture [31].

For any time-series forecasting architecture, the input data should be given in sequential order to the model. The approach for the Univariate dataset is as follows. In Fig. 9, for example, the first six data points are used as input data in sequential order, whereas the seventh data point is used as output data and so on for the n number of data. This is the same procedure for the Multivariate dataset but, the only difference is the number of input features.

For BTCUSD data, timestep of 30 is used as input data in sequential order, whereas the next data point that is 31, is used as output data and so on for the n number of data. Therefore, all our models use 30 minutes of data as input data to predict the next minute price.

C. Models Implementation

1) ARIMA (Autoregressive integrated moving average) Implementation: In contrast to the other models presented in this paper, the ARIMA model is only applicable to univariate datasets. As a result, the implementation is done only for our univariate dataset (BTCUSD close price dataset). If the data is not stationary, the first step in building an ARIMA model for time series forecasting is to make it stationary. This can be checked by plotting the moving average or moving mean of the data. If there is a trend in the plot, the data is not stationary. If there is no trend in the plot, then the dataset is considered stationary.

Figs. 10–12 are imported from the python notebook (Supporting material of the project)

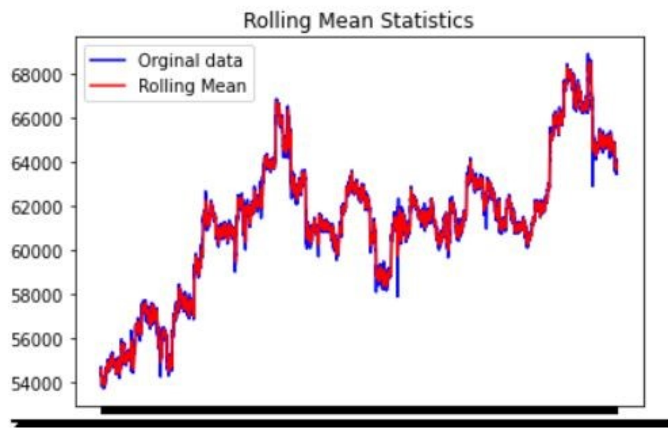
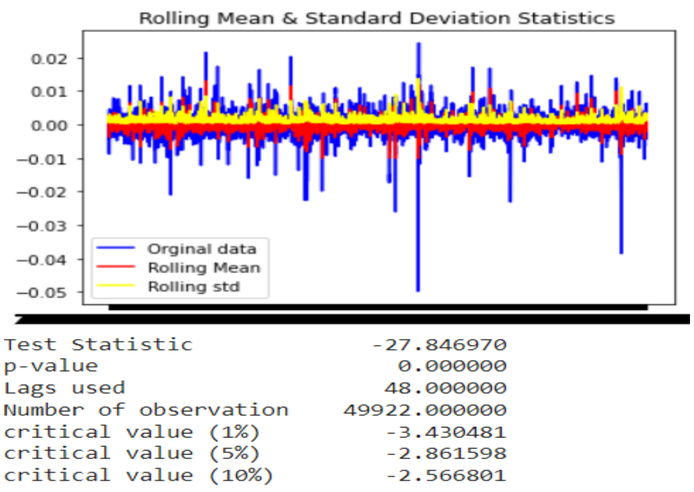


Fig. 10. Dataset Stationary check by rolling mean Statistics.

In Fig. 10. The plot confirms the uptrend in our dataset. Therefore, this dataset is made stationary by applying the Augmented Dickey-Fuller test (Adfuller) that provides a test value, p-value, etc. Again, the rolling mean statistics plot is plotted to check the data stationary now.



```
Test Statistic                 -27.846970
p-value                          0.000000
Lags used                       48.000000
Number of observation        49922.000000
critical value (1%)             -3.430481
critical value (5%)             -2.861598
critical value (10%)            -2.566801
```

Fig. 11. Represents Augmented Dickey-Fuller test (Adfuller).

From Fig. 11. The plot indicates no trend and, the P-value is almost zero that is less than 0.05. Considering this value and thereby rejecting the null hypothesis. Therefore, we conclude that the dataset is stationary. Selecting the best parameter value for (P, d, q) before training our model. The ACF and PACF plots are used to find these parameters.

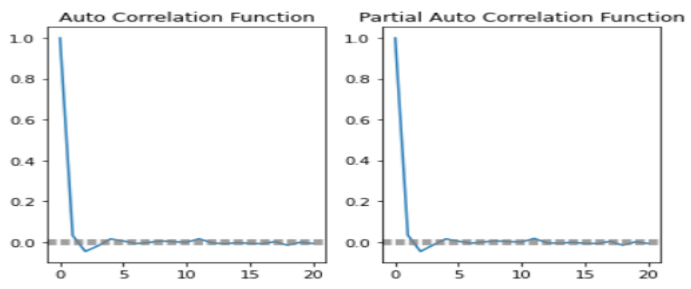


Fig. 12. ACF and PACF plot visualization.

Fig. 12. Confirms ACF and PACF curves cross the dashed line in the x-axis at a value of 1. In our case, we see the first order differencing makes the BTCUSD dataset stationary. Hence, d = 1. As a result, the best parameter of (p, d, q) is (1, 1, 1). Our model is trained based on this parameter for the BTCUSD dataset. Prediction results are discussed further in the evaluation section.

2) CNN Implementation: Fig. 13 illustrates the CNN model architecture used in this experiment. The following is how the CNN model layers configuration is built; The input training set data is a three-dimensional data vector (None, 30, 1), where 30 represents the time step size and 1 represents the number of features in the input dimension. None parameter in input layer basically takes the number of row data in the BTCUSD dataset. In our case, we have 50000 data points.

Figs. 13–15 are the models architecture imported from the python notebook (Supporting material of the project)

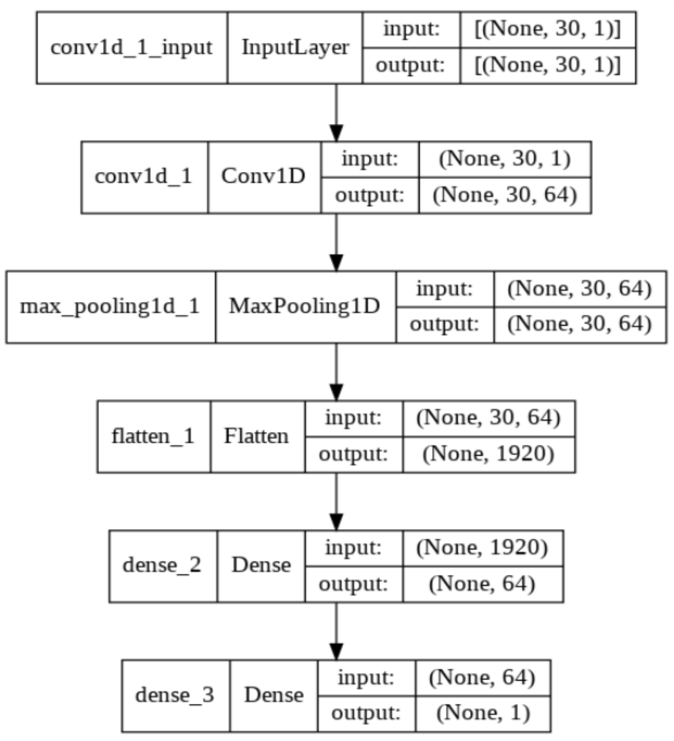

conv1d_1_input | InputLayer | input: [(None, 30, 1)] | output: [(None, 30, 1)]
conv1d_1 | Conv1D | input: (None, 30, 1) | output: (None, 30, 64)
max_pooling1d_1 | MaxPooling1D | input: (None, 30, 64) | output: (None, 30, 64)
flatten_1 | Flatten | input: (None, 30, 64) | output: (None, 1920)
dense_2 | Dense | input: (None, 1920) | output: (None, 64)
dense_3 | Dense | input: (None, 64) | output: (None, 1)

Fig. 13. CNN model architecture used in the python notebook.

The input data is first fed into a one-dimensional convolution layer for training, which extracts additional features and creates a three-dimensional output vector (None, 30, 64), where 64 indicates the filter size of the convolution layer. The vector then passes through the pooling layer, generating a three-dimensional output vector (None, 30, 64). The vector is then sent through the flatten layer, producing a two-dimensional output vector (None, 1920). The vector then passes through the dense layer, returning a two-dimensional output vector (None, 64). The output vector subsequently passes through another dense layer, resulting in the final output data (None, 1).

Various hyperparameters and parameters are tuned. For the CNN model, the best-performing parameters and hyperparameters are chosen. Table I. displays the final selected parameters.

TABLE I
CNN PARAMETER SETTINGS

Parameter and Hyperparameter	**Values**
Batch Size	64
Time step	30
Learning Rate	0.0001
Convolution layer filters	64
Convolution layer kernel size	1
Convolution layer activation function	RELU
Pooling layer size	1
First dense layer	64
Second dense layer	1
First dense layer activation function	RELU
Second dense layer activation function	Linear
Epochs	60
Optimizer	Adam
Loss Function	Mean Squared Error (MSE)

The above discussed CNN model architecture, parameter, and hyperparameter are implemented for univariate datasets (BTCUSD). The same probably applies to multivariate and PCA datasets. Except for the number of features in the input layer, because of the different number of variables present in multivariate and PCA datasets. Therefore, the CNN model is implemented for all the datasets (univariate, multivariate, and PCA). And their results are further discussed in the evaluation section.

3) Bidirectional LSTM Implementation: The Bidirectional LSTM model architecture implemented in this experiment is shown in Fig. 14. The Bidirectional LSTM model layers configuration is constructed as follows: the input training dataset is a three-dimensional data vector (None, 30, 1), where 30 denotes the time step size and 1 is the number of features in the input dimension. The number of rows in the BTCUSD dataset is mainly taken into account by the None parameter in the input layer. In our case, we have 50000 data points.

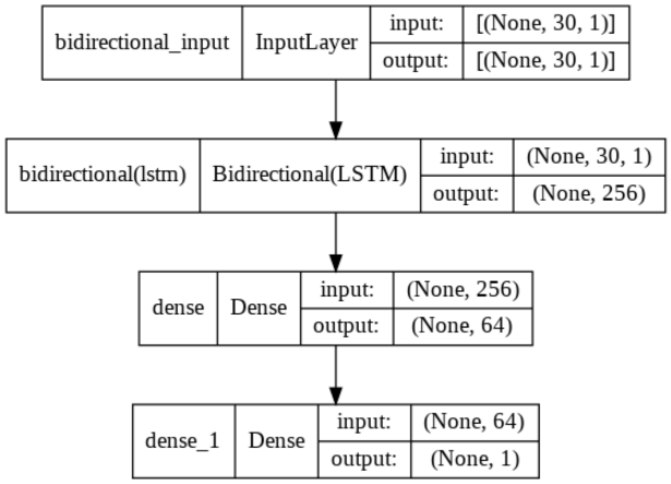


Fig. 14. Bidirectional LSTM model architecture used in the python notebook.

The input data is first fed into a Bidirectional LSTM layer for training, which returns a two-dimensional output vector (None, 256). After passing through the first dense layer, the vector produces a two-dimensional output vector (None, 64). Following that, the output vector passes through another dense layer, resulting in final output data (None, 1).

Several hyperparameters and parameters are tuned. The best-performing parameters and hyperparameters for the Bidirectional LSTM model are chosen. The final selected parameters are shown in Table II.

TABLE II
BIDIRECTIONAL LSTM PARAMETER SETTINGS

Parameter and Hyperparameter	**Values**
Batch Size	64
Time step	30
Learning Rate	0.0001
Bidirectional LSTM	256
First dense layer	64
Second dense layer	1
Second dense layer activation function	Linear
Epochs	60
Optimizer	Adam
Loss Function	Mean Squared Error (MSE)

The above discussed Bidirectional LSTM model architecture, parameter, and hyperparameter are implemented for univariate datasets (BTCUSD). The same probably applies to multivariate and PCA datasets. Except for the number of features in the input layer. Because of the different number of variables present in multivariate and PCA datasets. Therefore the Bidirectional LSTM model is implemented for all the datasets (univariate, multivariate, and PCA). And their results are further discussed in the evaluation section.

4) CNN-LSTM Implementation: Fig. 15 illustrates the CNN-LSTM model architecture used in this experiment. The input data for the training set is a three-dimensional data vector (None, 30, 1), where 30 represents the time step size and 1 denotes the number of features in the input dimension. The None parameter in the input layer is mainly responsible for the number of rows in the BTCUSD dataset.

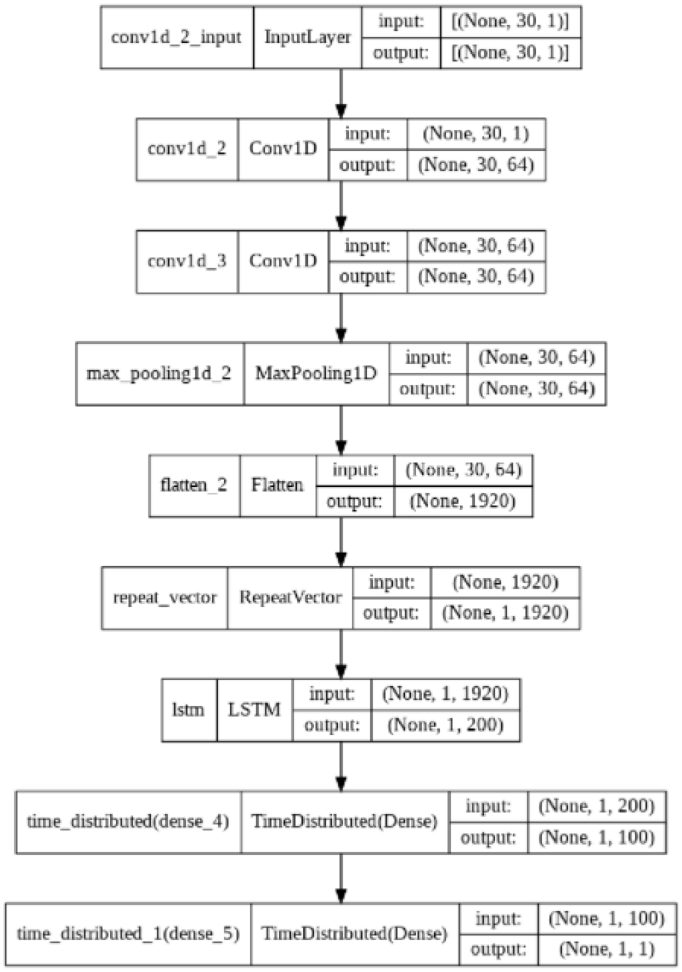


Fig. 15. CNN-LSTM model architecture used in the python notebook.

The input data is first fed into a one-dimensional convolution layer for training, which extracts additional features and generates a three-dimensional output vector (None, 30, 64), where 64 is the convolution layer's filter size. After passing through the pooling layer, the vector generates the same output vector (None, 30, 64). After that, the vector is transmitted via the flatten layer, resulting in a two-dimensional output vector (None, 1920). After that, the vector is passed via a repeat vector layer, which is simply another layer with the same output vector. The vector is then sent through the LSTM layer, producing a two-dimensional output vector (None, 200). Finally, the vector is passed to two dense layers resulting in the last output vector (None, 1).

Numerous hyperparameters and parameters are tuned. For the CNN-LSTM model, the best-performing parameters and hyperparameters are chosen. Table III. displays the final selected parameters.

The above discussed CNN-LSTM model architecture, parameter, and hyperparameter are implemented for univariate datasets (BTCUSD). The same probably applies to multivariate

TABLE III
CNN-LSTM Parameter settings

Parameter and Hyperparameter	Values
Batch Size	64
Time step	30
Learning Rate	0.0001
Convolution layer filters	64
Convolution layer kernel size	1
Convolution layer activation function	RELU
Pooling layer size	1
LSTM layer	200
LSTM layer activation function	RELU
First dense layer	100
First dense layer activation function	RELU
Second dense layer	1
Second dense layer activation function	Linear
Epochs	60
Optimizer	Adam
Loss Function	Mean Squared Error (MSE)

and PCA datasets. Except for the number of features in the input layer. Because of the different number of variables present in multivariate and PCA datasets. Therefore the CNN-LSTM model is implemented for all the datasets (univariate, multivariate, and PCA). And their results are further discussed in the evaluation section.

D. Model Training

One of the high-frequency cryptocurrency (BTCUSD) datasets were used to train and optimize all the models. With the goal of maximizing the price prediction performance on the test dataset for the most volatile price patterns. To minimize the loss function, the Mean Square Error (MSE), all models were trained for an optimum of 60 epochs with a 10% validation ratio. The batch size of 64 is used to reduce the training time and makes the training more efficient. Using an Adam optimizer, the learning rate is set at 0.0001 with no early stopping.

E. Model Evaluation

Due to the fact that the ARIMA model does not work with multivariate datasets. Table IV. confirms Even the ARIMA model failed miserably in the univariate dataset. In this paper, the model evaluation and the accuracy comparisons are discussed only for the other models such as CNN, Bidirectional LSTM, and CNN-LSTM.

TABLE IV
ARIMA Model Accuracy Metrics for univariate dataset

ARIMA Model	Value
RMSE	639.030
MAE	504.838
R^2	-3.822

Once the CNN, Bidirectional LSTM, and CNN-LSTM models are trained with the pre-processed training data. These models are used to test the data in the validation dataset. Where the actual value or price of the BTCUSD bitcoin are compared with the predicted value by the models.

1) Models Graphical Comparison: Totally, we have 5000 data on the testing dataset. Plotting all the test data will be hard to visualize. So, the first 500 data points are considered for plotting the actual and predicted price of the BTCUSD bitcoin. Whereas the performance metrics take all the test data points (5000) for the evaluation.

Figs. 16–24 are the Models Graphical representation imported from the python notebook (Supporting material of the project)

Univariate dataset Graphical Representation

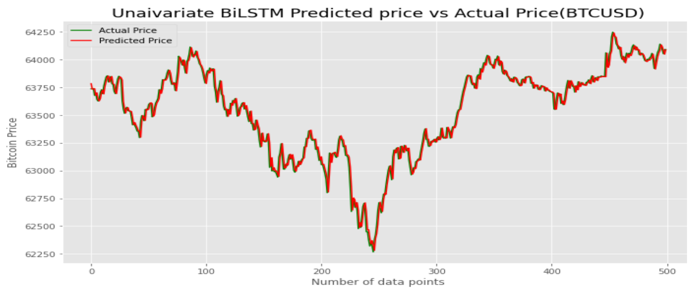


Fig. 16. Univariate dataset plot for the predicted and actual value of BiLSTM.

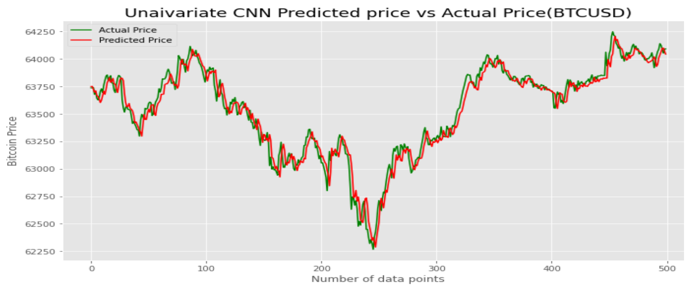


Fig. 17. Univariate dataset plot for the predicted and actual value of CNN.

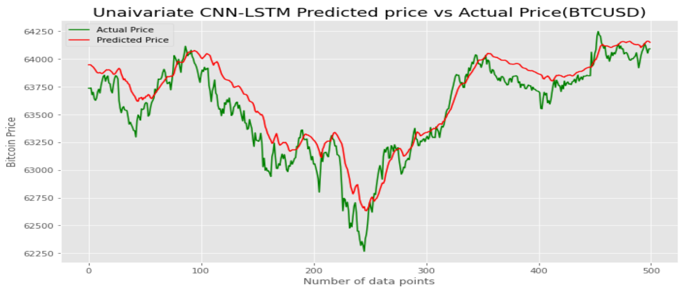


Fig. 18. Univariate dataset plot for the predicted and actual value of CNN-LSTM.

Multivariate dataset Graphical Representation

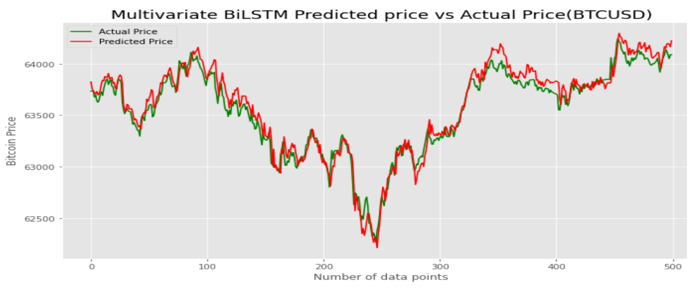


Fig. 19. Multivariate dataset plot for the predicted and actual value of BiLSTM.

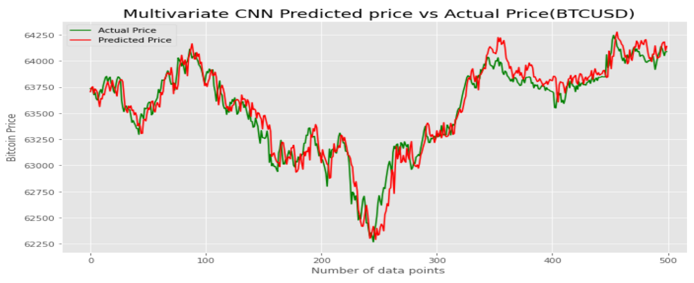


Fig. 20. Multivariate dataset plot for the predicted and actual value of CNN.

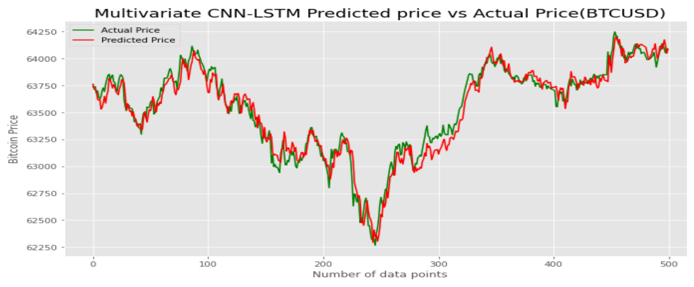


Fig. 21. Multivariate dataset plot for the predicted and actual value of CNN-LSTM.

PCA performed dataset Graphical Representation

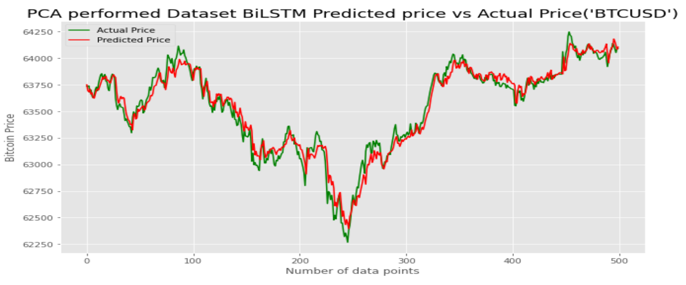


Fig. 22. PCA dataset plot for the predicted and actual value of BiLSTM.

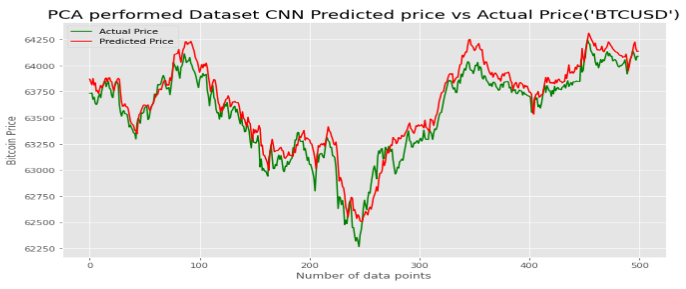


Fig. 23. PCA dataset plot for the predicted and actual value of CNN.

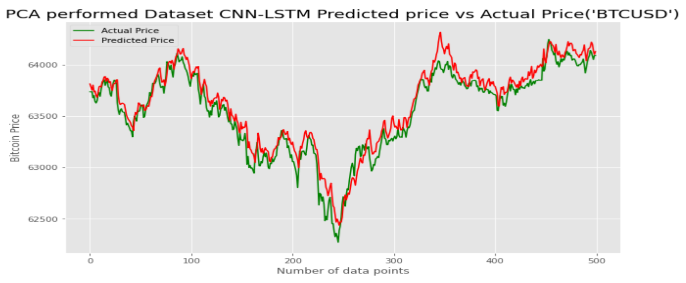


Fig. 24. PCA dataset plot for the predicted and actual value of CNN-LSTM.

Figs. 16–24 show the plotting results of combining univariate, multivariate, and PCA datasets with three forecasting methods such as CNN, BiLSTM, and CNN-LSTM. When compared to other models, Univariate BiLSTM has the highest degree of broken line fitting that closely coincides with each other, while Univariate CNN-LSTM has the lowest degree of broken line fitting.

2) Models Performance Metrics Evaluation: The Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R-square score (R^2) are used as evaluation criteria for all models in order to evaluate their predicting effectiveness. The formula for calculating RMSE is as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\widehat{y}_i - y_i\right)^2} \tag{5}$$

where y_i denotes the actual value and $\hat{y}_i$ denotes the predicted value. The better the forecasting model, the lower the RMSE value. The formula for calculating MAE is as follows:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}_i - ty_i\right| \tag{6}$$

where y_i denotes the actual value and $\hat{y}_i$ denotes the predicted value. The better the forecasting model, the lower the MAE value. The formula for calculating R-square score (R^2) is as follows:

$$R^2 = 1 - \frac{\left(\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2\right)/n}{\left(\sum_{i=1}^{n}\left(\bar{y}_i - t\hat{y}_i\right)^2\right)/n} \tag{7}$$

where y_i denotes the actual value and $\hat{y}_i$ denotes the predicted value. The better the forecasting model, the higher the R-square score (R^2) value. The R^2 value is that varies from 0 to 1. The closer R^2 value to 1, the better the model fits the data.

VII. VII Results and Discussion

The performance metrics score of each method can be computed based on the predicted and actual values of each method, and the comparison results of the nine methods are displayed in Table V and Figs. 25–27.

TABLE V
Comparison of nine method performance evaluation indexes.

Models	RMSE	MAE	R^2
Univariate CNN	110.399	81.799	0.977
Univariate CNN-LSTM	129.409	104.725	0.969
Multivariate BiLSTM	186.194	144.677	0.936
Multivariate CNN	90.904	71.197	0.984
Multivariate CNN-LSTM	73.722	57.961	0.990
PCA performed BiLSTM	70.786	56.422	0.990
PCA performed CNN	133.934	109.422	0.967
PCA performed CNN-LSTM	111.027	90.894	0.977
Univariate BiLSTM	**42.302**	**31.109**	**0.996**

Figs. 25–27. are the Models Bar chart representation of performance Metrics.

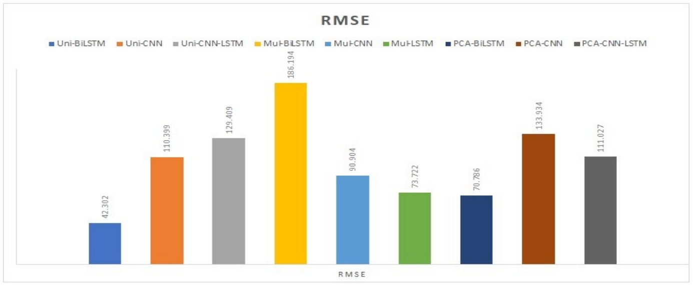


Fig. 25. The results of RMSE and Comparison between all models.

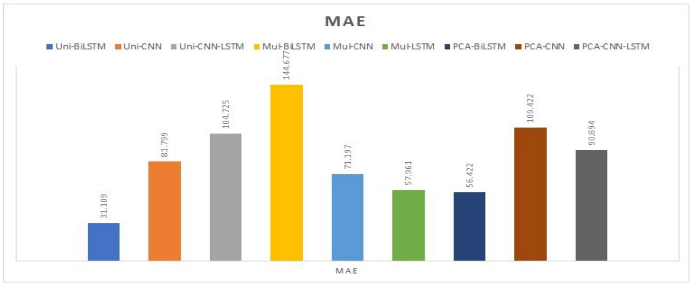


Fig. 26. The results of MAE and Comparison between all models.

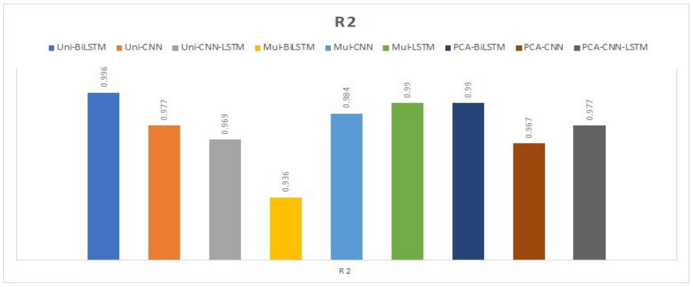


Fig. 27. The results of R^2 score and Comparison between all models.

The RMSE and MAE values of the Multivariate BiLSTM model are high, and the R^2 score is low, as shown in Table V and Figs. 25–27, whereas the RMSE and MAE values of the Univariate BiLSTM model is minimal, and the R2 score is high, very close to 1. Meanwhile, the PCA BiLSTM model performed quite good than the Multivariate BiLSTM model. Henceforth it is very evident that the BiLSTM works so well for the small dimensional variables like univariate and PCA datasets variables. By the above results, we state that the BiLSTM model suffers from a problem of dimensionality curse.

The RMSE and MAE values of the PCA performed CNN model are high, and the R^2 score is low, as shown in Table V and Figs. 25–27, whereas the RMSE and MAE values of the Multivariate CNN model is minimal, and the R2 score is high, close to 1. Meanwhile, the Univariate CNN model performed quite good than the PCA performed CNN model. Henceforth it is evident that the CNN works well for the high dimensional variables like Multivariate datasets variables than the low dimensional variables like univariate and PCA variables. The CNN-LSTM model also performs in the manner with univariate, multivariate, and PCA datasets.

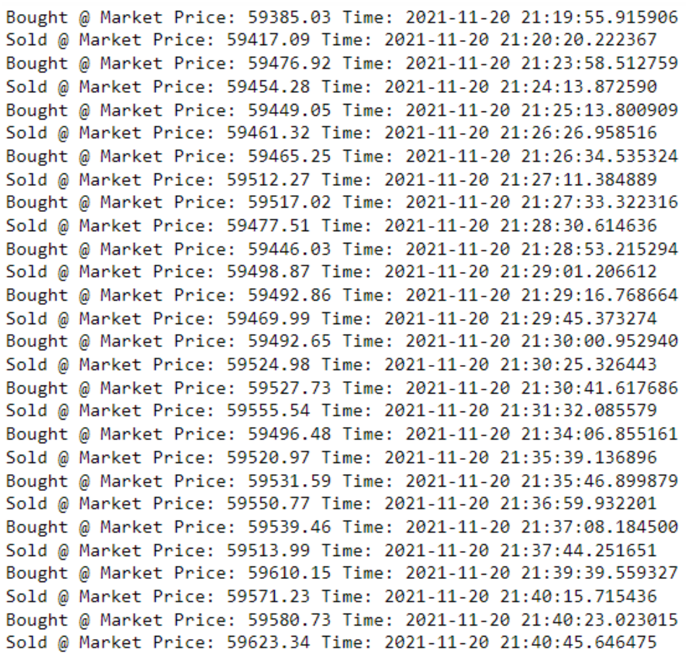
The results show that the performance metrics of the univariate BiLSTM model are the best among the nine methods. In terms of forecasting accuracy, RMSE is 42.302 and MAE is 31.109 is the smallest among the nine forecasting models and has high forecasting accuracy, in terms of forecasting performance and the R^2 of the univariate BiLSTM model is 0.996, which is improved by 0.019%, 0.027%, 0.06%, 0.012%, 0.006%, 0.006%, 0.029%, and 0.019%, respectively, compared with the other nine methods. Therefore, the univariate BiLSTM model proposed in this paper is superior to the other nine comparative models in terms of plotting line fitting degree and error value. It can predict the closing price of the next minute so well for the BTCUSD bitcoin.

VIII. Cryptocurrency Algorithmic Trading Bot Implementation via Binance Platform

An algorithmic trading bot was developed by feeding the univariate BiLSTM model to the bot that automatically buys and sells the BTCUSD bitcoin. It can be accomplished with the help of binance future testnet account, a cryptocurrency exchange demo platform. Binance tesnet server also offers an Application Programming Interface (API), which enables traders to connect to Binance servers using a variety of programming languages. Python is the programming language used in this experiment to interact with the Binance server. Traders can purchase and sell digital assets on Binance tesnet account and automate their trades depending on their strategy.

Once connected to the Binance testnet server, we can automate trades using our model prediction, which forecasts the result every minute based on the last 30 minutes of BTCUSD Close price data. Using a handy python library, the last 30 minutes of Close price data (BTCUSD) is retrieved from the Binance server and provided as input to our model to predict the 31st-minute price. These data are continuously extracted from the binance server and fed to our model. Therefore, our model forecasts the predicted close price data (BTCUSD) for every one-minute time.

Every minute, we obtain the price prediction by our model. Now, simple logic is applied to buy and sell the BTCUSD bitcoin. If our model prediction is high than the current price of the BTCUSD bitcoin, we trigger or place a buy order at the current price. The current price of the BTCUSD bitcoin is accessed via binance testnet server. The sell orders are placed when the BTCUSD bitcoin bought price is greater or lesser than the current bitcoin price by 20 USD. The threshold value of 20 is chosen since the BTCUSD is highly volatile we can easily attain a profit of 20 USD and vice versa. While building an Algorithmic trading bot on the lower timeframe we usually focus on the quantity of the trade than the quality of the trade [32]. The profit or loss depends on the lot of size of the order that has been placed or sold.



```
Bought @ Market Price: 59385.03 Time: 2021-11-20 21:19:55.915906
Sold @ Market Price: 59417.09 Time: 2021-11-20 21:20:20.222367
Bought @ Market Price: 59476.92 Time: 2021-11-20 21:23:58.512759
Sold @ Market Price: 59454.28 Time: 2021-11-20 21:24:13.872590
Bought @ Market Price: 59449.05 Time: 2021-11-20 21:25:13.800909
Sold @ Market Price: 59461.32 Time: 2021-11-20 21:26:26.958516
Bought @ Market Price: 59465.25 Time: 2021-11-20 21:26:34.535324
Sold @ Market Price: 59512.27 Time: 2021-11-20 21:27:11.384889
Bought @ Market Price: 59517.02 Time: 2021-11-20 21:27:33.322316
Sold @ Market Price: 59477.51 Time: 2021-11-20 21:28:30.614636
Bought @ Market Price: 59446.03 Time: 2021-11-20 21:28:53.215294
Sold @ Market Price: 59498.87 Time: 2021-11-20 21:29:01.206612
Bought @ Market Price: 59492.86 Time: 2021-11-20 21:29:16.768664
Sold @ Market Price: 59469.99 Time: 2021-11-20 21:29:45.373274
Bought @ Market Price: 59492.65 Time: 2021-11-20 21:30:00.952940
Sold @ Market Price: 59524.98 Time: 2021-11-20 21:30:25.326443
Bought @ Market Price: 59527.73 Time: 2021-11-20 21:30:41.617686
Sold @ Market Price: 59555.54 Time: 2021-11-20 21:31:32.085579
Bought @ Market Price: 59496.48 Time: 2021-11-20 21:34:06.855161
Sold @ Market Price: 59520.97 Time: 2021-11-20 21:35:39.136896
Bought @ Market Price: 59531.59 Time: 2021-11-20 21:35:46.899879
Sold @ Market Price: 59550.77 Time: 2021-11-20 21:36:59.932201
Bought @ Market Price: 59539.46 Time: 2021-11-20 21:37:08.184500
Sold @ Market Price: 59513.99 Time: 2021-11-20 21:37:44.251651
Bought @ Market Price: 59610.15 Time: 2021-11-20 21:39:39.559327
Sold @ Market Price: 59571.23 Time: 2021-11-20 21:40:15.715436
Bought @ Market Price: 59580.73 Time: 2021-11-20 21:40:23.023015
Sold @ Market Price: 59623.34 Time: 2021-11-20 21:40:45.646475
```

Fig. 28. Displays the trades performed by our bot. Imported from the python notebook (Supporting material of the project).

From Fig. 28 our bot performed 10 winning trades and performed 4 losing trades on average. Therefore, we can conclude that the created algorithmic trading bot for BTCUSD bitcoin has a 60% winning trades on the lower timeframe.

IX. Conclusion

The BTCUSD data of the cryptocurrency is used in this study to verify the experimental results. In comparison to all eight models, the experimental findings reveal that the univariate BiLSTM has the highest forecasting accuracy and best performance. Univariate BiLSTM has the minimal MAE and RMSE values among all other models, and R^2 is very close to 1.

The profitability of autonomous trading in BTCUSD bitcoin was also examined and researched in this study. The various results indicate that autonomous trading by deep learning models has the potential to be profitable. Adapting models to a real market is, however, still a challenge. Choosing the right parameters based on market behaviour is important, but it is a complex process. Further research is needed to accomplish this task.

X. Future Work

However, there are certain flaws in the model. It simply evaluates the impact of BTCUSD numerical data on closing prices, for example, and ignores emotive elements like news and national policy in its prediction. Our future research will primarily focus on improving sentiment analysis of bitcoin-related news and national policies in order to ensure bitcoin forecast accuracy. Sentiment analysis can be performed by NLP (Natural Language Processing). The combination of our model forecast, and NLP sentiment analysis might give us excellent results.

Reinforcement learning can be used to provide risk and reward for the trades that are taken by our bot. Therefore, our bot can learn from its mistakes depending on the risk and reward that has been generated by reinforcement learning. There are research going on in field of reinforcement learning for stock prediction.

Acknowledgment

I am indebted to Professor Jiadong Yu for her support and creative input throughout the project. I also wish to thank the binance future testnet platform for the high-quality dataset that made exploring, understanding, and implementing this project as success.

Appendix

All Figure represented in this section are imported from python notebook (Supporting material of the project)

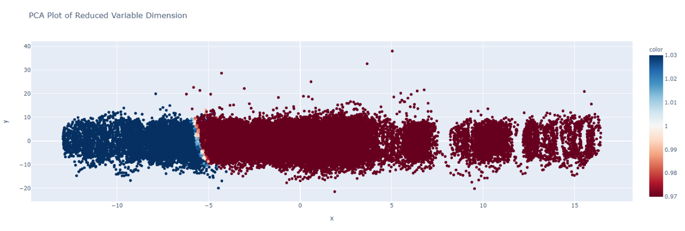


Fig. A. Represents Principal Component Analysis for the multivariate dataset and the variables of multivariate dataset are transformed into two-dimensional variable.

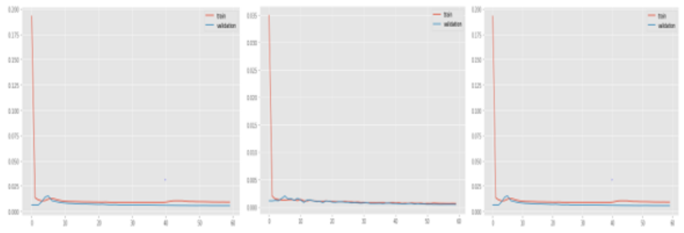


Fig. B. Represents the training and validation loss of BiLSTM, CNN, and CNN-LSTM for univariate dataset.

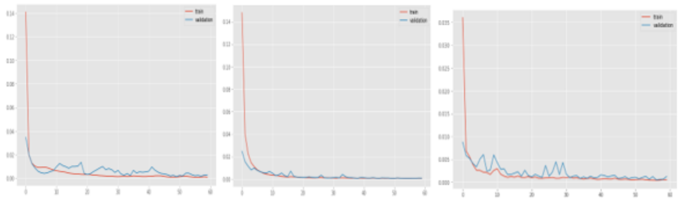


Fig. C. Represents the training and validation loss of BiLSTM, CNN, and CNN-LSTM for multivariate dataset.

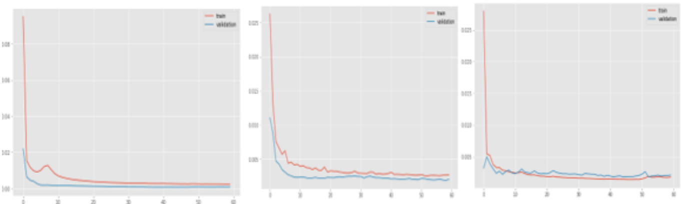


Fig. D. Represents the training and validation loss of BiLSTM, CNN, and CNN-LSTM for PCA performed dataset.

References

[1] R. Donovan, “Store atmosphere and purchasing behavior,” *Journal of Retailing*, vol. 70, no. 3, pp. 283–294, 1994. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/002243599490037X

[2] H.-C. Shu and J.-H. Chang, “Investor Sentiment and Financial Market Volatility,” *Journal of Behavioral Finance*, vol. 16, no. 3, pp. 206–219, Jul. 2015. [Online]. Available: http://www.tandfonline.com/doi/full/10.1080/15427560.2015.1064930

[3] T. Odean, “Volume, Volatility, Price, and Profit When All Traders Are Above Average,” *The Journal of Finance*, vol. 53, no. 6, pp. 1887–1934, Dec. 1998. [Online]. Available: http://doi.wiley.com/10.1111/0022-1082.00078

[4] J. Chen, “Founded in the 1600s, the Amsterdam Stock Exchange is the world’s oldest.” [Online]. Available: https://www.investopedia.com/terms/a/aex.asp

[5] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang, “A CNN-LSTM-Based Model to Forecast Stock Prices,” *Complexity*, vol. 2020, pp. 1–10, Nov. 2020. [Online]. Available: https://www.hindawi.com/journals/complexity/2020/6622927/

[6] P. K. Jain, “Financial Market Design and the Equity Premium: Electronic versus Floor Trading,” *The Journal of Finance*, vol. 60, no. 6, pp. 2955–2985, Dec. 2005. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.2005.00822.x

[7] T. Hendershott and R. Riordan, “Algorithmic Trading and Information,” *SSRN Electronic Journal*, 2011. [Online]. Available: http://www.ssrn.com/abstract=1472050

[8] R. K. Narang, *Inside the black box: the simple truth about quantitative trading*, ser. Wiley finance. Hoboken, N.J: Wiley, 2009, oCLC: ocn317777911.

[9] G. Nuti, M. Mirghaemi, P. Treleaven, and C. Yingsaeree, “Algorithmic Trading,” *Computer*, vol. 44, no. 11, pp. 61–69, Nov. 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5696713/

[10] D. Shah and K. Zhang, “Bayesian regression and Bitcoin,” *arXiv:1410.1231 [cs, math, stat]*, Oct. 2014, arXiv: 1410.1231. [Online]. Available: http://arxiv.org/abs/1410.1231

[11] I. Madan, “Automated Bitcoin Trading via Machine Learning Algorithms,” *Department of Computer Science, Stanford University*, 2014. [Online]. Available: https://www.semanticscholar.org/paper/Automated-Bitcoin-Trading-via-Machine-Learning-Madan/e0653631b4a476abf5276a264f6bbff40b132061

[12] S. McNally, J. Roche, and S. Caton, “Predicting the Price of Bitcoin Using Machine Learning,” in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. Cambridge: IEEE, Mar. 2018, pp. 339–343. [Online]. Available: https://ieeexplore.ieee.org/document/8374483/

[13] G. E. P. Box and G. M. Jenkins, *Time series analysis: forecasting and control*, rev. ed ed., ser. Holden-Day series in time series analysis and digital processing. San Francisco: Holden-Day, 1976.

[14] M. Saad, J. Choi, D. Nyang, J. Kim, and A. Mohaisen, “Toward Characterizing Blockchain-Based Cryptocurrencies for Highly Accurate Predictions,” *IEEE Systems Journal*, vol. 14, no. 1, pp. 321–332, Mar. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8840919/

[15] H. Jang and J. Lee, “An Empirical Study on Modeling and Prediction of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information,” *IEEE Access*, vol. 6, pp. 5427–5437, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8125674/

[16] J. Huisu, “Predicting Bitcoin Prices by Using Rolling Window LSTM model,” 2018. [Online]. Available: https://www.semanticscholar.org/paper/Predicting-Bitcoin-Prices-by-Using-Rolling-Window-Huisu/c7760a4a97f2a4ec7f2e04397479791d11035bb6

[17] R. Nochai and T. Nochai, “ARIMA MODEL FOR FORECASTING OIL PALM,” 2006. [Online]. Available: https://www.semanticscholar.org/paper/ARIMA-MODEL-FOR-FORECASTING-OIL-PALM-Nochai-Nochai/92cf62773a0247782f1d8cdb077adc8b587a1109

[18] P.-F. Pai and C.-S. Lin, “A hybrid ARIMA and support vector machines model in stock price forecasting,” *Omega*, vol. 33, no. 6, pp. 497–505, Dec. 2005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0305048304001082

[19] B. G. Tabachnick, *SAS for Windows workbook: for Tabachnick and Fidell ’Using multivariate statistics’, 4th ed*, 10th ed. Boston: Allyn and Bacon, 2003.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386

[21] L. Qin, N. Yu, and D. Zhao, “Applying the Convolutional Neural Network Deep Learning Technology to Behavioural Recognition in Intelligent Video,” *Tehnicki vjesnik - Technical Gazette*, vol. 25, no. 2, Apr. 2018. [Online]. Available: https://hrcak.srce.hr/199152

[22] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: https://direct.mit.edu/neco/article/9/8/1735-1780/6109

[23] O. Christopher, “Understanding LSTM Networks – colah’s blog.” [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[24] Q. Zhuge, L. Xu, and G. Zhang, “LSTM neural network with emotional analysis for prediction of stock price,” vol. 25, no. 2, pp. 64–72, 2017.

[25] J. Kim and N. Moon, “BiLSTM model based on multivariate time series data in multiple field for forecasting trading area,” *Journal of Ambient Intelligence and Humanized Computing*, Jul. 2019. [Online]. Available: http://link.springer.com/10.1007/s12652-019-01398-9

[26] Y. Song and I. Kim, “DeepAct: A Deep Neural Network Model for Activity Detection in Untrimmed Videos,” *Journal of Information Processing Systems*, vol. 14, no. 1, pp. 150–161, Feb. 2018. [Online]. Available: https://doi.org/10.3745/JIPS.04.0059

[27] J. Eapen, D. Bein, and A. Verma, “Novel Deep Learning Model with CNN and Bi-Directional LSTM for Improved Stock Market Index Prediction,” in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. Las Vegas, NV, USA: IEEE, Jan. 2019, pp. 0264–0270. [Online]. Available: https://ieeexplore.ieee.org/document/8666592/

[28] W. Rawat and Z. Wang, “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review,” *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, Sep. 2017. [Online]. Available: https://direct.mit.edu/neco/article/29/9/2352-2449/8292

[29] I. E. Livieris, E. Pintelas, and P. Pintelas, “A CNN–LSTM model for gold price time-series forecasting,” *Neural Computing and Applications*, vol. 32, no. 23, pp. 17 351–17 360, Dec. 2020. [Online]. Available: https://link.springer.com/10.1007/s00521-020-04867-x

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, May 2017, pp. 1097–1105.

[31] “Time series forecasting | TensorFlow Core.” [Online]. Available: https://www.tensorflow.org/tutorials/structured_data/time_series

[32] “Ultimate Guide to the Best Bitcoin Trading Bots 2021 - Do they Work?” Nov. 2021. [Online]. Available: https://blockonomi.com/bitcoin-trading-bots/