Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 7097044, 15 pages
https://doi.org/10.1155/2022/7097044

Research Article

Impact of Healthcare on Stock Market Volatility and Its Predictive Solution Using Improved Neural Network

Nusrat Rouf,[1] **Majid Bashir Malik**,[1] **Sparsh Sharma**,[2] **In-Ho Ra**,[3] **Saurabh Singh**,[4] **and Abhishek Meena**[5]

[1]*Department of Computer Sciences, Baba Ghulam Shah Badshah University, Rajouri 185234, India*
[2]*Department of Computer Science & Engineering, National Institute of Technology, Srinagar, Jammu and Kashmir, India*
[3]*School of Computer Information and Communication Engineering, Kunsan National University, Gunsan, Republic of Korea*
[4]*Department of Industrial & Systems Engineering, Dongguk University, Seoul 04620, Republic of Korea*
[5]*Division of Physics and Semiconductor Science, Dongguk University, Seoul 04620, Republic of Korea*

Correspondence should be addressed to Sparsh Sharma; sparsh.sharma@nitsri.net and In-Ho Ra; ihra@kunsan.ac.kr

Received 12 April 2022; Accepted 15 July 2022; Published 11 August 2022

Academic Editor: Yousaf Bin Zikria

The unprecedented Corona Virus Disease (COVID-19) pandemic has put the world in peril and shifted global landscape in unanticipated ways. The SARSCoV2 virus, which caused the COVID-19 outbreak, first appeared in Wuhan, Hubei Province, China, in December 2019 and quickly spread around the world. This pandemic is not only a global health crisis, but it has caused the major global economic depression. As soon as the virus spread, stock market prices plummeted and volatility increased. Predicting the market during this outbreak has been of substantial importance and is the primary motivation to carry out this work. Given the nonlinearity and dynamic nature of stock data, the prediction of stock market is a challenging task. The machine learning models have proven to be a good choice for the development of effective and efficient prediction systems. In recent years, the application of hyperparameter optimization techniques for the development of highly accurate models has increased significantly. In this study, a customized neural network model is proposed and the power of hyperparameter optimization in modelling stock index prices is explored. A novel dataset is generated using nine standard technical indicators and COVID-19 data. In addition, the primary focus is on the importance of selection of optimal features and their preprocessing. The utilization of multiple feature ranking techniques combined with extensive hyperparameter optimization procedures is comprehensive for the prediction of stock index prices. Moreover, the model is evaluated by comparing it with other models, and results indicate that the proposed model outperforms other models. Given the detailed design methodology, preprocessing, exploratory feature analysis, and hyperparameter optimization procedures, this work gives a significant contribution to stock analysis research community during this pandemic.

1. Introduction

The stock market plays an important role in country's economy, as it is a platform for most of the money exchange throughout the world. Investing in a stock market through a disciplined strategy can lead to substantial gains. The stock market prediction is a challenging task due to the nonlinear, dynamic, and chaotic data [1]. A wise investment is only possible if the stock data is analyzed properly before investment. The analysis of the voluminous volatile and dynamic financial data is challenging. With the advent of online trading, people are moving towards the automated intelligent decision support systems rather than using classical fundamental analysis approaches for stock price prediction [2, 3].

The global economy and stock markets have been greatly affected by COVID-19. Significant market indexes, such as the Standard and Poor's (S&P 500), plummeted by 41% on February 19, 2020 [4]. Economic operations of numerous countries suddenly halted as tight quarantine rules were

implemented to combat the devastations of this outbreak [5]. Transportation between countries has been limited, leading to the hampering of global economic activity [6]. Global markets have labelled this disaster a black swan event. This pandemic has significantly disturbed and influenced stock market activity around the world [7]. In year 2020, due to the effects of pandemic, it was estimated that the China's Gross Domestic Product (GDP) may fall by 6.2 percent, while the United States (US) GDP may plunge down by 8.4 percent [2, 8]. The rest of the world's currency could lose 5.9% of its value. Due to the increased volatility, foreign investors have been withdrawing money from the markets that has weakened the economy [9, 10]. Panic selling has induced confusion among investors. Therefore, predicting the markets in such conditions is challenging. The stock index is a representative of whole stock market or a sector or a part of market. It is a temporal characteristic with low signal-to-noise ratio and heavy-tailed distribution [11]; thus, prediction is a challenging task. Prediction of index prices aims to build a model where the independent variable is predicted using numerous dependent variables. Artificial Neural Network (ANN) models are the most popular method that adapt to the dynamic nature of markets easily [12]. These are the models that understand the context of a problem by creating multiple linear transformations on the feature space followed by a nonlinearity to create its simplified representation [7, 13]. Neural network models can represent any distribution over the input feature space; therefore, they are known as universal function approximator [14]. Neural networks prove to be a better option when there is a nonlinear relationship between independent and the dependent variables [15]; thus, they can predict stock index prices to a good extent.

In this study, a novel approach for stock price prediction is implemented in which a customized neural network is developed and the power of hyperparameter optimization in modelling stock index prices is investigated. A novel dataset is generated using nine standard technical indicators and COVID-19 data. In addition, the importance of selecting significant features is emphasised. A novel feature selection approach is developed in which multiple feature ranking algorithms are used for optimal feature selection. The extensive hyperparameter optimization procedures are employed for optimal predictions. Moreover, the model is evaluated by comparing it with other models, and results show that the proposed model outperforms other models.

Some of the notable contributions of this study are as follows:

(i) A novel dataset has been generated using technical indicators and COVID-19 data.

(ii) An exhaustive hybrid feature engineering has been performed where five feature selection techniques have been applied to the data.

(iii) An improved ANN has been built and trained using an automatic hyperparameter tuning procedure.

(iv) The performance of the proposed model has been compared to other models.

2. Background

The ANN is one of the popular and intelligent techniques that has a capability to adapt to the dynamic nature of nonlinear data [16]. It is one of the bioinspired algorithms that mimics the function of brain. By detecting the patterns in data, ANN gathers knowledge and learns through experience [17, 18]. ANN is typically a layered network consisting of one or more artificial neurons or processing elements. The interconnection of the neurons contributes to the computational power of ANN. Each neuron comprises of weighted inputs, an activation function, and an output. Figure 1 presents the typical structure of neuron. The weighted sum of inputs is passed through activation function to generate the output. The activation function leads to nonlinearity in network [19]. As shown below, Y_{in} is the weighted sum of inputs and the output Y is the function of Y_{in}.

$$Y_{\text{in}} = X_1.W_1 + X_2.W_2 + X_3.W_3 + \ldots X_n.W_{\text{n}}.$$

$$\text{That is net input } Y_{\text{in}} = \sum_{i}^{n} X_i.W_i,$$

$$Y = F(Y_{\text{in}}). \tag{1}$$

2.1. Feed Forward Neural Networks. Feed Forward Neural Networks (FFNNs) are the simplest type of ANN where the processing is done in only one direction (forward) with no cycles or loops [20, 21]. The processing of input data starts with input layer and passes through hidden layers and then to output layer. The construction of FFNN can be done from different units like binary McCulloch–Pitts neurons [22, 23]. An example of an FFNN is a perceptron. Figure 2 presents the structure of simple FFNN.

2.2. The Error Back Propagation. It is a multilayer FFNN with a series of hidden interconnected layers containing an arbitrary number of neurons. Back Propagation (BP) represents how the error that comes at the last layer of neural network is percolated back so that each of the individual layers can be trained [24]. In the training process, weights are adjusted in a way to reduce the error in predictions to a minimal value. Through BP, errors are minimized by updating the weights and minimizing the cost function using the gradient descent approach [25, 26]. Figure 3 presents the BP neural network having three layers: input, one hidden, and output layer. The nodes are connected through the links.

Between the input and hidden layer weights are $W1$ and between the hidden layer and output layer weights are $W2$. The last layer parameters can be trained directly as they directly impact the loss function but to train the weights of previous layers the feedback needs to travel backwards [27]. The sequence in which weights will be updated is from $W2$ to $W1$. Weights of $W2$ will be updated as follows:

$$W2 = W2 - \alpha * \Delta W2, \tag{2}$$

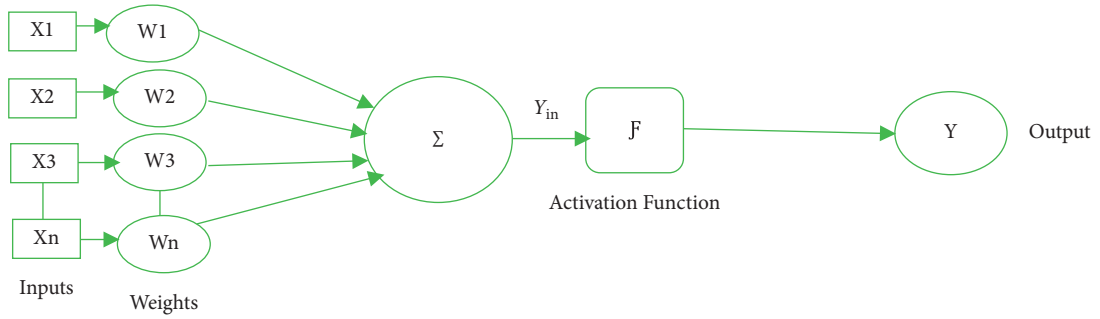

Figure 1: Structure of a neuron.

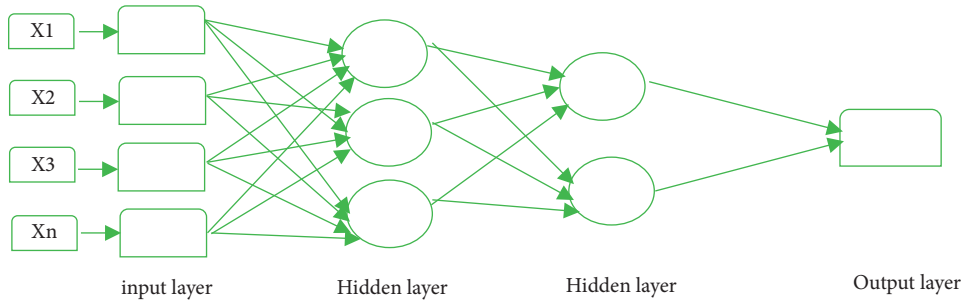

Figure 2: Structure of a feed forward neural network.

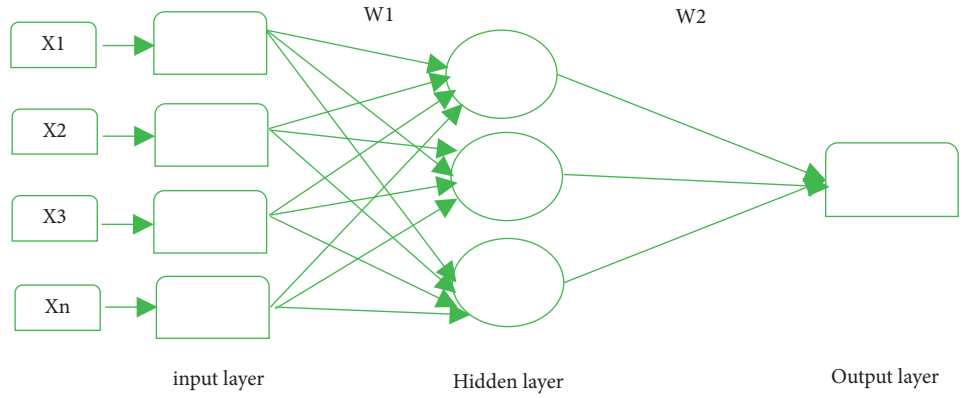

Figure 3: The error back propagation in neural networks.

where $\Delta W2$ is $(Y' - Y)^2$ and for $W1$ the weights will be updated as follows:

$$W1 = W1 - \alpha * W1, \tag{3}$$

where $\Delta W1$ is directly proportional to $\Delta W2$, which means that any change in $\Delta W2$ will introduce a change in $\Delta W1$.

2.3. The Loss Function. At the heart of any machine learning algorithm is a loss function, which is the sum measure of errors that we are trying to minimize. A way to minimize this loss is to find out the point over the function where it exhibits the minimum value and gradient/error tends to zero. The slow clipping away of the path to minima is known as gradient descent. In regression problem, the mean squared error is usually chosen as a loss function [28]. The generalized formula is given in the equation below.

$$J = \left(\frac{1}{2n}\right) * \sum_{j=1}^{n} \left[Y_j' - Y_j\right]^2. \tag{4}$$

J is the loss function wherein for each example starting from $j = 1$ to n from true value the predicted value is subtracted and squared and added across all the examples, and it is divided by $2n$, where n is the number of examples, where Y_j' is true value and Y_j is predicted value.

$$Y_j = F\left(\sum_{i=1}^{m} X_i \cdot W_i\right)^2. \tag{5}$$

Y_j or predicted value is given by summation from 1 to m, where m is the number of features and multiplying each feature with corresponding weights and adding it.

If we have to find the optimum parameters which are typically the weights that are between neurons. The loss function will be decreased by decreasing these weights by a factor of alpha (α) multiplied by $(\partial J/\partial W)$ as shown in the following equation.

$$W = W - \alpha * \frac{\partial J}{\partial W}, \tag{6}$$

where α is the learning rate and J is the cost function. At the end, the objective is to minimize the loss function constrained on W given as

$$\text{argmin } J \longrightarrow \text{optimized weights } W. \tag{7}$$

2.4. The Weight Initialization. There are various methods for initializing weights in neural networks [29]. To begin with, zero initialization initializes all weights to zero. It does not give good results because all weights start at the same point. So, it takes longer time to converge to minima because of lack of generalization [30]. In normal initialization, weights are sampled from a normal distribution. It gives better results and tends to provide more room for exploration in terms of feature space. Glorot/Xavier initialization is variation of normal initialization where the weights are initialized from a normal distribution centered at 0 and variance is $2/(\text{fan\_in} + \text{fan\_out})$ [31]. Fan-in is the number of nodes in previous layer and fan-out is the number of nodes in next layer. This initialization gives good results and helps to converge quickly by taking into consideration the architecture of network. In the similar line is He initialization, which is a random initialization depending on the size of previous layer and helps to attain global minimum of cost function faster and more efficiently [30].

3. Related Work

A wide range of problems are solved using ANNs due to its versatile nature. With the introduction of multilayer concept, ANNs are getting more attention for prediction besides other methods. ANNs are built primarily on the features extracted from input data. ANNs deal with the nonlinear relationships of time series data by learning from experience and building complex neural networks to generate optimal solutions for prediction problems [32, 33]. ANN consists of input layer, zero or more hidden layers, and an output layer. It is one of the frequently used machine learning algorithms that is applied to time series data for stock trend, index, and market predictions [34]. Some of the studies on stock market predictions using ANNs are discussed below.

Yudong and Lenan [35] developed a BP ANN model where weights are updated using IBCO tool to predict the next day and 15-day ahead price movement of stock index. The stock data of Standard and Poor's (S&P500) of the United States of America (USA) is used along with the technical indicators. The model is compared with {9-3-1}, and the proposed method achieved significant results.

Bijari and Bijari [36] developed a hybrid ANN where the model follows two phases. In the first phase, an Autoregressive Integrated Moving Average (ARIMA) model is used to extract essential features from time series data. In the second phase, ANN model is used to generate predictions from extracted data. The model is applied to three well-known datasets: the Canadian lynx data, the US dollar exchange rate data, and the Wolf's sunspot data. The model performance is calculated using MSE and Mean Absolute Error (MAE). Comparison is done with the ANN, ARIMA, and Zhang's model [37]. The proposed model achieved a better accuracy as compared to other mentioned models.

Kara et al. [38] compared the performance of two models, ANN and SVM, in predicting the daily direction of ISE National 100 index. The authors used ten technical indicators and extensive parameter tuning for models was performed to improve prediction accuracies. The average performance of SVM was around 71% and for ANN was around 75%, which is significantly better than the former.

Guresen et al. [39] compared the three stock market prediction models, namely, multilayer perceptron (MLP), dynamic artificial neural network (DAN2), and hybrid neural networks, which use generalized autoregressive conditional heteroscedasticity (GARCH) to generate new input variables [39]. Each model was evaluated on two points: MSE and Mean Absolute Deviate (MAD). The models were applied to NASDAQ stock data. The results showed that MLP outperformed DAN2 and GARCH-MLP.

Bing et al. [40] did an extensive comparison between eight models for prediction of stock value of central bank of Turkey. The six models were based on ANNs, where the number of hidden layers was changed in each model and other two models were based on moving averages. The models were evaluated using coefficient of determination. The ANN models with one hidden layer outperformed all models.

Yetis et al. [21] used generalized feed forward neural networks to predict the stock value of NASDAQ index. The model was trained on the one-year share market data of NASDAQ [21]. The model performed best with five parameters and 10 neurons in input layer. The performance of model was calculated using MSE. Furthermore, the authors have drawn a conclusion that when stock index prices are greater than 3000, the error reduces to less than 2%.

Patel et al. [41] compared the two approaches to predict the stock values of two market indices (CNX Nifty and S&P Bombay Stock Exchange Sensex) and two companies (Reliance Industries and Infosys Ltd.). The market data from 2003 to 2012 is extracted. The first approach focuses on the calculation of ten technical indicators from market data while in the second approach, the technical indicators are represented as trend determination layer. Four models, ANN, SVM, random forest, and naïve Bayes, are applied. In this first approach, random forest outperforms all other models and in the second approach the performance of each model increases.

Wang et al. [42] proposed a hybrid model that combines the powers of Elman recurrent neural network with stochastic time effective function. The model was tested on four indices: SSE, TWSE, KOSPI, and Nikkei225. The performance of model was compared with stochastic time effective neural network, backpropagation neural network, and Elman recurrent neural network. From the empirical results, it was revealed that the proposed model outperformed all other mentioned models.

Hajirahimi and Hajirahimi [43] compared the predictive power of five hybrid models using serial and parallel approaches. Two serial hybrid models ARIMA-MLP and MLP-ARIMA and three parallel models based on simple average, genetic algorithm, and linear regression were applied to two datasets SZII and S&P 500. The results indicate that all hybrid models with serial combinations outperformed other hybrid models.

Chopra et al. [34] investigated the prediction ability of ANN before and after demonetization in India. The ANN with Levenberg–Marquardt algorithm is applied to the data of nine companies and stock index CNX Nifty50. The model performance is measured using MSE. The regression value of 0.99 is achieved, which depicts the efficiency of model.

Vijh et al. [15] compared the ANN and Random Forest models to predict the next day stock prices of five companies. The market data is used to generate the technical indicators RMSE and MBE and MAPE metric is used to evaluate the models. ANN outperforms RF and obtains low values of performance metrics.

The following are the few papers that have analyzed the impact of COVID-19 on stock markets:

Zeren and Hizarchi [44] have used regression method to investigate the effect of COVID-19 on the stock market and discovered a significant relationship between COVID-19 variables and stock returns. Aravind and Manojkrishnan [45] investigated the association between pharmaceutical stocks and COVID-19 and found that pharmaceutical stocks had a negative reaction. Zhang et al. [46] investigated the effect of COVID-19 on financial markets using statistical analysis revealed that COVID-19 has raised global financial market risk. Individual stock market volatility is directly proportional to the intensity of viral outbreak in that country. Goodell [47] analyzed a large body of material on the economic impact of natural disasters and noted that the global economic destruction caused by COVID-19 is unparalleled. Baker et al. [48] used text analysis and other methods to investigate the influence of COVID-19 news on stock market volatility. The results revealed that COVID-19 affected the markets to greater extent than similar other diseases. Bekhet et al. [5] have developed a technique for early COVID-19 detection using medical experiences. The hyperparameter optimization of light-weight Convolution Neural Networks has been done and authors achieved an accuracy rate of 96% on some benchmark datasets. Ibrahim et al. [6] have developed a hierarchical twitter sentiment model (HSTM) to analyze the sentiment of people on High Tech Computer (HTC) product and COVID-19 in Twitter messages without predetermining the depth and width hierarchy of first discovered hierarchical tree. The authors have employed valence-aware dictionary and sentiment reasoner (VADER) sentiment analyzer to analyze sentiments of people and have grouped sentiments into groups like positive, negative, strongly positive, and strongly negative. The results show that HSTM analyzes the sentiments of people in an easy and effective way.

Some of the conclusions that can be drawn from the surveyed literature are as follows:

(1) It is evident from the surveyed literature, not much work has been done on impact of COVID-19 on stock markets. Mostly, either statistical analysis or regression analysis has been used in the literature so far.

(2) Identification of determining features is imperative as it may increase the prediction accuracies [41]. Moreover, all the presented works follow a certain feature selection algorithm and select the features that are best voted. However, each algorithm has a selection criterion, which may or may not benefit your model. Therefore, relying on a single feature selection algorithm is not a good practice.

(3) Moreover, technical indicators have a significant impact on investment decisions [49]. Therefore, it is important to consider the preprocessing and importance of standard technical indicators.

Thus, in this study, an idea is devised to derive the important technical indicators from the existing market data along with the COVID-19 data and apply multiple feature selection approaches to generate the important features so that the model prediction accuracy increases significantly. Furthermore, the hyperparameters govern the overall training process, so it is imperative to understand the hyperparameters and fine-tune them to get optimal predictions.

4. Methodology

This work attempts to address three research questions:

(i) What is the effectiveness of COVID-19 and derived technical indicators on the prediction model?

(ii) How effective is feature engineering in increasing the model performance?

(iii) How the hyperparameter optimization procedures effect the model performance?

Our first question is based on the effectiveness of the COVID-19 and technical indicators. As seen from the previous works, technical indicators play a significant role in predicting the markets [50]. Also, COVID-19 has significantly affected the financial markets [7]. In our approach, an attempt has been made to predict the Nifty50 index prices using basic market data COVID-19 data and derived technical indicators. Second research question is based on the effectiveness of feature engineering. From the previous works, it can be concluded that stock data is chaotic and features are highly correlated, which makes it difficult to predict markets with significant accuracies. Therefore, it is

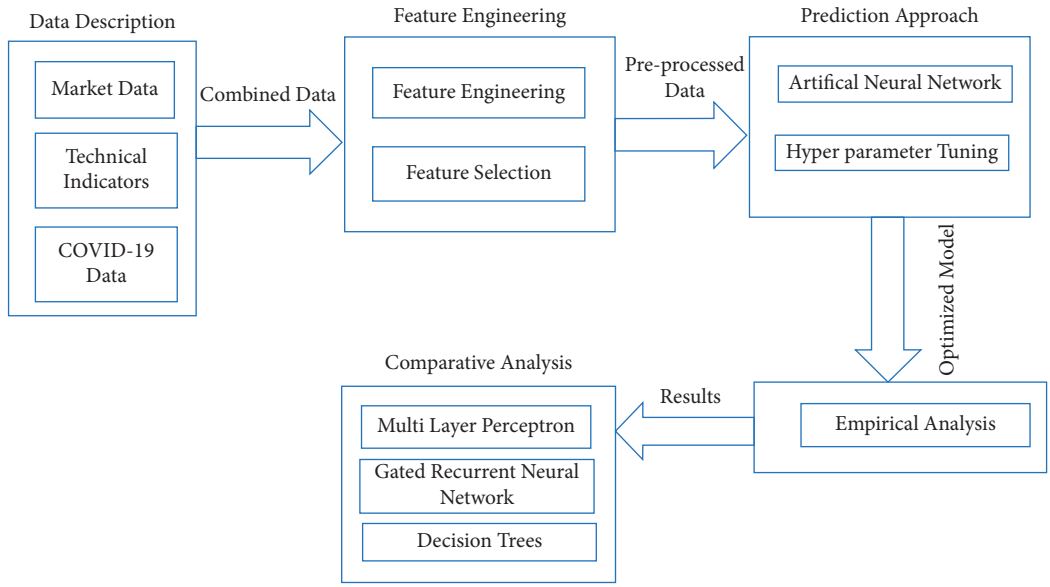

Figure 4: Methodology for stock market prediction.

essential to apply feature engineering procedures to stock data so that the results may get optimized. In our approach, multiple feature selection models have been applied to select the best features and the average means of coefficients of all models have been used to select the features that have higher average impact on dependent variable. Furthermore, we check how the selection of different hyperparameters impact the model accuracies. Figure 4 presents the methodology of the work carried out.

4.1. The Dataset Description. This section provides the detailed introduction to research data and extracted technical predictors. The openly available Nifty50 open high, low, close and volume (OHLCV) data is retrieved from the following website (https://finance.yahoo.com) for a period of two years. The choice of choosing two-year time span is due to the reason that investors usually perform stock trend analysis using the recent data mostly within 2 years [51]. Massive volumes of COVID-19-related data are constantly being generated as a result of the rapid growth of big data technologies, making it challenging to access correct information resources [52, 53]. The COVID-19 data has been collected from a reliable source https://www.ecdc.europa.eu/en that gives access to the COVID-19 data for more than 200 virus-affected nations and regions. Furthermore, nine technical variables have been added to the market data. There are a wide range of technical indicators. Some indicators perform better under trending markets, and some are effective under nontrending market [54]. From these previous works, it is concluded that with the addition of technical indicators prediction models perform better [35]. Based on the previous works, we selected nine technical indicators, out of which seven have 4 levels [15, 38, 41]. Table 1 presents the selected technical indicators and their formulas. As the COVID-19 dataset contained the data for more than 200 countries, we extracted the data by setting a query for extraction of "Daily_Cases" of "India" in python. Furthermore, we added a feature "Total_Cases," which is a cumulative sum of "Daily_Cases." We merged the two datasets on the basis of a common feature "Date."

4.2. Exploratory Feature Analysis. In this section, first, data cleaning is performed by filling the missing values using averaging method. Furthermore, outliers have been detected using simple statistical method called Interquartile Range (IQR) [55]. IQR is the difference between the first ($Q1$) and third quartiles ($Q3$) of data. For each feature in dataset, the data beyond the range of ($Q1 - 1.5$ IQR to $Q3 + 1.5$ IQR) is possible outlier. The symmetric distribution is visualized using box plots for each column.

4.3. Hybrid Feature Selection Mechanism. The feature selection leads to the better performance of prediction models. As analyzed from the previous studies, the noisy stock data makes predictions difficult; it is important to understand the features and structure of data so that predictions can be optimized [38].

We list three primary reasons why feature selection is important. Firstly, with the reduction in the number of features, less chances of model overfitting are there. The second reason is to extensively understand the underlying features and their relationships with the dependent variable. Lastly, it reduces the training time of the algorithm [51].

In this work, five feature selection methods have been employed. A python dictionary is maintained to store the coefficient scores or feature ranks obtained from all the methods applied. The mean coefficient value for each feature in the dictionary is calculated and the features with high mean coefficient values are selected for model training. The most important features have the highest coefficients in the model and least important features have coefficient values tending to zero.

Table 1: Technical indicators, formulas, and levels.

Technical indicators	Formula	Levels
Return	Change = $((C_t - C_{t-1})/C_{t-1})$ If change is greater than 0 return = change, else return = 0	1
Volatility	SD (change) ∗ sqrt (360)	1
Simple n day moving average	$C_t + C_{t-1} + \cdots + C_{t-n-1}/n$	4 (3, 6, 9, 12)
Weighted n day moving average (WMA)	$(((n)C_t + (n-1)C_{t-1} + \cdots + C_{t-(n-1)})/(n + (n-1) + \cdots + 1))$	4 (3, 6, 9, 12)
Close lag		4 (3, 6, 9, 12)
RSI	$100 - (100/(1 + (\sum_{i=0}^{n-1}(\mathrm{UP}_{t-i}/n))/(\sum_{i=0}^{n-1}(\mathrm{DW}_{t-i}/n))))$	4 (3, 6, 9, 12)
Moving average convergence/divergence (MACD)	$\mathrm{MACD}(n)_{t-1} + (2/(n+1)) * (\mathrm{DIFF}_t - \mathrm{MACD}(n)_{t-1})$	4 (3, 6, 9, 12)
Momentum stochastic (K%)	$((C_t - \mathrm{LL}_{t-(n-1)})/(\mathrm{HH}_{t-(n-1)} - \mathrm{LL}_{t-(n-1)})) * 100$	4 (3, 6, 9, 12)
Commodity channel index (CCI)	$((M_t - \mathrm{SM}_t)/(0.015 * D_t))$	4 (3, 6, 9, 12)
Total		30

Note. n is the time period n days ago, C_t, L_t , and H_t are the closing, lowest, and highest prices at time t, respectively, SD is the standard deviation, SQRT is the square root function, LL_t and HH_t are the lowest low and highest high price in last t days, respectively, UP_t and DW_t are the upward and downward index change at time t, DIFF is the $\mathrm{EMA}(12)_t - \mathrm{EMA}(26)_t$, where EMA is exponential moving average, $\mathrm{EMA}(p)_t = \mathrm{EMA}(p)_{t-1} + \propto (C_t - \mathrm{EMA}(p)_{t-1})$, where $\propto = 2/(1+p)p = 10$ in p-day EMA. $M_t = ((C_t + L_t + H_t)/3)$; $\mathrm{SM}_t = \sum_{i=1}^{n}(M_{t-i+1}/n)$; and $D_t = \sum_{t=1}^{n}(|M_{t-i+1} - \mathrm{SM}_t|/n)$.

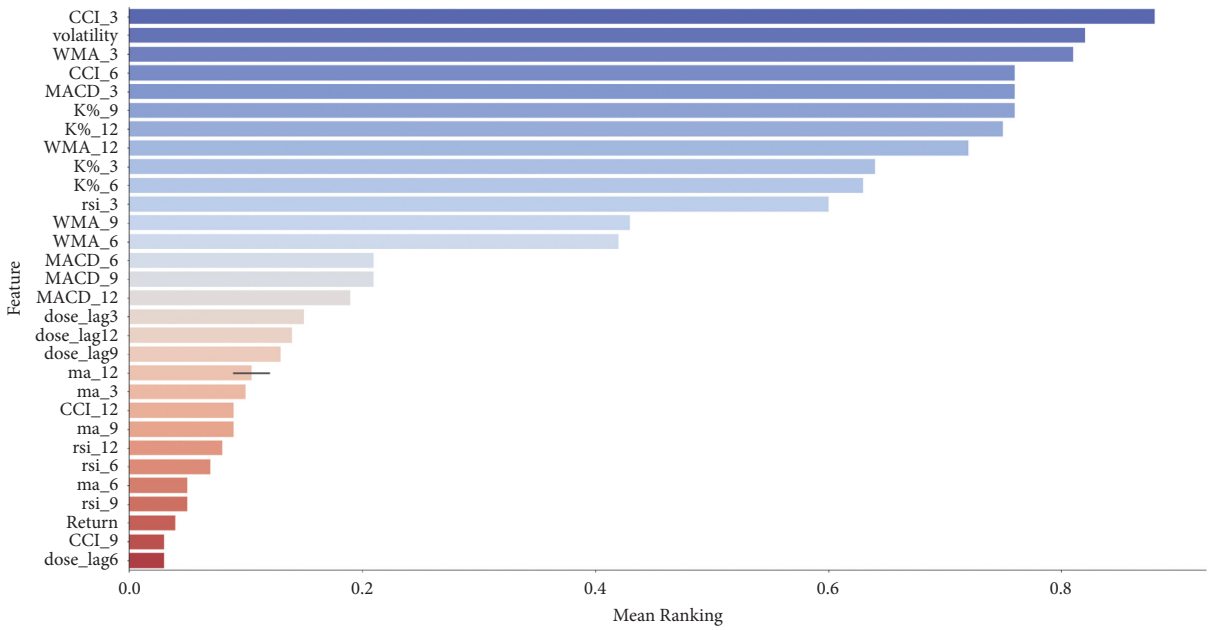

Figure 5: Mean feature rankings.

The models that have been employed are Lasso regression, Ridge regression, Random Forest regression, Stability selection via Randomized Lasso, and Recursive feature elimination (RFE) via Linear Regression. In lasso regression, feature selection is done by picking up the features that have a significant effect on dependent variable. This model is useful for reducing the feature count; however, it fails to interpret the features [56]. Ridge regression model equally distributes the coefficient values between correlated variables [57]. Random forest model with impurity-based ranking selects the top features, and the coefficients of other features aggressively drops off [58, 59]. Stability selection model takes into consideration both the data interpretation and feature importance to improve the models [60]. RFE is widely used algorithm for feature selection. It is based on the greedy optimization to select the significant subset of features [61]. The mean rankings for all the features are visualized in Figure 5. The features are selected where mean ranking is greater than or equal to 0.5. Therefore, out of 30 features, we have selected 10 features as input to our prediction model. Furthermore, it can be analyzed from the Figure 5 that "Daily_cases" feature has a mean feature ranking of greater than 0.5; therefore, it is an important feature and greatly affects the stock market price prediction. Algorithm 1 presents the pseudocode for hybrid feature selection mechanism.

4.4. Prediction Approach. ANN is a set of densely interconnected processing units called neurons that are activated through input. The learning mechanism involves the adaption of weights in a way so that prediction error is minimized [62]. In this work, we have employed a four-layer neural network. The input layer has 11 features represented

```
    INPUT: stock dataset
    OUTPUT: feature rankings
    # Define dictionary to store feature ranking
(1) SET ranks TO {}
    #Create function which stores the feature rankings to the ranks dictionary
(2) DEFINE FUNCTION ranking(ranks, names, order = 1):
(3)    SET minmax TO MinMaxScaler()
(4)    SET ranks TO minmax.fit_transform(order* np.array([ranks]).T).T[0]
(5)    SET ranks TO map(lambda x: round(x, 2), ranks)
(6)    RETURN dict(zip(names, ranks))
    #Run Selection Stability method with Randomized Lasso
(7)    SET rlasso TO RandomizedLasso(alpha = 0.04)
    rlasso.fit(X, Y)
(8) SET ranks["rlasso/Stability"] TO ranking(np.abs(rlasso.scores_), colnames)
(9)    OUTPUT("finished")
    #Use RFE and stop the search when only the last feature is left
(10)   SET rfe TO RFE(lr, n_features_to_select = 1, verbose = 3)
    rfe.fit(X, Y)
(11) SET ranks["RFE"] TO ranking(list(map(float, rfe.ranking_)), colnames, order = −1)
    # Using Ridge
(12)   SET ridge TO Ridge(alpha TO 7)
    ridge.fit(X,Y)
(13)   SET ranks["Ridge"] TO ranking(np.abs(ridge.coef_), colnames)
    # Using Lasso
(14)   SET lasso TO Lasso(alpha = 0.05)
    lasso.fit(X, Y)
(15)   SET ranks["Lasso"] TO ranking(np.abs(lasso.coef_), colnames)
    # Using Random Forest
(16)   SET rf TO RandomForestRegressor(n_jobs = −1, n_estimators = 50, verbose = 3)
    rf.fit(X, Y)
(17 )  SET ranks["RF"] TO ranking(rf.feature_importances_, colnames);
    #Create empty dictionary to store the mean value calculated from all the scores
(18)   SET r TO {}
(19)   FOR name IN colnames:
(20)   SET r[name] TO round(np.mean([ranks[method][name]
(21)      FOR method IN ranks.keys()]), 2)
(22)   SET methods TO sorted(ranks.keys())
(23)   SET ranks["Mean"] TO r
    methods.append("Mean")
(24)   OUTPUT("\t%s" % "\t".join(methods))
(25)   FOR name IN colnames:
(26)   OUTPUT("%st%s" % (name, "\t".join(map(str,
    [ranks[method][name] FOR method IN methods])))
    #Put the mean scores into a Pandas dataframe
(27) SET meanplot TO pd.DataFrame(list(r.items()), columns = ["Feature," "Mean Ranking"])
    #Sort the dataframe
(28)   SET meanplot TO meanplot.sort_values("Mean Ranking," ascending = False)
    #Plot the ranking of the features
(29)   SET sns.factorplot(x = "Mean Ranking," y = "Feature," data TO meanplot, kind = "bar," size = 14, aspect = 1.9,
    palette = "coolwarm")
```

Algorithm 1: Hybrid feature selection mechanism.

by 11 neurons as input. The output layer has a single neuron with no activation function owing to the fact that it is a regression problem. For each hidden layer, we have employed ReLU activation function. To maintain the balance between variance and bias in a model, it is imperative to select optimal hyperparameters [63].

An extensive hyperparameter tuning has been done for model fitting. The hyperparameters are number of neurons in the input layer, number of hidden layers (*nh*), learning rate (*lr*), number of epochs (*ne*), batch size, dropout, activation function, and shapes of which are discussed as follows: "*nh*" and the number of neurons significantly impact the prediction accuracies. Choosing at least one of the parameters with higher values allows the algorithm to model the relationship between dependent and independent variables effectively. "*lr*" is an essential hyperparameter. If "*lr*" is set high, then the algorithm may adopt to new data quickly but will forget old data and that results in divergence. If "*lr*"

```
    INPUT: stock dataset
    OUTPUT: predictions
    #set up the params dictionary
(1) SET p TO {"lr": (0.1, 0.5, 10),
          "first_neuron": [10, 20],
          "hidden_layers": [1–3],
          "batch_size": (10, 30, 5),
          "epochs": [100],
          "dropout": (0, 0.5, 5),
          "shapes":["brick," funnel"],
          "activation": ["relu"],
          "optimizer": ["RMSProp," "Adadelta," "Adam," "Adagrad"]
        }
    #Build the model
(2) DEFINE FUNCTION stock_prediction_model(x_train, y_train, x_val, y_val, params):
      # next we build the model exactly like we would normally do it
(3)   SET model TO Sequential()
      model.add(Dense(params["first_neuron"], INPUT_dim = xtrain.e [1],
    activation = "relu,"
            kernel_initializer = "he_normal"))
            model.add(Dropout(params["dropout"]))
      hidden_layers(model, params, 1)
      model.add(Dense(1, activation = "relu,"
            kernel_initializer = "he_normal"))
      model.compile(loss = "mean_squared_error,"
            optimizer = params["optimizer"],
            metrics = ["mape"])
            RMSProp(learning_rate = params["lr"]),
            metrics = ["mape"])
(4) SET history TO model.fit(x_train, y_train,
            validation_data = (x_val, y_val),
            batch_size = params["batch_size"],
            epochs = params["epochs"],
            verbose = 0)
    # history object and model RETURNed
(5) RETURN history, model
    #Use Talos for hyper parameter tuning
(6)   SET t TO talos.Scan(x = X_train,
(7)   SET y = Y_train, params TO p,
        x_val = X_val, y_val = Y_val,
        model = stock_prediction_model)
```

Algorithm 2: Hyperparameter tuning of ANN.

is set low, the training algorithm learns slowly and will be less susceptible to outliers and noise [64]. Therefore, we should start with a high "*lr*" value and decrease it till model is optimized. Epoch size represents the number of times the training algorithm will perform through entire training set [65]. Therefore, "*ne*" is an integer that can have values between 0 to infinity. The value of "*ne*" needs to be chosen carefully, as the lower values cause under fitting and higher values cause overfitting due to overoptimizing the weights [63]. Batch size determines learning speed of model and keeps a check on how stable a learning process is. In order to increase the performance of model, the selection of right optimizer is important. So here we have used different variants of stochastic gradient descent (SGD) optimizers like RMSProp, AdaGrad, AdaDelta, and Adam. The description of each variant and differences can be found in [64]. Pseudocode for the feature selection mechanism is presented in Algorithm 1.

Regularization methods enable neural networks to select the models that have efficient generalizing capabilities. Here the idea is to reduce the variance on training datasets, without increasing the bias values. Implementation of regularization using dropout layers is one of the popular techniques. This approach has proven successful, and it can boost the prediction accuracies up to 2% [66]. The value of dropout rate varies between 0 and 1. Whenever there is a possibility of overfitting in model, dropout rate should be increased and conversely, dropout rate should be decreased when model is under fitting the training data [64]. Shapes for hidden layers could be funnel or brick. In funnel shape, the first layer starts with maximum of neurons and subsequently the number of neurons in each layer decreases ending with

one neuron in the last layer. While in the brick shape, each hidden layer has the same number of neurons and the last layer contains a single neuron.

To explore the hyperparameter space, different approaches are used like manual search, gird search, and random search. Although, from the empirical evidence, random search method has proven to be efficient in finding the optimal models than grid and manual search approaches [67, 68], however if the dataset is large and complex, this method only explores a tiny hyperparameter space. Moreover, this method is time consuming for high-dimensional data. Therefore, a more efficient solution is to use the python libraries for optimizing the hyperparameters. The main idea is to explore more region of space. The advantage of using these libraries is that it takes care of "zooming" process, which ultimately leads to a better model training in less time [64]. In this work, we have used Talos API for automated hyperparameter optimization that works well on all sorts of complex search spaces [69]. Here the optimal model can be retrieved by running the code with all combinations one time, instead of running code for each combination separately. Furthermore, this library creates an efficient pipeline by following prepare, optimize, and deploy workflow. Therefore, Talos is handy, time-efficient, and powerful method for parameter optimization. Algorithm 2 presents the hyperparameter tuning procedure using Talos.

4.5. Performance Analysis. At the heart of any machine learning algorithm is a loss or cost function that calculates the prediction error. The aim is to minimize this error as much as possible. After each iteration, the model calculates the cost function and accordingly adjusts the weights to minimize the error. In our model, we have used mean squared error (MSE), which is the default cost function for regression models, and calculated the average mean of squared differences between actual and predicted values [70]. The MSE ranges between 0 and infinity. The smaller MSE indicates the higher accuracy of models. For a given model, the value of evaluation metrics determines how accurately the model is performing. The model performance is evaluated using percentage-dependent metric, mean absolute percentage error (MAPE) that measures the prediction accuracies, mainly in trend estimation [70]. The MAPE is calculated as follows:

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{V_{\text{pred}} - V_{\text{act}}}{V_{\text{act}}} \right|. \quad (8)$$

5. Empirical Analysis

The preprocessed data is divided into training set, test set, and validation set. The 80% of total data is used for training and 50% of remaining 20% is used for test and remaining 50% for validation. The hyperparameters that are used are mentioned in table. The experimentation has been carried out on Google colab with a Python 3 Google Compute Engine backend (TPU). A function has been defined in python to build a basic sequential model using dense and dropout layers. The model is visualized using the pydot library as shown in Figure 6.

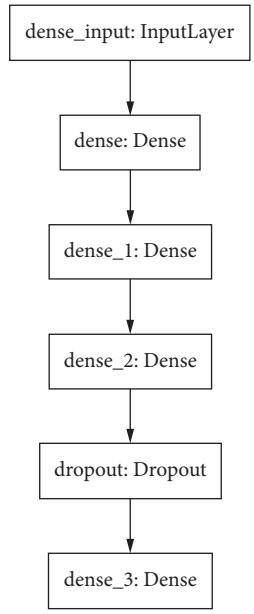

Figure 6: Neural network architecture.

Table 2: Hyperparameters with range of values.

Hyperparameters	Values
Learning rate	Range (0.5, 0.1) 10 values
First_neuron	[10, 20]
Hidden layers	[1, 2, 3]
Batch_size	Range (10, 30) 5 values
Epochs	100
Dropout	Range (0, 0.5) 5 values
Shapes	[Brick, funnel]
Activation	ReLU
Optimizer	RMSProp, AdaDelta, Adam, AdaGrad

5.1. Hyperparameter Tuning Analysis. To tune the hyperparameters, we install the Talos library and import the required dependencies. A hyperparameter dictionary is built in which different ranges for hyperparameters are defined for tuning the model. The layout of dictionary is as follows: for "*lr*" ten different values are taken from 0.1 to 0.5 that are equally separated. The number of neurons in input layer is taken as [10, 20], the "*nh*" in the range of 1 to 3, for batch size five different values are taken from 10 to 30, "*ne*" vary from [500, 1000], values of dropout layer are varied between 0 and 0.5, shapes for hidden layer are taken as "brick" or "funnel," activation function for hidden layers is rectified linear activation function (ReLU) that is nearly linear and has a power of easy optimization with gradient-based methods, and numerous optimizers are applied like RMSProp, AdaGrad, AdaDelta, and Adam.

Table 2 presents the range of values for selected hyperparameters. Instead of passing hypermeters, the dictionary is passed to the model that we built earlier. So here each hyperparameter is defined in a dictionary. The "Scan"

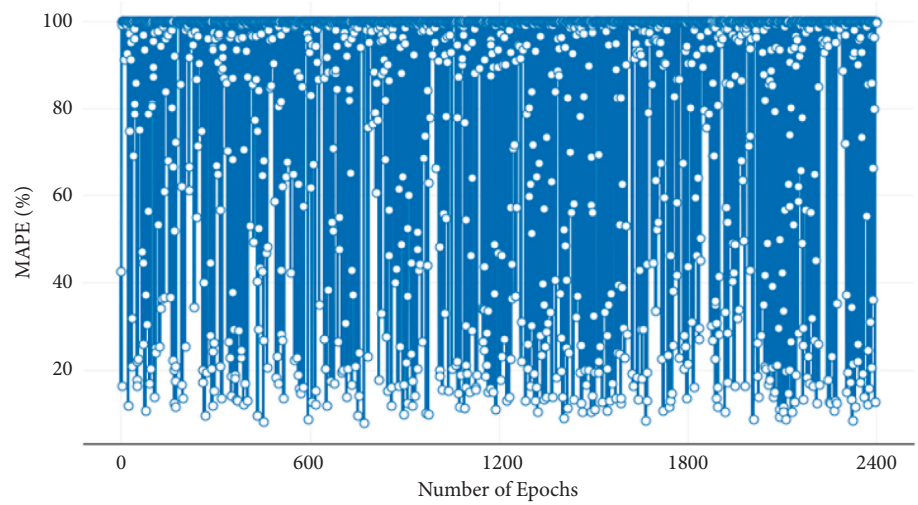

Figure 7: Variation of validation MAPE with number of rounds performed.

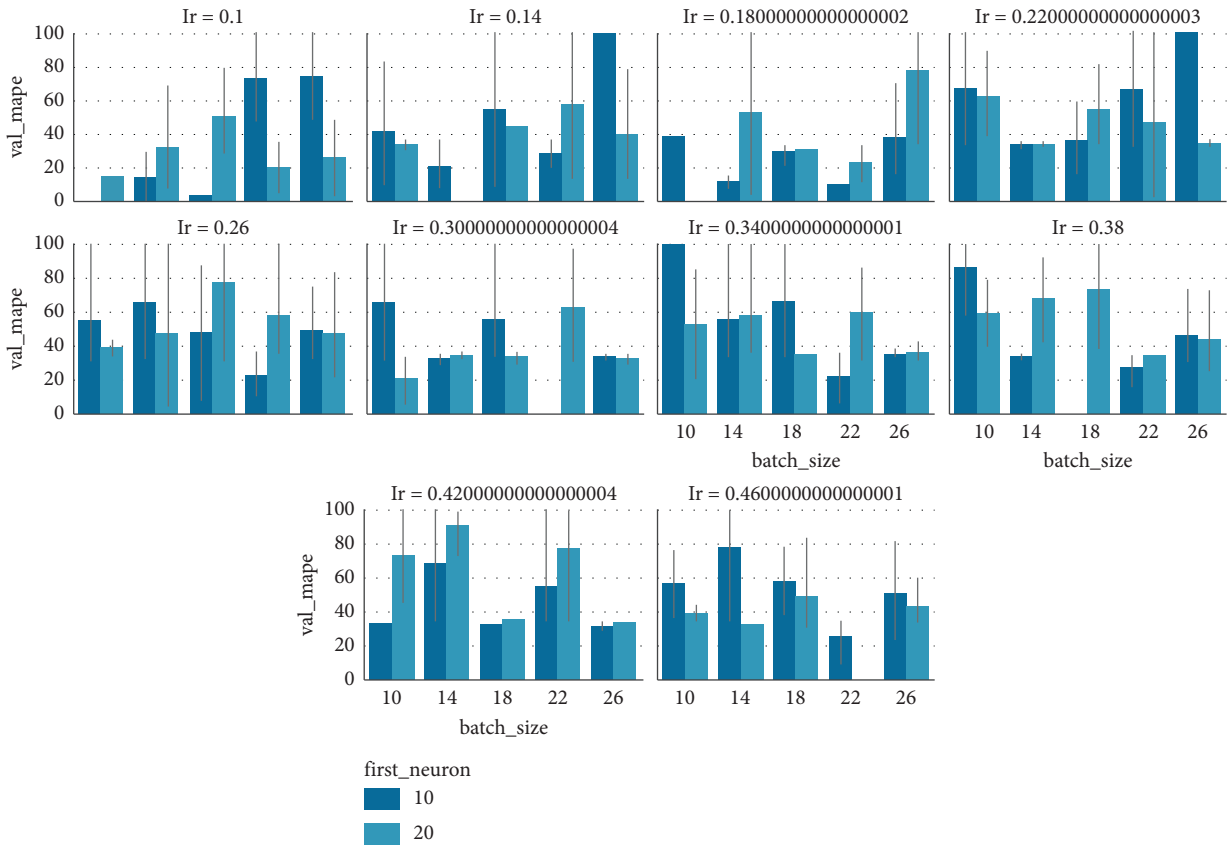

Figure 8: 4-dimensional bar plot representing the variation of validation MAPE by using different values for hyperparameters.

function of Talos is used for hyperparameter tuning by passing both the training and validation datasets. This function uses permutation combination of all hyperparameters. The "Analyze" object of Talos carries the results of hypermeter tuning activity. The number of rounds performed for parameter tuning was 2400. To get the lowest validation MAPE value, analyze_object.low() function is used, and it shows that in the 42th round, the lowest validation MAPE of 4.379007 is achieved.

The line plot in Figure 7 represents the variation in validation MAPE values with each round during the hyperparameter tuning. In addition, the four-dimensional bar grid plot in Figure 8 is used to visualize how validation MAPE is varying with different parameters. The dark blue bars represent 10 neurons and light blue represents 20 neurons in the first layer. It can be visualized from the bar plot that when "*lr*" is 0.1, the validation MAPE for 10 neurons was significantly lower than 20 neurons. To get the

Table 3: Optimized hyperparameters.

Parameters	Values
Start	07/02/21
End	07/02/21
Duration	2 hours and 45 minutes
Round_epochs	100
Loss	618206.8125
MAPE	8.039555
Val_loss	161916.171875
Val_MAPE	4.379007
Activation	ReLU
Optimizer	Adam
Batch_size	10
Dropout	0.0
Epochs	100
First_neuron	10
Hidden layers	2
Lr	0.14
Shapes	Funnel
Name	42, dtype: object

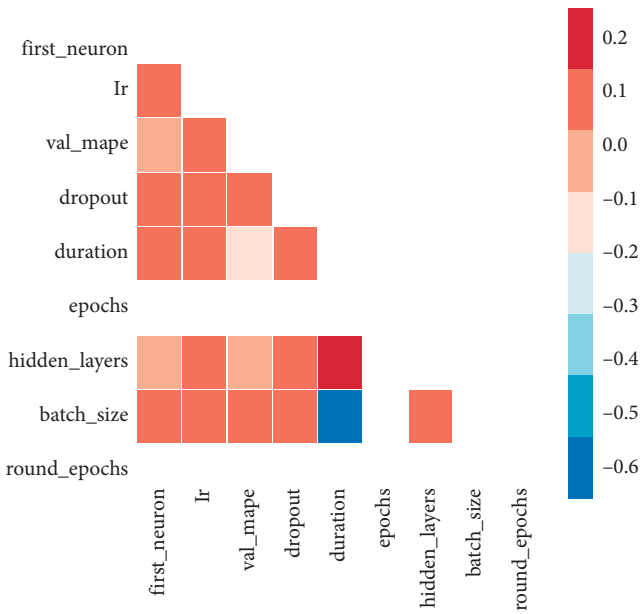

Figure 9: Correlation heat map of hyperparameters.

best model, talos. Best_model("metric: low MAPE") is used and the best model summary is shown in Table 3. For the test set, the MAPE value for the best model is 8.03. It can be interpreted that the model has good balance between variance and bias.

Moreover, the correlation heat map in Figure 9 represents the correlation between hyperparameters. It can be inferred from the heat map that validation MAPE is highly correlated with duration, number of hidden layers, and "*lr*."

6. Comparative Analysis

In this section, the proposed model is compared with the already existing models like multilayer perceptron (MLP) with three hidden layers, Decision Tree and Two-stream

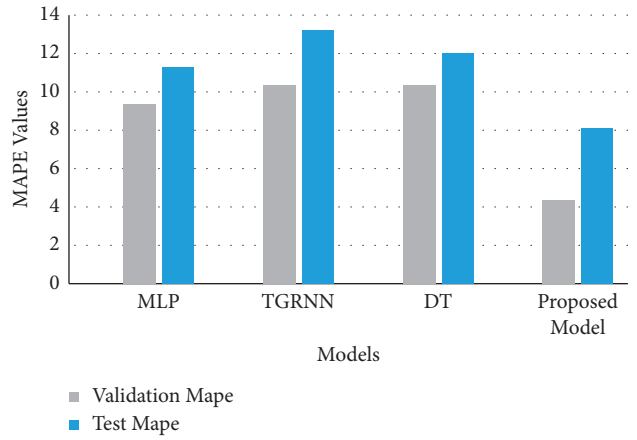

Figure 10: Comparison of MAPE scores of other models with the proposed model.

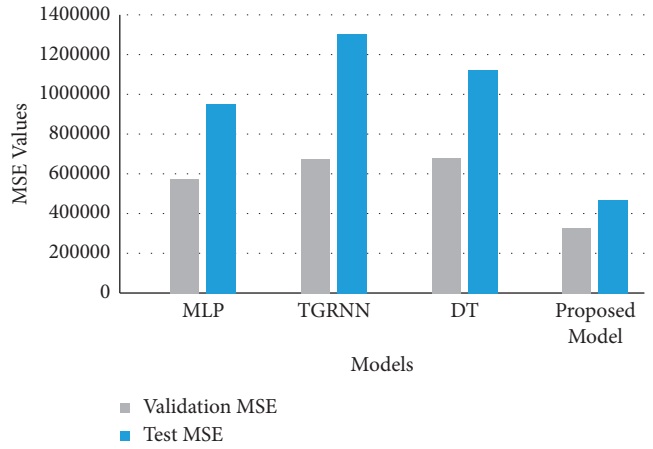

Figure 11: Comparison of MSE scores of other models with the proposed model.

Gated Recurrent Neural Network. The comparative graph is shown in Figures 10 and 11. It can be clearly interpreted that the proposed model with hyperparameter tuning achieved the lowest MAPE and MSE values for both validation and test sets.

7. Conclusions

The work is carried out in three parts: the data extraction and derivation of technical indicators, extensive feature engineering, and stock index price prediction using customized neural network with hyperparameter optimization. We extracted the structured Nifty50 index data and COVID-19 data. From the extracted market data, 9 technical indicators have been derived of which 7 have 4 levels. Five feature selection methods have been applied and averaged coefficient values have been used to select the correct and efficient features. A customized neural network has been built and regressive hyperparameter optimization procedures are applied to achieve faster and optimized predictions. The proposed model has been compared with three other models

and results indicate that the proposed model achieved highest accuracy. The MAPE scores for validation and test sets are 4.379007 and 8.03, respectively. In comparison with other models, the proposed model also achieved a low MSE score for both the validation and test sets. Hence, we conclude that the proposed model is a unique approach in comparison to previous works due to the fact that rather than proposing a yet another neural network model, we propose a customized neural network prediction system combined with the extensive feature engineering and hyperparameter optimization procedures to predict stock index prices. Furthermore, the "Daily_Cases" of COVID-19 in India have substantial effect on Nifty50 price changes. Although better results are achieved by the proposed model, this research can further be extended by including the sentiment data from diverse social media platforms, text databases, and news feeds. Furthermore, the model can be extended by analyzing the effect of both fundamental and technical indicators on market.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (no. 2021R1A2C2014333).

References

[1] T. Z. Tan, C. Quek, and G. S. Ng, "Biological Brain-Inspired Genetic Ccomplementry Learning for Stock Market and Bank Failure Prediction," *Computational intelligence*, vol. 23, no. 2, 2007.

[2] N. Rouf, M. B. Malik, and T. Arif, "Predicting the stock market trend: an ensemble approach using impactful exploratory data analysis," *Communications in Computer and Information Science*, pp. 223–234, Springer, Heidelberg, Germany, 2021.

[3] D. Selvamuthu, V. Kumar, and A. Mishra, "Indian stock market prediction using artificial neural networks on tick data," *Financial Innovation*, vol. 5, 2019.

[4] F. Wu, S. Zhao, B. Yu, Y. M. Chen, W. Wang, and Z. G. Song, "A new coronavirus associated with human respiratory disease in China," *Nature*, vol. 579, no. 7798, pp. 265–269, 2020.

[5] S. Bekhet, M. H. Alkinani, R. T. Soto, and M. Hassaballah, "An efficient method for covid-19 detection using light weight convolutional neural network," *computers, materials and continua*, vol. 69, no. 2, 2021.

[6] A. F. Ibrahim, M. Hassaballah, A. A. Ali, Y. Nam, and A. Ibrahim, "COVID19 outbreak: a hierarchical framework for user sentiment analysis," *computers, materials and continua*, vol. 70, no. 2, 2022.

[7] N. Rouf, M. B. Malik, and T. Arif, "A regression based approach to predict the Indian stock market trend amid COVID-19," in *Proceedings of the 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pp. 2014–2020, Greater Noida, India, December 2021.

[8] W. McKibbin and R. Fernando, "The global macroeconomic impacts of COVID-19," *Asian Economic Papers*, vol. 20, pp. 1–43, 2020.

[9] C. Iwendi, K. Mahboob, Z. Khalid, A. R. Javed, M. Rizwan, and U. Ghosh, "Classification of COVID-19 individuals using adaptive neuro-fuzzy inference system," *Multimedia Systems*, vol. 28, 2021.

[10] A. Kumar, N. K. Viswakarma, A. Adlakha, and K. Mukherjee, "How successful have the lockdowns been in controlling the (COVID-19/SARS-CoV-2) pandemic — a simulation-based analysis," *International Journal of Modeling Simulation and Scientific Computing*, vol. 12, no. 3, 2020.

[11] P. Gao, R. Zhang, and X. Yang, "The application of stock index price prediction with neural network," *Mathematical and Computational Applications*, vol. 25, no. 3, p. 53, 2020.

[12] K. j. Kim and W. B. Lee, "Stock market prediction using artificial neural networks with optimal feature transformation," *Neural Computing & Applications*, vol. 13, no. 3, pp. 255–260, 2004.

[13] W. Xu, J. Meng, S. K. S. Raja, M. P. Priya, and M. K. Devi, "Artificial intelligence in constructing personalized and accurate feedback systems for students," *International Journal of Modeling*, Article ID 2341001, 2021.

[14] Y. Zhao, "Artificial neural network modeling with application to nonlinear dynamics," *Computational Intelligence and Its Applications*, pp. 101–125, 2011.

[15] M. Vijh, D. Chandola, V. A. Tikkiwal, and A. Kumar, "Stock closing price prediction prediction using machine learning techniques," *procedia computer. Science*, vol. 167, pp. 599–606, 2019.

[16] R. Beresford, "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research," *Journal of pharmaceutical and biomedical analysis*, vol. 22, pp. 717–727, 2000.

[17] M. Babar, M. S. Khan, U. Habib, B. Shah, F. Ali, and D. Song, "Scalable edge computing for IoT and multimedia applications using machine learning," *Human-centric Computing and Information Sciences*, vol. 11, 2021.

[18] B. P. V. Milligen, V. Tribaldos, J. A. Jimenez, and C. S. Cruz, "Comments on "Accelerated learning algorithm for multilayer perceptrons: optimization layer by layer"," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 339–341, 1998.

[19] K. Koyamada, Y. Long, T. Kawamura, and K. Konishi, "Data-driven derivation of partial differential equations using neural network model," *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 12, no. 2, Article ID 2140001, 2021.

[20] M. Cottrell, B. Girard, Y. Girard, M. Mangeas, and C. Muller, "Neural modeling for time series: a statistical stepwise method for weight elimination," *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1355–1364, 1995.

[21] Y. Yetis, H. Kaplan, and M. Jamshidi, "Stock market prediction by using artificial neural network," in *Proceedings of the World Automation Congress Process*, pp. 718–722, Waikoloa, Hawaii, USA, August 2014.

[22] J. A. Anderson, "McCulloch-Pitts neurons," *encycl. Cogn. Sci*, pp. 1–5, 2006.

[23] M. Zhang, R. Fu, Y. Guo, L. Wang, P. Wang, and H. Deng, "Cyclist detection and tracking based on multi-layer laser scanner," *Human-centric Computing and Information Sciences*, vol. 10, no. 1, p. 20, 2020.

[24] M. Rizwan, A. Shabbir, A. R. Javed, M. Shabbir, T. Baker, and D. Al-Jumeily Obe, "Brain tumor and glioma grade classification using Gaussian convolutional neural network," *IEEE Access*, vol. 10, Article ID 29740, 2022.

[25] A. Belgrano, B. A. Malmgren, and O. Lindahl, "Application of artificial neural networks (ANN) to primary production time-series data," *Journal of Plankton Research*, vol. 23, no. 6, 2001.

[26] N. Rouf, M. B. Malik, T. Arif et al., "Stock market prediction using machine learning techniques: a decade survey on methodologies, recent developments, and future directions," *Electronics*, vol. 10, no. 21, p. 2717, 2021.

[27] Y. Liu, Z. Bao, Z. Zhang, D. Tang, and F. Xiong, "Information cascades prediction with attention neural network," *Human-centric Computing and Information Sciences*, vol. 10, no. 1, p. 13, 2020.

[28] A. Naseer, T. Yasir, A. Azhar, T. Shakeel, and K. Zafar, "Computer-aided brain tumor diagnosis: performance evaluation of deep learner CNN using augmented brain MRI," *International Journal of Biomedical Imaging*, vol. 202111 pages, Article ID 5513500, 2021.

[29] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, "A review on weight initialization strategies for neural networks," *Artificial Intelligence Review*, vol. 55, no. 1, pp. 291–322, 2021.

[30] "Weight Initialization in Neural Network, inspired by Andrew Ng | by Sadia Afrin | Medium," 2021, https://medium.com/@safrin1128/weight-initialization-in-neural-network-inspired-by-andrew-ng-e0066dc4a566.

[31] "Initializing neural networks - deeplearning.ai," 2021, https://www.deeplearning.ai/ai-notes/initialization/.

[32] M. Inthachot, V. Boonjing, and S. Intakosum, "Artificial neural network and genetic algorithm hybrid intelligence for predicting Thai stock price index trend," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 3045254, 8 pages, 2016.

[33] K. Om, S. Boukoros, A. Nugaliyadde et al., "Modelling email traffic workloads with RNN and LSTM models," *Human-centric Computing and Information Sciences*, vol. 10, no. 1, p. 39, 2020.

[34] S. Chopra, D. Yadav, and A. N. Chopra, "Artificial neural networks based Indian stock market price prediction: before and after demonetization," *int. J. Swarm intell. Evol. Comput*, vol. 8, no. 1, pp. 1–7, 2019.

[35] Z. Yudong and W. Lenan, "Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network," *Expert Systems with Applications*, vol. 36, no. 5, pp. 8849–8854, 2009.

[36] M. Bijari and M. Bijari, "An artificial neural network (p,d,q) model for timeseries forecastingq) model for timeseries forecasting," *Expert Systems with Applications*, vol. 37, no. 1, pp. 479–489, 2010.

[37] G. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[38] Y. Kara, M. A. Boyacioglu, and Ö. K. BaykanBaykan, "Predicting direction of stock price index movement using artificial neural networks and support vector machines: the sample of the Istanbul Stock Exchange," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5311–5319, 2011.

[39] E. Guresen, G. Kayakutlu, and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Systems with Applications*, vol. 38, no. 8, Article ID 10389, 2011.

[40] Y. Bing, J. K. Hao, and S. C. Zhang, "Stock market prediction using artificial neural networks," *Advanced Engineering Forum*, vol. 6–7, pp. 1055–1060, 2015.

[41] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques," *Expert Systems with Applications*, vol. 42, no. 1, pp. 259–268, 2015.

[42] J. Wang, J. Wang, W. Fang, and H. Niu, "Financial time series prediction using elman recurrent random neural networks," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 4742515, 14 pages, 2016.

[43] M. Hajirahimi and Z. Hajirahimi, "Performance evaluation of series and parallel strategies for financial time series forecasting," *Financial Innovation*, vol. 3, no. 1, p. 24, 2017.

[44] F. Zeren and A. Hizarchi, "the Impact Of covid-19 coronavirus on stock markets: evidence from selected countries," *Muhasebe ve finans i?ncelemeleri derg*, vol. 1, pp. 78–84, 2020.

[45] M. Aravind and C. G. Manojkrishnan, "COVID 19: effect on leading pharmaceutical stocks listed with NSE," *International Journal of Research in Pharmaceutical Sciences*, vol. 11, no. 1, pp. 31–36, 2020.

[46] D. Zhang, M. Hu, and Q. Ji, "Financial markets under the global pandemic of COVID-19," *Finance Research Letters*, vol. 36, Article ID 101528, 2020.

[47] J. W. Goodell, "COVID-19 and finance: agendas for future research," *Finance Research Letters*, vol. 35, Article ID 101512, 2020.

[48] S. R. Baker, N. Bloom, J. Davis, K. Kost, M. Sammon, and T. Viratyosin, "The unprecedented stock market reaction to Covid-19," *PANDEMICS LONG-RUN Efflatounia*, vol. 1, pp. 33–42, 2020.

[49] O. Bustos, A. Pomares, and E. Gonzalez, "A comparison between SVM and multilayer perceptron in predicting an emerging financial market: Colombian stock market," in *Proceedings of the 2017 Congreso Internacional de Innovacion y Tendencias en Ingenieria (CONIITI)*, pp. 1–6, Bogota, Colombia, February 2018.

[50] S. R. Das, D. Mishra, and M. Rout, "Stock market prediction using Firefly algorithm with evolutionary framework optimized feature reduction for OSELM method," *Expert Systems with Applications X*, vol. 4, Article ID 100016, 2019.

[51] J. Shafiq and M. O. Shafiq, "Short-term stock market price trend prediction using a comprehensive deep learning system," *Journal of Big Data*, vol. 7, no. 1, p. 66, 2020.

[52] F. Hao and D. S. Park, "CoNavigator: a framework of FCA-based novel coronavirus COVID-19 domain knowledge navigation," *Human-centric Computing and Information*, vol. 11, 2021.

[53] Y. Yang, F. Hao, D. S. Park, S. Peng, H. Lee, and M. Mao, "Modelling prevention and control strategies for COVID-19 propagation with patient contact networks," *Human-centric Computing and Information*, vol. 11, 2021.

[54] R. Tsaih, Y. Hsu, and C. C. Lai, "Forecasting S&P 500 stock index futures with a hybrid AI system," *Decision Support Systems*, vol. 23, no. 2, pp. 161–174, 1998.

[55] W. C. Navidi, *Statistics for engineers and scientists*, McGraw-Hill, NY, USA, 2015.

[56] J. Cook and J. A. Cook, "LASSO regression," *British Journal of Surgery*, vol. 105, no. 10, p. 1348, Sep. 2018.

[57] D. W. Snee and R. D. Snee, "Ridge regression in practice," *The American Statistician*, vol. 29, no. 1, pp. 3–20, 1975.
[58] Y. L. Pavlov, *Random forests*, vol. 45, pp. 5–32, 2001.
[59] A. Shabbir, M. Shabbir, A. R. Javed, M. Rizwan, C. Iwendi, and C. Chakraborty, "Exploratory data analysis, classification, comparative analysis, case severity detection, and internet of things in COVID-19 telemonitoring for smart hospitals," *Journal of Experimental & Theoretical Artificial Intelligence*, pp. 1–28, 2022.
[60] N. Bühlmann and B. Peter, "Stability selection," *Journal of the Royal Statistical Society: Series B*, vol. 72, no. 4, pp. 417–473, 2010.
[61] L. D. Persio and O. Honchar, "Artificial Neural Networks architectures for stock price prediction: Comparisons and Applications," *International journal of circuits, systems and signal processing*, vol. 10, 2016.
[62] A. F. Aleroud, I. M. Alsmadi, A. I. Alaiad, and Q. A. A. Radaideh, "The effects of neural networks training factors on stock price prediction errors," in *Proceedings of the International Conference on Communication, Internet, and Information Technology*, Baltimore, USA, August 2014.
[63] M. L. Thormann, J. Farchmin, C. Weisser, R. M. Kruse, B. Säfken, and A. Silbersdorff, "Stock price predictions with LSTM neural networks and twitter sentiment," *Statistics, Optimization & Information Computing*, vol. 9, no. 2, pp. 268–287, 2021.
[64] A. Geron, *Hands on machine learning with scikit learn keras and tensorflow concepts tools and techniques to build intelligent systems*, O'Reilly Media, Inc, sebastopol, 2019.
[65] C. Bashir, A. Peshkar, R. Sujatha et al., "COVID-19 patient health prediction using boosted random forest algorithm," *Frontiers in Public Health*, vol. 8, p. 357, 2020.
[66] A. Srivastava and B. N. SomvanshiMishra, "Reconstruction and visualization of carbohydrate, N-glycosylation pathways in Pichia pastoris CBS7435 using computational and system biology approaches," *Systems and synthetic biology*, vol. 7, no. 1-2, pp. 7–22, 2013.
[67] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of machine learning research*, vol. 13, pp. 281–305, 2012.
[68] A. C. Andonie and R. Andonie, "Weighted random search for hyperparameter optimization," *International Journal of Computers, Communications & Control*, vol. 14, no. 2, pp. 154–169, 2019.
[69] M. Kotila, "Talos Hyperparameter Tuning for Keras," 2021, https://pypi.org/project/talos/.
[70] A. Jierula, S. Wang, T. Oh, and P. Wang, "study on accuracy metrics for evaluating the predictions of damage locations in deep piles using artificial neural networks with acoustic emission data," *applied sciences*, vol. 11, p. 2314, 2021.