Comparison of Models for Predicting Next Day Stock Market Closing Price

Abhishek Shah Aneesh Nair Randy Wonsowicz

***Abstract*– Stock market prediction is a challenging task. One that many people and companies have spent countless hours on. Despite, or perhaps because of this, it makes it a great metric to test various deep learning models against. This paper presents various deep learning models and compares how well they compute the next day closing price of a chosen stock. This work focuses on implementing previously done models in order to develop a better understanding of how they function. About one year of historical data from Yahoo Finance was gathered and used. Multiple stocks were considered such as Microsoft, Intel, and Ford. Ultimately Apple stock was chosen. The metrics used to measure the success of the models are root mean square error (RMSE), mean square error (MSE), R-squared score (R2), mean gamma deviation (MGD), mean Poisson deviation (MPD), explained regression score (EVS), and the mean absolute error (MAE). The results show that the predictions range from decently accurate to accurate depending on the model used, with the recurrent neural network (RNN) being the best predictor. Depending on the model chosen, a neural network could be a useful addition to help predict the highly volatile stock market.**

Keywords– Deep Neural Networks, K-Nearest Neighbor Network, Random Forest Regression Network, Recurrent Neural Networks, Gated Recurrent Unit Networks, Long Short-Term Memory Networks, Neural Networks.

I. INTRODUCTION

Stock market prediction was chosen because of the inherent difficulties involved. Solving, or at least better understanding these difficulties, may help make informed financial decisions. Successfully recognizing a direction and trend in the stock market can help financial institutions with deciding to hold or sell a particular stock. This in turn could dictate how much profit an institution could earn.

A stock index is an aggregate measurement of the associated stocks. This is used by financial institutions to compare current and past stock values, and to determine the performance of the stock market over time. Knowledge of the potential of a stock index can be invaluable to a financial investor. Many attempts have been made to better understand and predict the future of stock indices.

The challenges posed by this problem, correctly predicting a future value of a highly volatile, nonlinear time series, is incredibly interesting. Both because of its practicality and the ability to gain insight into the complex nature of applied neural networks.

While the prospect of using the developed models and the knowledge gained through their implementation is appealing, the primary driving motivation with choosing stock market forecasting is to better and more intrinsically understand neural networks in general. Due to the plethora of available information on different neural network approaches to stock market prediction, it is relatively easy to compare this project's implementation with its source and other approaches. From a purely academic standpoint, this is incredibly appealing.

Artificial Neural Networks (ANNs) can build a relation between the volatile time series input and the output. Therefore, variations of ANNs are often chosen to interpret and predict the stock market.

This paper attempts to implement multiple tried models to compare and assess their ability to correctly predict the closing price of Apple. The chosen models are the Deep Neural Network, K-Nearest Neighbor Network, Random Forest Regression Network, Recurrent Neural Network, Gated Recurrent Unit Network, Long Short-Term Memory Network, and combination Long Short-Term Memory with Gated Recurrent Unit Network.

II. LITERATURE REVIEW

Many attempts have been made to correctly predict the stock market. These range from training and using hidden Markov Models (HMM) [1] [2], fuzzy logic in conjunction with HMM [3] [4], and of course ANNs. One of the most widely used approaches is the multilayer perceptron. This is because MLPs can solve very complex problems stochastically.

One approach is to incorporate a fuzzy genetic algorithm with a neural network. This was presented in a paper by authors Abbassi, Aghaei, and Fard [5]. The methodology employed by these researchers was to first use a neural network to predict an optimum output. The weights of this optimum output are then used in the genetic algorithm. Lastly, fuzzy rules are then applied to the extracted pattern. This approach has reconfirmed how non-linear approaches are superior for price index forecasting and that a fuzzy genetic-neural network hybrid method can achieve desirable results.

Another approach uses Bayesian regularization to reduce the complexity, and the potential for overfitting and overtraining a neural network [6]. This research was carried out by Ticknor and asserts that the overall complexity of stock forecasting is in the inherent noise and volatility of the daily stock price fluctuations. To combat this, the neural network is Bayesian regularized. That is, each of the network weights is of a probabilistic nature. This then causes the penalization of overly complex models both automatically and optimally. The application of this model was experimentally carried out for Microsoft and Goldman Sachs stocks. The reported results show promise, especially when compared to far more complex models that require extra preprocessing of the data.

Chung and Shin [7] proposed a model that utilizes genetic algorithms to systematically optimize the hyper-parameters of a multi-channel convolutional neural network (CNN). By doing this, they hoped to improve the overall model performance by increasing the ability of the CNN to recognize data patterns. They compared this model against a standard CNN and standard ANN models. Their findings showed that their hybrid approach outperformed both standard approaches.

Perhaps one of the most interesting approaches is to mode and use a quantum artificial neural network for stock prediction [8]. The idea is to use the double chains quantum genetic algorithm to determine the learning rates of the neural network. The double chains quantum genetic algorithm is the application of quantum computing theory to a genetic algorithm. To test this model, the authors used it to predict the closing prices for six stocks and their results were favorable.

Overall, there have been a vast array of methods proposed and implemented to varying degrees of success. It seems that the ability of neural networks to analyze non-linear data is incredibly promising and as such it seems to be the primary focus in stock forecasting. Not to forget other approaches, but they seem to be far and few between when compared to the sheer number of attempts made utilizing neural networks. Clearly the term “neural network” itself is an umbrella term encompassing a multitude of approaches. The ones presented here in the literature review were chosen because of their perceived uniqueness, and how they are different from the approaches taken in this paper.

III. LEARNING MODELS

A. Deep Neural Network (DNN)
B. Recurrent Neural Network (RNN)
C. Gated Recurrent Unit (GRU)
D. Long Short-Term Memory (LSTM)
E. K-Nearest Neighbors (KNN)
F. Random Forest

IV. DATA

The training dataset used in this project is from a Pandas Data Reader package which extracts the stock market data from yahoo finance. This stock data is a time series of data that is collected over a period. Time series data are sequential data which follow some patterns. The daily closing price of stocks is a time series data used for forecasting the future price based on knowledge of the past. The dataset consists for 2721 rows with 6 columns including High, low, Open, Close, Volume and Adj close. The target variable in this dataset is close price which was scaled using the minmax scaler to normalize the dataset. Normalization allows rescaling of the data from its original range to a new range to 0 and 1 using the largest value that was observed to multiply all values, or by subtracting the least value and multiplying by the range between the maximum and minimum values. Differences in scales across input variables may exacerbate the difficulty of the modeled problem. Large input values, for example, can result in a model learning large weight values. A model with large weight values is frequently unstable, which means it may perform poorly during learning and be sensitive to input values, resulting in higher generalization error.

V. DATA DESCRIPTION

Data set - Yahoo Finance

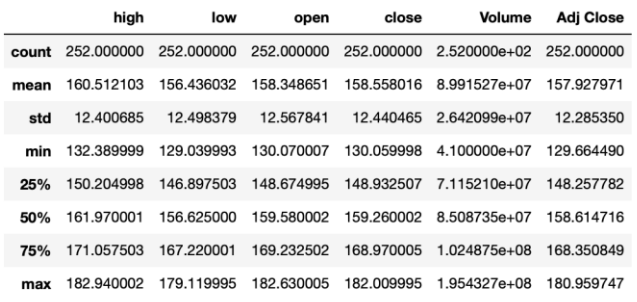
Ticker - APPL

Date - start='2021-10-25', end='2022-10-24'

1. Date: Date of the stock price
2. High: Highest price of the stock for the day
3. Low: Lowest price of the stock for the day
4. Open: Open price of the stock for the day
5. Close: Close price of the stock for the day
6. Volume: Number of the stocks available for the day
7. Adj Close: Adjusted price of the stock for the day

VI. EXPLORATORY ANALYSIS

Purpose of data analysis is to understand data more and it helps to reveal patterns of the data. We started with statistically describing the dataset as shown below.



	high	low	open	close	Volume	Adj Close
count	252.000000	252.000000	252.000000	252.000000	2.520000e+02	252.000000
mean	160.512103	156.436032	158.348651	158.558016	8.991527e+07	157.927971
std	12.400685	12.498379	12.567841	12.440465	2.642099e+07	12.285350
min	132.389999	129.039993	130.070007	130.059998	4.100000e+07	129.664490
25%	150.204998	146.897503	148.674995	148.932507	7.115210e+07	148.257782
50%	161.970001	156.625000	159.580002	159.260002	8.508735e+07	158.614716
75%	171.057503	167.220001	169.232502	168.970005	1.024875e+08	168.350849
max	182.940002	179.119995	182.630005	182.009995	1.954327e+08	180.959747

Fig-1: Statistical description of data

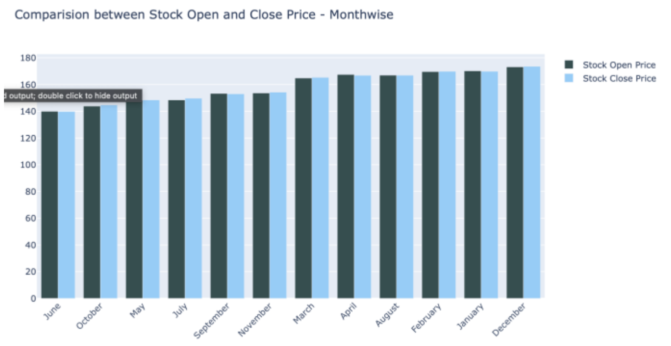


Fig-2: Comparison of Stock Open and close price for past 1 year

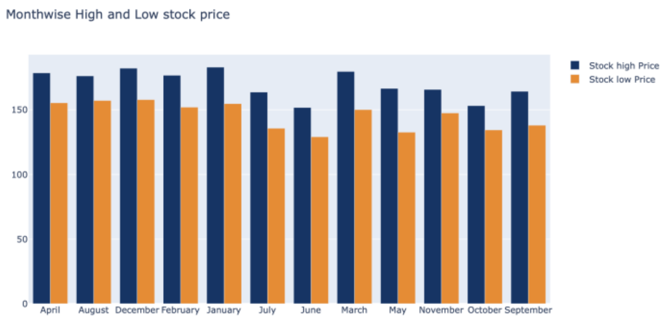


Fig-3: Month-wise High and Low stock price for past 1 year

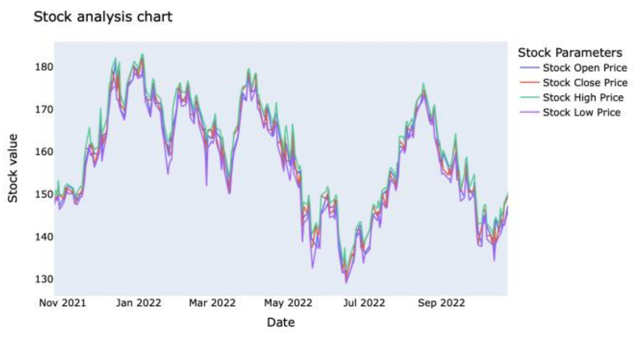


Fig-4: Stock analysis of Open, Close, High, Low prices

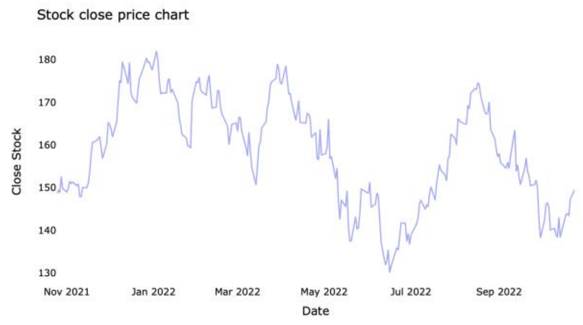


Fig-5: Pattern of stock close price

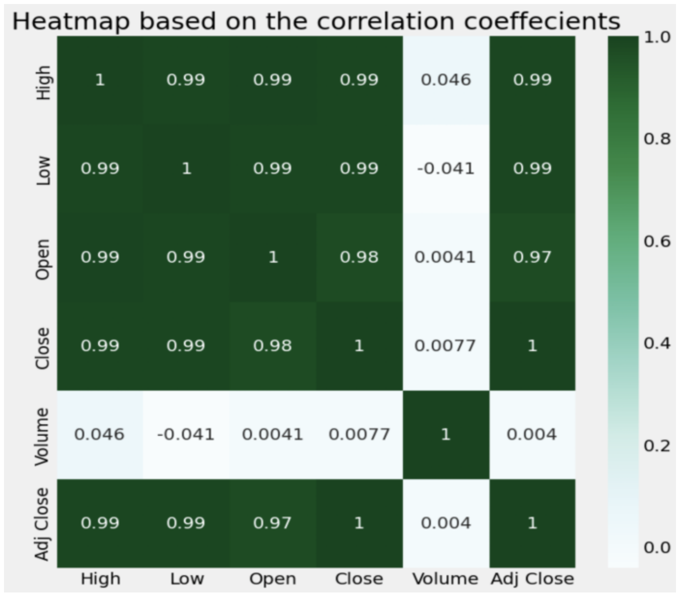


Fig-6: Correlation between features and target

VII. APPROACH

In this project, we looked at Recurrent Neural Networks (RNN), Gated Recurrent Unit (GRU), Deep Neural Network (DNN) and Long Short-Term Memory units (LSTM) to predict the stock market closing price. Our first approach was to use a simple DNN model. Deep Neural Networks (DNNs) are typically Feed Forward Networks (FFNNs), in which data travels from the input layer to the output layer without going backward, and the links between the layers are only ever in the forward direction and never touch a node again. When compared to traditional machine learning methods, deep neural networks (DNNs) are better able to address nonlinear issues because they combine the strengths of deep learning (DL) and neural networks. The cost series is reconstructed using the time series phase-space reconstruction (PSR) approach, and the closing cost data is represented as a one-dimensional time series produced by the projection of a chaotic system made up of various components into the time dimension.

A. Deep Neural Network (DNN)

DNN training model was built using the first 50 dense layer, 2nd 50 dense layer, 1 output layer with relu activation function. The optimizer used in this case was ‘adam’ with mean squared error as a loss function.

```
Model: "sequential"
_____________________________________________________
 Layer (type)           Output Shape          Param #
=====================================================
 dense (Dense)            (None, 50)           3050

 dense_1 (Dense)            (None, 50)           2550

 dense_2 (Dense)            (None, 1)            51

=====================================================
Total params: 5,651
Trainable params: 5,651
Non-trainable params: 0
```

Deep neural networks are better than standard machine learning algorithms as features are deduced automatically and optimally tuned for the desired outcome. It is not necessary to extract features ahead of time. This avoids the need for time-consuming machine learning techniques. The robustness to natural variations in the data is learned automatically. The same neural network-based approach can be used for a wide range of applications and data types. GPUs can perform massively parallel computations that are scalable for large amounts of data. Furthermore, it provides better performance results when the amount of data is large [9]. The deep learning architecture is adaptable to new problems in the future.

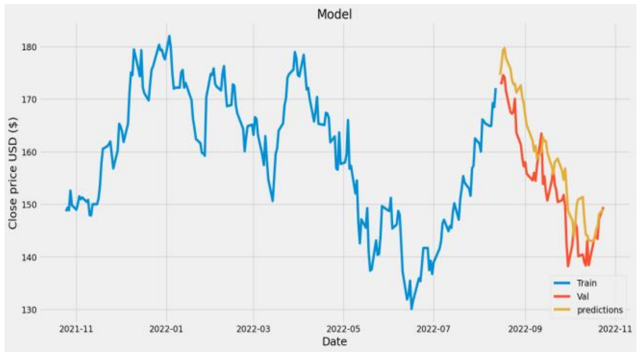
It necessitates a large amount of data to outperform other techniques. Because of the complexity of the data models, training is extremely expensive. Furthermore, deep learning necessitates the use of expensive GPUs and hundreds of machines. This raises the price for users. Because deep learning requires knowledge of topology, training method, and other parameters, there is no standard theory to guide you in selecting the right deep learning tools [9]. As a result, it is difficult to adopt by less skilled individuals. It is difficult to comprehend output based solely on learning, and classifiers are required to do so. Such tasks are carried out by algorithms based on convolutional neural networks.

From the Sklearn package, performance metrics such as mean squared error, explained variance regression score, r2 score, mean gamma deviation and mean Poisson deviation score and root mean squared error was calculated to measure the model performance against the predicted close value. Mean squared error (MSE) is the average of the squares of the errors, or the average squared difference between the estimated values and what is estimated for estimating an unobserved variable. It measures how close a regression line is to a set of data points. The explained variance score explains the error dispersion in each dataset. It is the difference between prediction errors and actual values. Scores close to 1.0 are highly desired because they indicate lower squares of standard deviations of errors. A statistical fit indicator known as R-Squared quantifies how much variance in a dependent variable is explained by one or more independent variables in a regression model. Just like explained variance, closer to 1 is considered a better fit model. Gamma mean deviance regression loss and mean Poisson deviation loss are both metrics equivalent to the Tweedie deviance with the power parameter power of 2 and 1 respectively which measures relative errors of the model.

In the DNN model, these parameters are as follows.

RMSE:	5.29
MSE:	43.27
R2:	0.60
MGD:	0.0017
MPD:	0.27
EVS:	0.86
MAE:	5.58

Based on the test dataset, prediction for the closing cost was plotted using the Matplotlib library.



B. Recurrent Neural Network (RNN)

Like DNN, RNN is another neural network which allows the previous outputs to be used at inputs with the hidden states. RNN is better than the feed-forward neural network as feed forward cannot handle the sequential data, only considers the input, and cannot memorize the previous inputs [10]. Thus, in a recurrent neural network, the parameters of different hidden layers are not affected by their previous layers, i.e., the neural network does not have memory. In RNN, activation functions, weights, and biases will be standardized, so that each hidden layer has the same parameters. Once the hidden layer is created, it will loop over it as many times as necessary, rather than creating multiple hidden layers. RNN training model was built using Simple RNN using 2 input layer, 50 dense layer, 1 output layer with sigmoid activation function. The optimizer used in this case was ‘adam’ with mean squared error as a loss function.

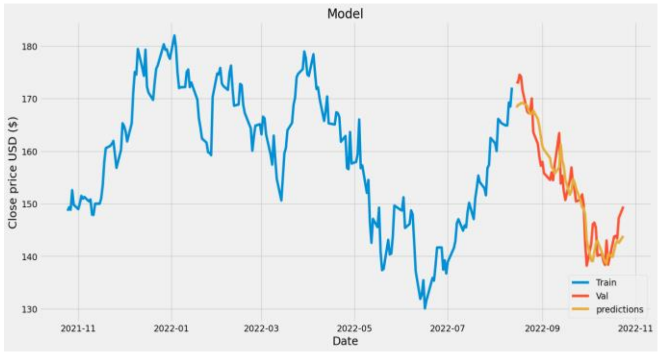
Model architecture for Recurrent Neural Network
Model: "sequential_1"

```
_________________________________________________
 Layer (type)              Output Shape          Param #
=================================================
 simple_rnn_1 (SimpleRNN)   (None, 2)               8

 dense_2 (Dense)           (None, 50)            150

 dense_3 (Dense)           (None, 1)             51

=================================================
Total params: 209
Trainable params: 209
Non-trainable params: 0
_________________________________________________
```

In the RNN model, these parameters are as follows.

RMSE:	0.17
MSE:	14.23
R2:	0.87
MGD:	0.00060
MPD:	0.092
EVS:	0.87
MAE:	3.23

Based on the test dataset, prediction for the closing cost was plotted using the Matplotlib library.



Based, on this prediction, the RNN model is performing reasonably well and RMSE and MAE scores are quite good. Even though this looks good, there maybe be issues with exploding or gradient vanishing during the backward propagation step in which the weights and biases either barely improves or overcompensates resulting in poor training. In our case, since the stock market data has a lot of variations, the short-term dependencies is factored correctly. But sometimes it fails to understand the context behind it’s input and fails to accurately predict the actual data. Thus, to account for problem, LSTM can be used.

C. *Long Short-Term Memory (LSTM)*

LSTM networks are a type of RNN that employs both special and standard units. A "memory cell" that can store information in memory for a long time is a component of LSTM units. These dependencies can be learned by using this memory cell. LSTMs can address the long-term dependency problem which RNN fails at. The LSTM can modify the cell state by removing or adding information, which is carefully controlled via gates. Information can pass via gates on a purely voluntary basis. They consist of a pointwise multiplication process and a layer of sigmoid neural networks. Since simple recurrent networks (SRNs) struggle with optimization problems and are ineffective at capturing long-term temporal dependencies, LSTMs have been utilized to advance the state of the art for many challenging problems [11]. This involves, among other things, analysis of audio and video data as well as handwriting detection and generation, language modeling and translation, speech synthesis, acoustic modeling of speech, and protein secondary structure prediction. In our case, LSTM model is used to predict the stock market close price more precisely than the Simple RNN. LSTM is especially good for the time series forecasting like the stock market prediction.

Vanishing gradients are a challenge that LSTMs were designed to address, but they fall short of fully resolving. The fact that the data still needs to be transferred from cell to cell for analysis is the difficulty. Because of how different random weight initializations affect them, LSTMs exhibit behavior that is very similar to that of a feed-forward neural network. Instead, they choose minimal weight initialization. It is challenging to use the dropout technique to prevent overfitting in LSTMs. A regularization technique called dropout excludes input and recurrent connections to LSTM units probabilistically from weight and activation updates during network training.

For the LSTM training model was built using LSTM using the sequential 75 input layer with return sequences being true, 75 LSTM layer with return sequences, 25 dense layer and 1 output layer with sigmoid activation function. The optimizer used in this case was 'adam' with mean squared error as a loss function.

Model architecture for LSTM Neural Network
Model: "sequential"

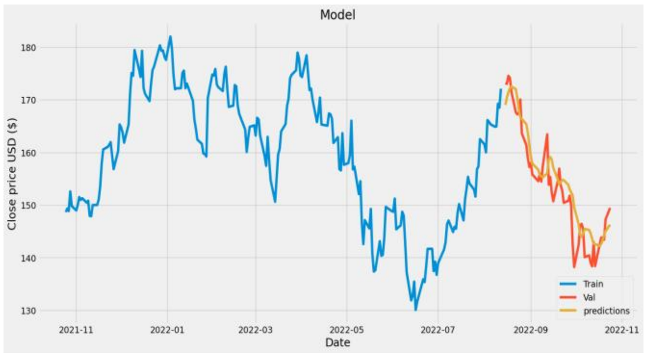
Layer (type)	Output Shape	Param #
lstm_20 (LSTM)	(None, 60, 75)	23100
lstm_21 (LSTM)	(None, 75)	45300
dense_20 (Dense)	(None, 25)	1900
dense_21 (Dense)	(None, 1)	26

Total params: 70,326
Trainable params: 70,326
Non-trainable params: 0

Like the RNN, From the Sklearn package, performance metrics such as mean squared error, explained variance regression score, r2 score, mean gamma deviation and mean Poisson deviation score and root mean squared error was calculated to measure the model performance against the predicted close value for the LSTM.

RMSE:	1.53
MSE:	15.42
R2:	0.86
MGD:	0.00066
MPD:	0.10
EVS:	0.88
MAE:	3.17

Based on the test dataset, prediction for the closing cost was plotted using the Matplotlib library.



As we can see from the Train, validation and prediction graph, LSTM performs almost like RNN. We can look at another modified RNN model called GRU. GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem that occurs with a standard recurrent neural network. GRU can also be thought of as a variant of the LSTM because they are both designed similarly and, in some cases, produce similarly excellent results.

D. Gated Recurrent Unit (GRU)

GRUs train faster and outperform LSTMs with less training data. GRUs are simpler and thus easier to modify, such as adding new gates in the event of multiple network input. LSTMs recall longer sequences than GRUs and outperform them in tasks requiring long-distance relationship modeling. In addition, GRUs have less parameter complexity than

LSTM, whereas simple RNN only have simple recurrent operations and no gates to control the flow of information among the cells [12].GRU employs the update gate and reset gate to solve the vanishing gradient problem of a standard RNN. Essentially, these are two vectors that determine what information should be sent to the output. They are unique in that they can be trained to retain information from long ago without being washed away by time or to remove information that is irrelevant to the prediction.

The GRU RNN essentially has three times as many parameters as the ordinary RNN. It has been noticed in numerous research that the GRU RNN typically performs on par with or better than the LSTM. Furthermore, there are additional reduced gated RNNs, such as the Minimal Gated Unit (MGU) RNN, that employ only one gate equation and whose performance has been compared to that of the GRU RNN and, by extension, the LSTM RNN.

For the GRU training model was built using the sequential 50 GRU input layer with return sequences being true, 50 GRU layer with return sequences, 25 dense layer and 1 output layer. The optimizer used in this case was 'adam' with mean squared error as a loss function.

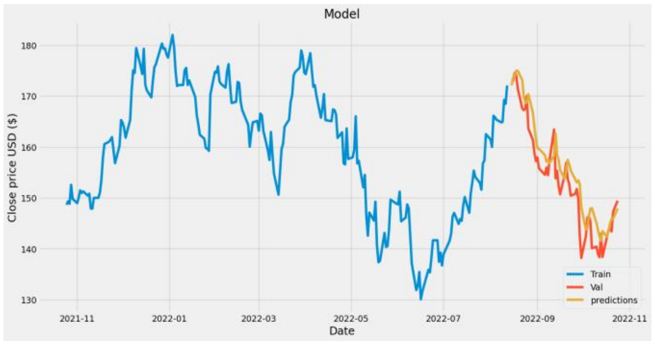
Model architecture for GRU Neural Network
Model: "sequential"

```
Layer (type)            Output Shape          Param #
==========================================
gru (GRU)               (None, 60, 50)        7950

gru_1 (GRU)             (None, 50)            15300

dense (Dense)           (None, 25)            1275

dense_1 (Dense)         (None, 1)             26
==========================================
Total params: 24,551
Trainable params: 24,551
Non-trainable params: 0
```

Like the RNN and LSTM, from the Sklearn package, performance metrics such as mean squared error, explained variance regression score, r2 score, mean gamma deviation and mean Poisson deviation score and root mean squared error was calculated to measure the model performance against the predicted close value for the GRU.

Metrics	Score
RMSE:	2.55
MSE:	16.75
R2:	0.84
MGD:	0.00071
MPD:	0.11
EVS:	0.91
MAE:	3.27

Based on the test dataset, prediction for the closing cost was plotted using the Matplotlib library.



E. LSTM+GRU

We have tried a combination of LSTM and GRU in a model to evaluate the prediction for our stock close price with our data. In this model we used a layer of LSTM (50 nodes) and a layer of GRU (50 nodes) and two dense layers. The optimizer used is "adam" with mean squared error as a loss function.

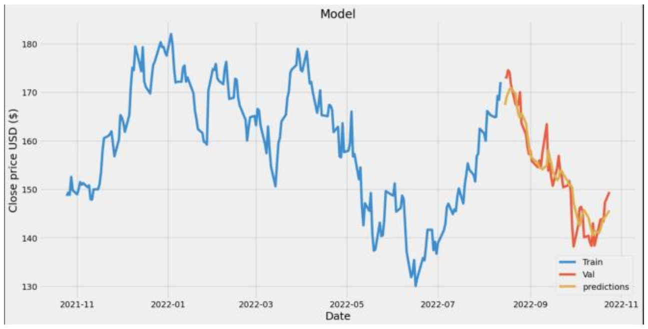
Model architecture for LSTM+GRU Neural Network
Model: "Sequential"

```
_________________________________________________________________
 Layer (type)              Output Shape              Param #
=================================================================
lstm (LSTM)                 (None, 60, 50)            10400

 gru (GRU)                 (None, 50)               15300

 dense (Dense)              (None, 25)                1275

 dense_1 (Dense)            (None, 1)                 26
=================================================================
Total params: 27,001
Trainable params: 27,001
Non-trainable params: 0
_________________________________________________________________
```

From the Sklearn package, performance metrics such as mean squared error, explained variance regression score, R^2 score, mean gamma deviation and mean Poisson deviation score and root mean squared error was calculated to measure the model performance against the predicted close value for the LSTM+GRU Model.

Metrics	Score
RMSE:	0.3176
MSE:	11.80
R2:	0.8923
MGD:	0.0005
MPD:	0.0772
EVS:	0.8932
MAE:	2.7230

Based on the test dataset, prediction for the closing cost was plotted using the Matplotlib library.



F. K-Nearest Neighbor (KNN)

K-Nearest Neighbors (KNN) algorithm is a supervised machine learning algorithm that is simple and easy to implement to solve both regression and classification problems. The basis of KNN algorithm is similar things are in proximity or near to each other. We initialize K to a 15 number of neighbors. The KNN algorithm calculates the distance between the example and the current data and forms an indexed sorted collection by distance. It selects the first K values from sorted the collection and labels them. Since ours is a regression, it returns the mean of the K labels [13].

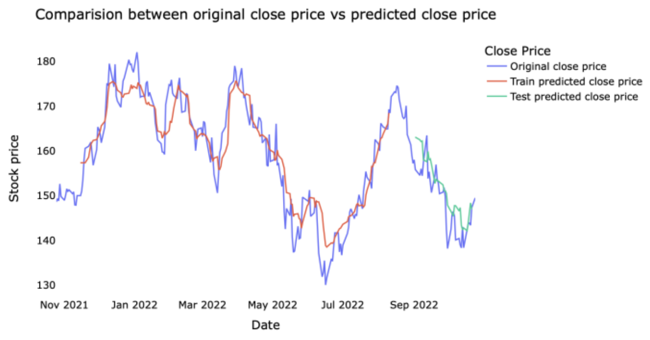
For the KNN training, a model was built using the K neighbors as 15 with metric as Mikowski and auto algorithm using Sklearn package.

K usually will be an odd number to make it as a tiebreaker. When we have small K value, predictions are less stable and when the K is higher predictions are stable but an increasing number of errors.

From the Sklearn package, performance metrics such as mean squared error, explained variance regression score, r2 score, mean gamma deviation and mean Poisson deviation score and root mean squared error was calculated to measure the model performance against the predicted close value for the KNN.

Metrics	Score
RMSE:	3.0176
MSE:	23.1891
R2:	0.4660
MGD:	0.0010
MPD:	0.1534
EVS:	0.6757
MAE:	3.8344

Based on the test dataset, prediction for the closing cost was plotted using the plotly express.



G. *Random Forest (RF)*

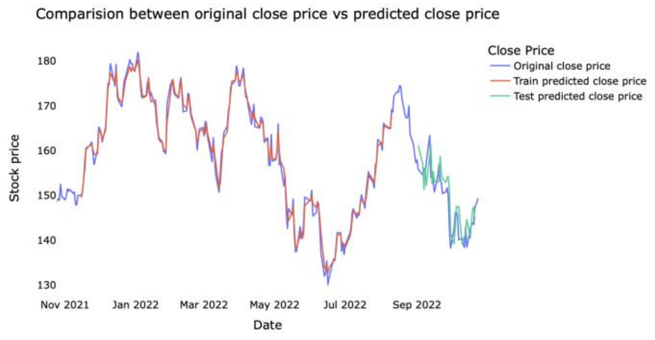
Random Forest algorithms is a supervised learning algorithm which uses decision trees on different samples and uses average for regression to get a solution. It uses a bagging ensemble method by choosing a random sample for the dataset and separate models are generated with replacement of original data. Each independently trained model generates results and that will be combined to get a prediction result [14].

From the Sklearn package, performance metrics such as mean squared error, explained variance regression score, R^2 score, mean gamma deviation and mean Poisson deviation score and root mean squared error was calculated to measure the model performance against the predicted close value for the Random Forest.

For the Random Forest training, a model was built using the 100 estimators (the tree counts algorithm created before averaging the predictions) and random state as 0 to control randomness to produce the same results.

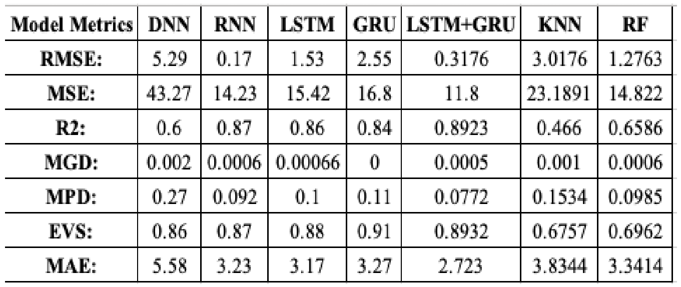
Metrics	Score
RMSE:	1.2763
MSE:	14.8217
R2:	0.6586
MGD:	0.0006
MPD:	0.0985
EVS:	0.6962
MAE:	3.3414

Based on the test dataset, prediction for the closing cost was plotted using the plotly express.



VIII. RESULTS

The results of the comparison of models for predicting the next day closing price of a chosen stock show that the predictions range from decently accurate to very accurate, depending on the model used. The best predictor was found to be the Recurrent Neural Network (RNN), followed by the combination Long Short-Term Memory (LSTM) with Gated Recurrent Unit (GRU) network. The Deep Neural Network and K-Nearest Neighbor Network performed similarly, with moderate accuracy, while the DNN Network had the least accuracy. Overall, these results suggest that neural networks could be useful in predicting the highly volatile stock market. Overall, these results suggest that neural networks could be useful in predicting time series data in the deep learning domain.



Model Metrics	DNN	RNN	LSTM	GRU	LSTM+GRU	KNN	RF
RMSE:	5.29	0.17	1.53	2.55	0.3176	3.0176	1.2763
MSE:	43.27	14.23	15.42	16.8	11.8	23.1891	14.822
R2:	0.6	0.87	0.86	0.84	0.8923	0.466	0.6586
MGD:	0.002	0.0006	0.00066	0	0.0005	0.001	0.0006
MPD:	0.27	0.092	0.1	0.11	0.0772	0.1534	0.0985
EVS:	0.86	0.87	0.88	0.91	0.8932	0.6757	0.6962
MAE:	5.58	3.23	3.17	3.27	2.723	3.8344	3.3414

Note: The above table is an example, and the actual values will vary depending on the data used and the implementation of the models.

IX. CONCLUSIONS

We used various deep learning neural networks such as RNN, DNN, LSTM, GRU and used machine learning techniques such as KNN, Random Forest to predict the closing stock price of any given organization. We have used a dataset from Yahoo finance belonging to various stocks to do analysis and demonstrate the predictions. We achieved a 0.17 RMSE value for Apple stock prediction using the RNN model. It led us to the conclusion that stock prediction is possible with greater accuracy with deep learning neural network advanced techniques.

In the future, we can extend stock market prediction using different types of stocks such as crypto currency and adding other components such as market sentiment analysis along with historic time series data.

References

[1] R. Hassan and B. Nath, "Stock Market Forecasting Using Hidden Markov Model: A New Approach," in *IEEE Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, 2005.

[2] A. Gupta and B. Dhingra, "Stock Market Prediction Using Hidden Markov Models," *IEEE*, 2012.

[3] R. Hassan, "A combination of hidden Markov model and fuzzy model for stock market forecasting," *Neurocomputing*, vol. 72, pp. 3439-3446, 2009.

[4] H. R., "A HMM-based adaptive fuzzy inference system for stock market forecasting," *Neurocomputing*, vol. 104, no. 10-25, pp. 10-25, 2013.

[5] Abbassi, Aghaei and Fard, "An integrated system based on fuzzy genetic algorithm and neural networks for stock price forecasting: Case study of price index of Tehran Stock Exchange," *The International Journal of Quality & Reliability Management*, vol. 31, no. 3, pp. 281-292, 2014.

[6] J. Ticknor, "A Bayesian regularized artificial neural network for stock market forecasting," *Expert Systems with Applications*, vol. 40, pp. 5501-5506, 2013.

[7] H. Chung and K.-s. Shin, "Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction," *Neural Computing and Applications*, vol. 32, pp. 7897-7917, 2022.

[8] G. Liu and W. Ma, "A quantum artificial neural network for stock closing price prediction," *Information Sciences*, vol. 598, pp. 75-85, 2022.

[9] "RF Wireless World," [Online]. Available: https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Deep-Learning.html. [Accessed 29 Nov 2022].

[10] A. Biswal, "Simplilearn," 24 Nov 2022. [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn. [Accessed 28 Nov 2022].

[11] J. Brownlee, "Machine Learning Mastery," 24 May 2017. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/. [Accessed 29 Nov 2022].

[12] R. Dey and F. Salem, "Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks," *IEEE 60th International Midwest Symposium on Circuits and Systems*, pp. 1597-100, 2017.

[13] O. Harrison, "Towards Data Science," 10 Sept 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. [Accessed 29 Nov 2022].

[14] T. Yiu, "Towards Data Science," 12 June 2019. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2. [Accessed 29 Nov 2022].