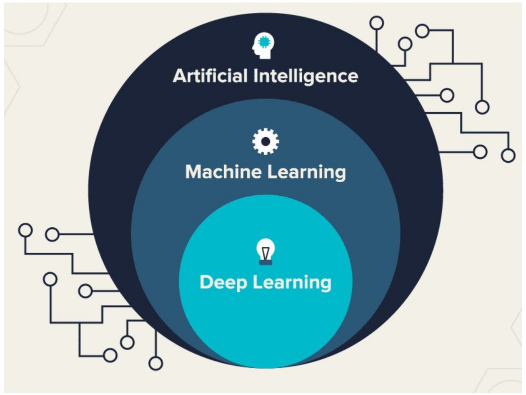
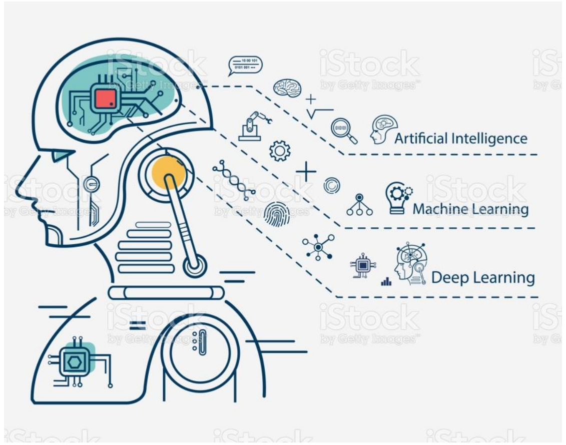
Stock Market Forecasting

Presentation By: Dhruv Kumar

What is Artificial Intelligence?

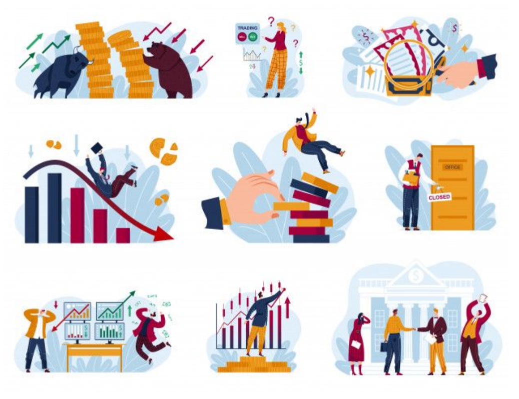




Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

The ideal characteristic of artificial intelligence is its ability to rationalize and take actions that have the best chance of achieving a specific goal. A subset of artificial intelligence is machine learning, which refers to the concept that computer programs can automatically learn from and adapt to new data without being assisted by humans. Deep learning techniques enable this automatic learning through the absorption of huge amounts of unstructured data such as text, images, or video.

What is Stock Market ?



A stock market, equity market or share market is the aggregation of buyers and sellers of stocks (also called shares), which represent ownership claims on businesses; these may include securities listed on a public stock exchange, as well as stock that is only traded privately, such as shares of private companies which are sold to investors through equity crowdfunding platforms. Investment in the stock market is most often done via stock brokerages and electronic trading platforms. Investment is usually made with an investment strategy in mind.
The concept of Stocks/Shares usually helps a company/enterprise procure capital, which it could use for increasing production (or) for setting up an R&D etc, stocks therefore helps a company to work in the long run.

What is Stock Market Prediction/Forecasting ?

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. Usually a successful analyst would predict a stock by analyzing the trends of stock prices over a period of time, he could buy certain stocks/shares at one point of time and sell at another suitable point of time. Successful prediction and Forecasting can often yield a lot of profit for the person.

There are three conventional approaches for stock price prediction: technical analysis, traditional time series forecasting, and machine learning method. Artificial Neural Networks (ANN) are widely used for the prediction of stock price movements. Every algorithm has its way of learning patterns and then predicting.

Artificial Neural Network (ANN) is a popular and more recent method which also incorporates technical analysis for making predictions in financial markets. ANN includes a set of threshold functions. These functions are trained on historical data after connecting each other with adaptive weights and they are used to make future predictions.

Artificial Neural Networks:

Artificial neural networks (ANNs) are computing systems inspired by the biological neural networks that constitute animal brains.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

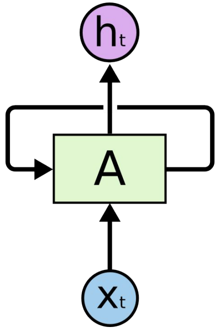
Problems with ANNs

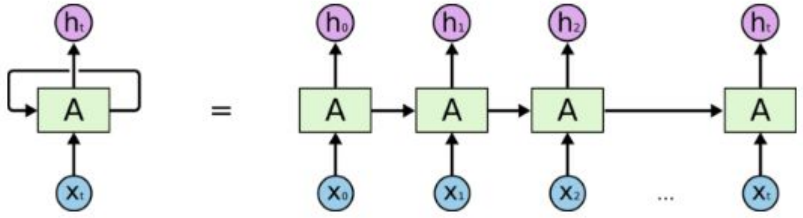
Humans don’t start their thinking from scratch every second. While reading this, we understand each word based on our understanding of previous words. We don’t throw everything away and start thinking from scratch again. Our thoughts have persistence.

Traditional neural networks can’t do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It’s unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.

Recurrent Neural Networks





An unrolled recurrent neural network.

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (nlp), stock forecasting etc.

They are distinguished by their "memory" as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence.

Problems with RNNs

Vanishing Gradient Problem:

RNNs suffer from the problem of vanishing gradients, which hampers learning of long data sequences. The gradients carry information used in the RNN parameter update and when the gradient becomes smaller and smaller, the parameter updates become insignificant which means no real learning is done.

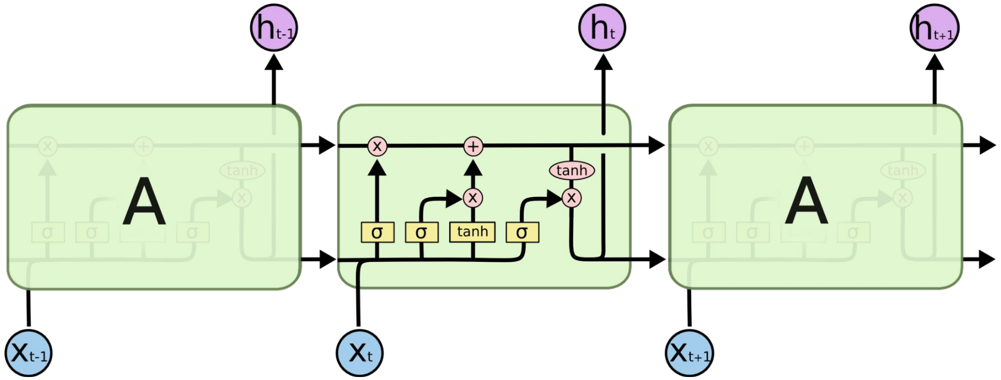
Long Term Dependencies:

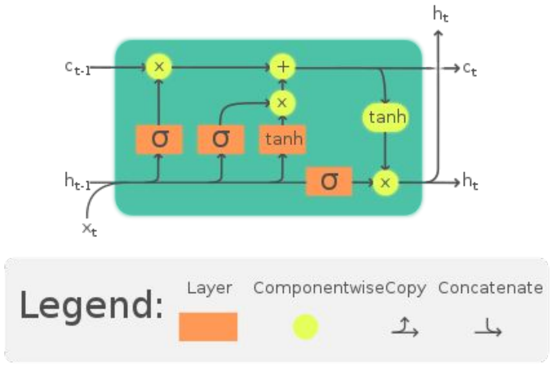
RNNs are able to predict the present task based on the previous information. However it only does so when the gap between the relevant information and where it’s needed is small. It’s entirely possible for the gap between the relevant information and the point where it is needed to become very large. Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

Exploding Gradient Problem

Exploding gradients is a problem in which the gradient value becomes very big and this often occurs when we initialize larger weights and we could end up with NaN. If our model suffered from this issue we cannot update the weights at all.

LSTM (Long Short Term Memory)

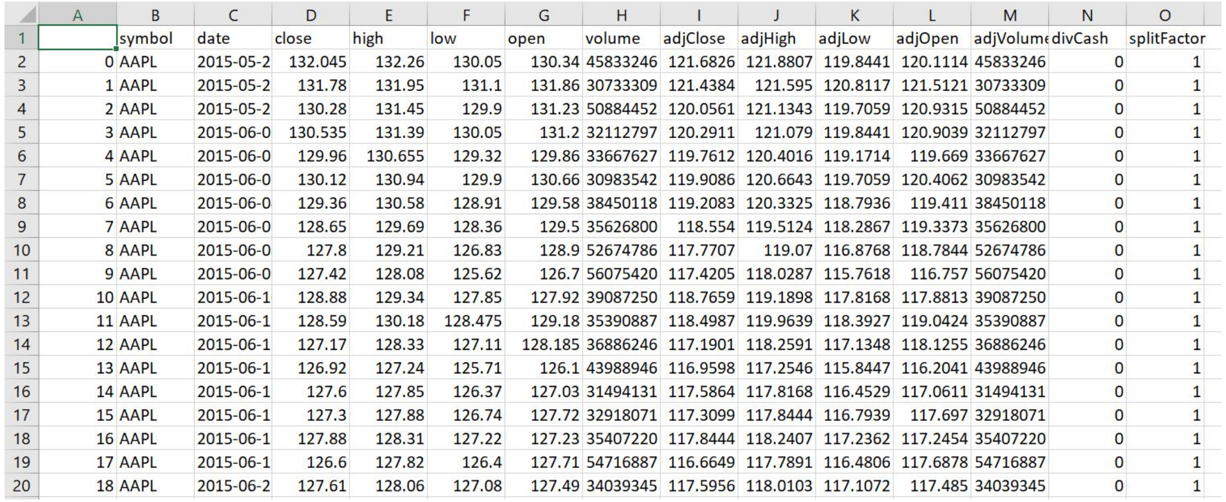




Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. There are three different gates in an LSTM cell: a forget gate, an input gate, and an output gate. Relative insensitivity to gap length due to these gates solves the long-term dependency problem is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

WORKING



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		symbol	date	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash	splitFactor
2	0	AAPL	2015-05-2	132.045	132.26	130.05	130.34	45833246	121.6826	121.8807	119.8441	120.1114	45833246	0	1
3	1	AAPL	2015-05-2	131.78	131.95	131.1	131.86	30733309	121.4384	121.595	120.8117	121.5121	30733309	0	1
4	2	AAPL	2015-05-2	130.28	131.45	129.9	131.23	50884452	120.0561	121.1343	119.7059	120.9315	50884452	0	1
5	3	AAPL	2015-06-0	130.535	131.39	130.05	131.2	32112797	120.2911	121.079	119.8441	120.9039	32112797	0	1
6	4	AAPL	2015-06-0	129.96	130.655	129.32	129.86	33667627	119.7612	120.4016	119.1714	119.669	33667627	0	1
7	5	AAPL	2015-06-0	130.12	130.94	129.9	130.66	30983542	119.9086	120.6643	119.7059	120.4062	30983542	0	1
8	6	AAPL	2015-06-0	129.36	130.58	128.91	129.58	38450118	119.2083	120.3325	118.7936	119.411	38450118	0	1
9	7	AAPL	2015-06-0	128.65	129.69	128.36	129.5	35626800	118.554	119.5124	118.2867	119.3373	35626800	0	1
10	8	AAPL	2015-06-0	127.8	129.21	126.83	128.9	52674786	117.7707	119.07	116.8768	118.7844	52674786	0	1
11	9	AAPL	2015-06-0	127.42	128.08	125.62	126.7	56075420	117.4205	118.0287	115.7618	116.757	56075420	0	1
12	10	AAPL	2015-06-1	128.88	129.34	127.85	127.92	39087250	118.7659	119.1898	117.8168	117.8813	39087250	0	1
13	11	AAPL	2015-06-1	128.59	130.18	128.475	129.18	35390887	118.4987	119.9639	118.3927	119.0424	35390887	0	1
14	12	AAPL	2015-06-1	127.17	128.33	127.11	128.185	36886246	117.1901	118.2591	117.1348	118.1255	36886246	0	1
15	13	AAPL	2015-06-1	126.92	127.24	125.71	126.1	43988946	116.9598	117.2546	115.8447	116.2041	43988946	0	1
16	14	AAPL	2015-06-1	127.6	127.85	126.37	127.03	31494131	117.5864	117.8168	116.4529	117.0611	31494131	0	1
17	15	AAPL	2015-06-1	127.3	127.88	126.74	127.72	32918071	117.3099	117.8444	116.7939	117.697	32918071	0	1
18	16	AAPL	2015-06-1	127.88	128.31	127.22	127.23	35407220	117.8444	118.2407	117.2362	117.2454	35407220	0	1
19	17	AAPL	2015-06-1	126.6	127.82	126.4	127.71	54716887	116.6649	117.7891	116.4806	117.6878	54716887	0	1
20	18	AAPL	2015-06-2	127.61	128.06	127.08	127.49	34039345	117.5956	118.0103	117.1072	117.485	34039345	0	1

It works on the Dataset as shown, it is a .csv file that contains 1258 records each corresponding to the various information related to stocks of a company (Apple in this case) for a particular day.

We will be predicting the Stock Prices using the close price on a day using LSTMs.

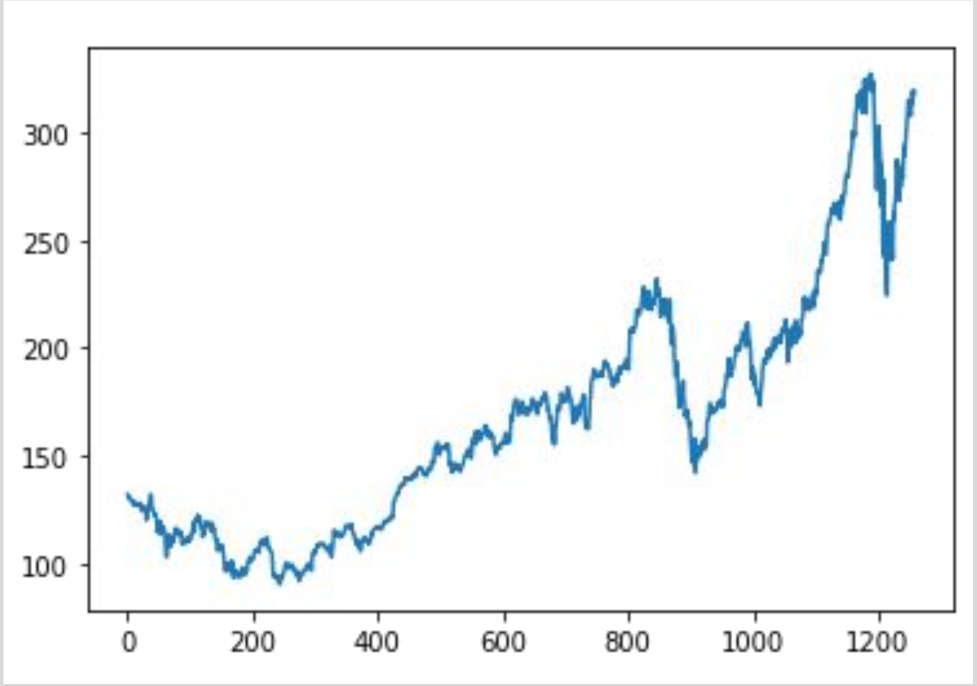
With the help of LSTMs we train the model such that it studies the close prices for 100 days, with the help of the trends observed the model predicts the close price for the 101th day.

Graph for Close Prices for Apple

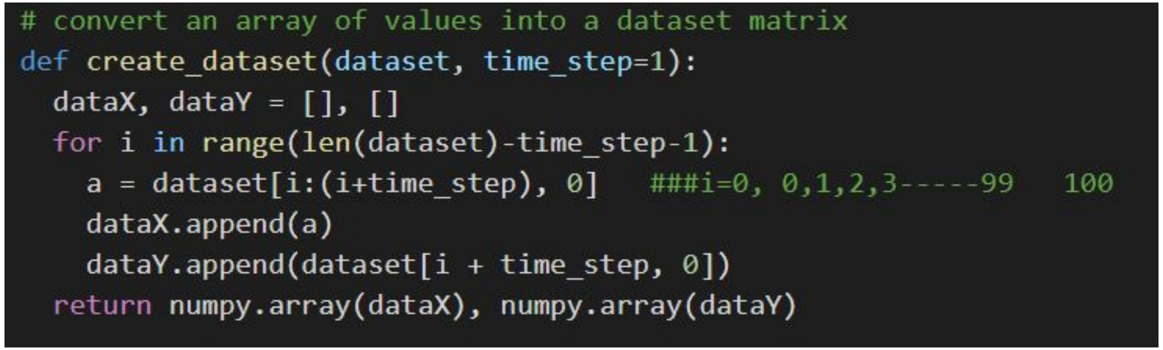
LSTM is used to study the trends as shown in the graph and make predictions for the close prices of the stock. We divide the data into training set and test set. With 65% of the data used for training while the rest is used for predictions.

We then calculate various error metric to see how our model performs.

LSTMs require data to be scaled to avoid any possible discrepancies.

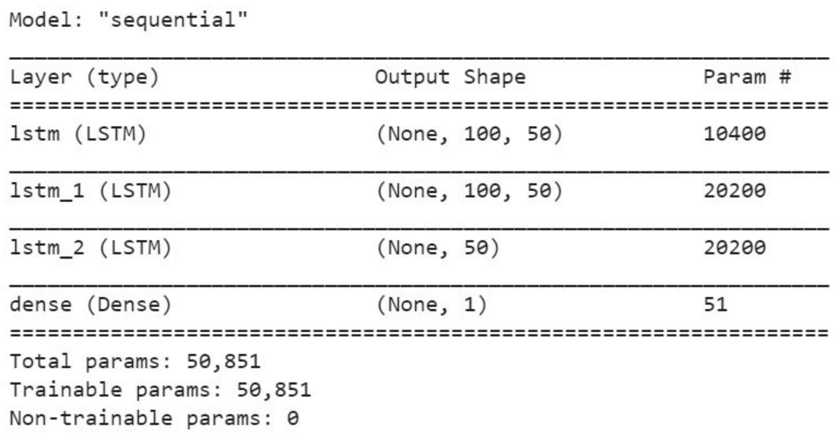


Conversion Of Dataset

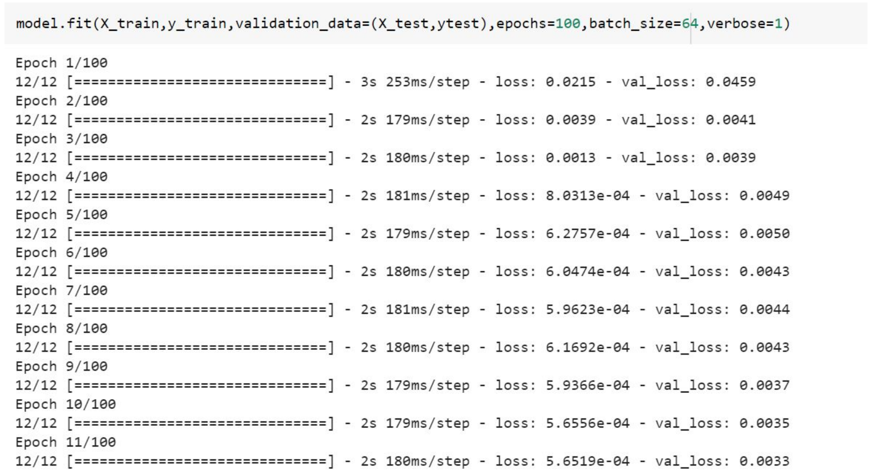


```
# convert an array of values into a dataset matrix
def create_dataset(dataset, time_step=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-time_step-1):
    a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
    dataX.append(a)
    dataY.append(dataset[i + time_step, 0])
  return numpy.array(dataX), numpy.array(dataY)
```

We convert the provided dataset into a time series data. With the help of LSTMs we train the model using the time series data such that it studies the close prices for 100 days, with the help of the trends observed the model predicts the close price for the 101th day. We create smaller datasets containing 100 days of closing stock prices and push it into an array. We also create another array with stores the stock price for 101th day for comparison and training purposes. We repeat the above steps to generate training and testing datasets.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 50)           10400
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 50)           20200
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
```



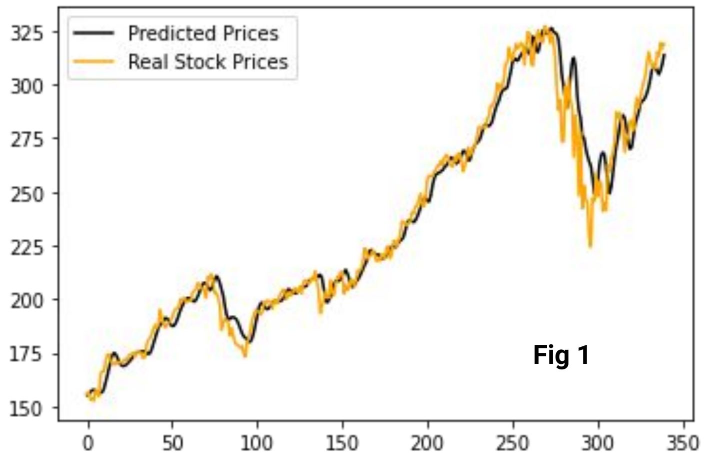
```
model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=100,batch_size=64,verbose=1)
```

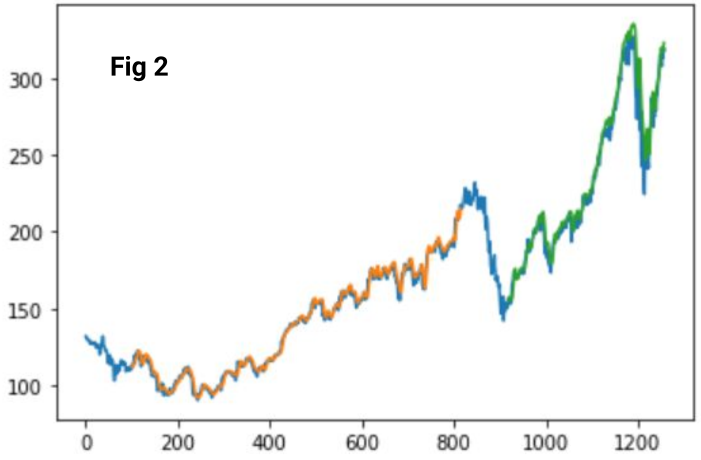
```
Epoch 1/100
12/12 [==============================] - 3s 253ms/step - loss: 0.0215 - val_loss: 0.0459
Epoch 2/100
12/12 [==============================] - 2s 179ms/step - loss: 0.0039 - val_loss: 0.0041
Epoch 3/100
12/12 [==============================] - 2s 180ms/step - loss: 0.0013 - val_loss: 0.0039
Epoch 4/100
12/12 [==============================] - 2s 181ms/step - loss: 8.0313e-04 - val_loss: 0.0049
Epoch 5/100
12/12 [==============================] - 2s 179ms/step - loss: 6.2757e-04 - val_loss: 0.0050
Epoch 6/100
12/12 [==============================] - 2s 180ms/step - loss: 6.0474e-04 - val_loss: 0.0043
Epoch 7/100
12/12 [==============================] - 2s 181ms/step - loss: 5.9623e-04 - val_loss: 0.0044
Epoch 8/100
12/12 [==============================] - 2s 180ms/step - loss: 6.1692e-04 - val_loss: 0.0043
Epoch 9/100
12/12 [==============================] - 2s 179ms/step - loss: 5.9366e-04 - val_loss: 0.0037
Epoch 10/100
12/12 [==============================] - 2s 179ms/step - loss: 5.6556e-04 - val_loss: 0.0035
Epoch 11/100
12/12 [==============================] - 2s 180ms/step - loss: 5.6519e-04 - val_loss: 0.0033
```

Training the LSTM Model

We utilise a Sequential Model (A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor) and we use 4 layers of LSTM in our model.

We run 100 epochs in batches of 64. Epochs refers to the number of times the model is trained over the same data in order to increase the accuracy.



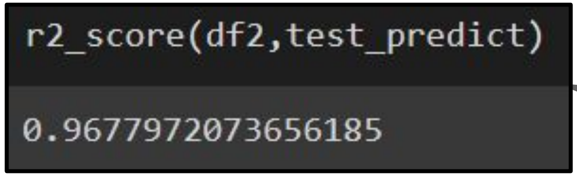


Predictions Made By the Model

In fig 1 we compare real stock prices (test data) with predicted stock prices. In fig 2 we observe both training as well as test predictions overlapping with the original graph for stock prices of the company.

The overlapping of the prediction model with our original data set represents a high level of accuracy of our model with minimal loss.

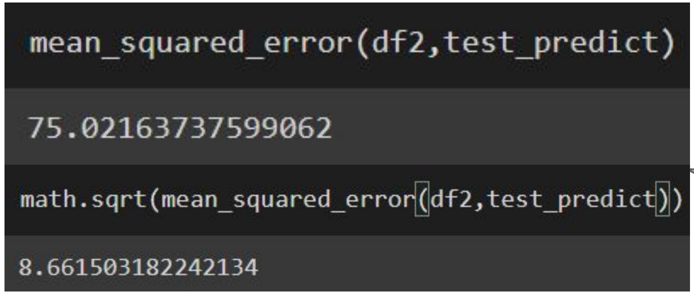
Test Metrics



```
r2_score(df2,test_predict)
0.9677972073656185
```

R^2 Test

We can see that the R^2 value is closer to one which also indicates that our model has high accuracy.



```
mean_squared_error(df2,test_predict)
75.02163737599062
math.sqrt(mean_squared_error(df2,test_predict))
8.661503182242134
```

Mean Squared Error And Root Mean Squared Error

The smaller values of the error tests also indicate that our model is accurate.

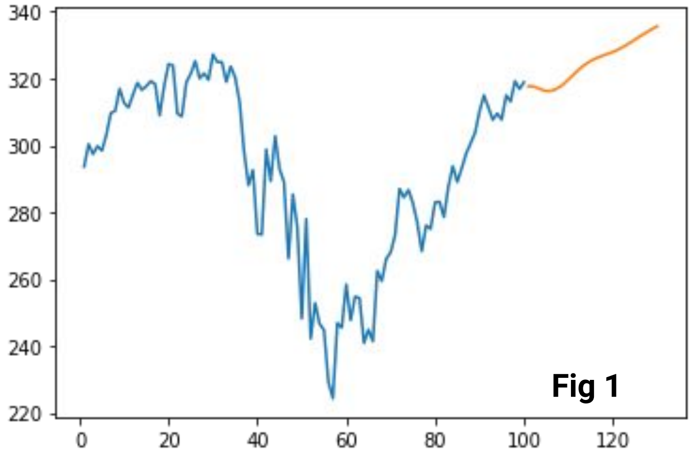


Fig 1

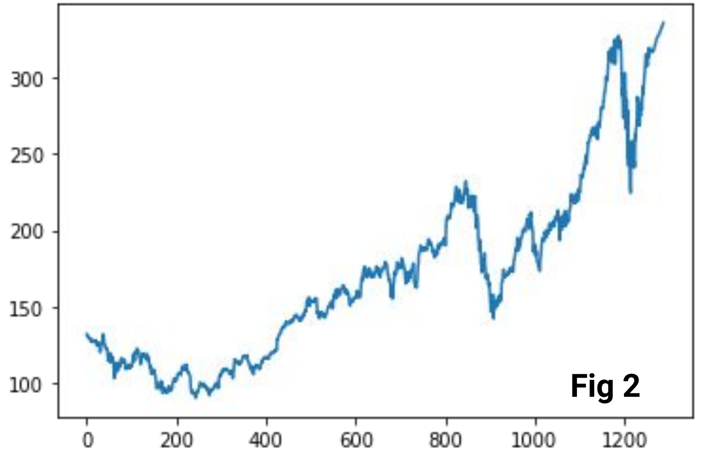


Fig 2

Forecasting of stock prices for the next 30 days

The Orange Line shows the Stock Pricing forecasted for the next 30 days and it suggests it will show an increasing trend in stock prices in the coming days (Fig 1).

The final graph combining the existing Dataset with the forecasted stock prices (shown in Fig 2).

Conclusion

With the help of this project we learnt about Neural Networks, RNNs, problems faced due to RNNs like Long Term Dependencies, exploding and vanishing gradient problem fixed this issue using Long Short Term Memory (LSTM).

We then used a Recurrent neural network LSTM for making stock price predictions by gathering data from multiple reliable sources. We used the close prices in order to train the model to forecast the close prices for the next 30 days. We then compared it to the real values and found it to be somewhat closer to them.

The model was able to predict the future trends effectively after training. The model can be further expanded to predict the stock prices for more days with more accuracy and can help investors in making effective investment decisions.

THANK YOU!