Name : Thines Kumar Nadaraja (TP063097)

Intake Code : APUMF2101DSBA(DE)(PR)

Module Name : Deep Learning

Module Code : CT100-3-M-DL

Lecturer : Prof. Dr. Mandava Rajeswari

Assignment Title : Deep Learning for Stock Market Prediction

Table of Contents

Table of Figures

Table of Tables

DEEP LEARNING FOR STOCK MARKET PREDICTION

Introduction

The stock market is a trading zone for consumers to buy and sell shares of a company, based on the current price, and expected future price of the share. The shareholders of a company are owners of the company, and trading these shares enables the value of the company to grow exponentially. However, trade analysts and share market experts have difficulties in predicting trends in the market, as there are various factors that affect the price of a stock. Research shows that with the application of deep learning, predicting the stock market using these techniques can improve accuracy and provide better returns on investment (ROI) for traders in the market. There are various deep learning models, such as Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) that will be explored further in this paper. The conclusion is that, in the contemporary times, deep learning models are ideal for prediction in the stock market.

Keywords: deep learning; stock market; prediction

Literature Review

The field of Artificial Intelligence (AI) has brought upon changes in the way predictions and forecasting works (Hu et al., 2021). (Selvamuthu et al., 2019) states that AI can predict with a high degree of accuracy, trends, and forecasts in various domains. Predicting trends in the stock market has proven to be a gamely task (Agrawal et al., 2022). Deep learning consists of extremely complicated non-linear connections between variables (Hu et al., 2021). Models used in deep learning has been utilized in various research fields, of which, stock market prediction is one of them (Selvamuthu et al., 2019). Changes that occur in the stock market have been reflected by predictions made by trade analysts (Kalyoncu et al., 2020). Stock market traders use a variety of mathematical algorithms and statistical applications to predict trends in the graphs of a stock (Rezaei et al., 2021). However, the application of deep learning differs from conventional statistical and mathematical methods (Selvamuthu et al., 2019).

There are two conventional evaluation that trade analysts use in the stock market: fundamental analysis and technical analysis (Hu et al., 2021). Fundamental analysis consists of stock assessment of a public listed company via the core value of the organization (Nunes et al., 2019). This ensures that long-term investment can be made in the company (Selvamuthu et al., 2019), as previous and current data is analysed to assess the financial performance of said organization. This differs from technical analysis, which is more in-depth and focused on terms of prices of stock (Nabipour, Nayyeri, Jabani, S., et al., 2020), past ROI, and trade volume (Li & Pan, 2021). Technical analysis is used for traders due to its short-term view on the stock market (Hu et al., 2021).

Forecasting prices of a stock in the market can be performed using regression analysis (Nabipour, Nayyeri, Jabani, S., et al., 2020), which is where deep learning plays a pivotal role (Li & Pan, 2021). The Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) is used for stock market prediction, as shown in the equations below:

$$RMSE = \sqrt{(\frac{1}{N} \times \sum_{t=1}^{N} |At - Ft^2|)}$$

Equation 1: Root Mean Square Error Function (Rezaei et al., 2021)

$$MAPE = \frac{1}{N} \times \sum_{t=1}^{N} \left| \frac{At - Ft}{At} \right|$$

Equation 2: Mean Absolute Percentage Error (%) Function (Rezaei et al., 2021)

where N = number of time points, At = actual stock price, Ft = Forecast value of stock price.

RMSE calculates the significant variance between the actual value of the stock price and its forecasted value (Rezaei et al., 2021). MAPE calculates in percentage, the average variance between forecasted stock price and the true stock price (Kalyoncu et al., 2020). MAPE of 7% indicates that the mean variation between Ft and At is 7% (Zhang et al., 2020). RMSE is pivotal in correcting significant inaccuracies in the Deep Learning algorithms (Nunes et al., 2019), of which the result in the outcome variables match that of the units in the input variables (Li & Pan, 2021). Moreover, the usage of RMSE enables computation of validation of the algorithms, utilizing a loss function (Agrawal et al., 2022). However, there are limitations with using RMSE (Sen & Mehtab, 2020). Firstly, RMSE rises alongside the variance of the rate of recurrence allocation of error levels. Next, RMSE is not definitely construed, due to the

variations in the squaring functions (Mehtab et al., 2021). RMSE is however, a pivotal algorithm for deep neural networks (Selvamuthu et al., 2019).

Problem Statement

Deep learning applies neural networks in forecasting of stock markets, because of the adaptable landscape of said networks. (Li & Pan, 2021) states that the index movement of the price of a stock, implemented using two models; Artificial Neural Network (ANN) and Support Vector Machines (SVM) were compared. ANN fared better than SVM models on average. (Shen & Shafiq, 2020) also states that applying prediction of stock market using ANN can produce extrapolation of the trend charts in the stock market. However, (Rezaei et al., 2021) states that Long Short-Term Memory (LSTM) is widely used in stock market prediction, with a significantly greater accuracy (Zhang et al., 2020). The architecture of LSTM is shown below:

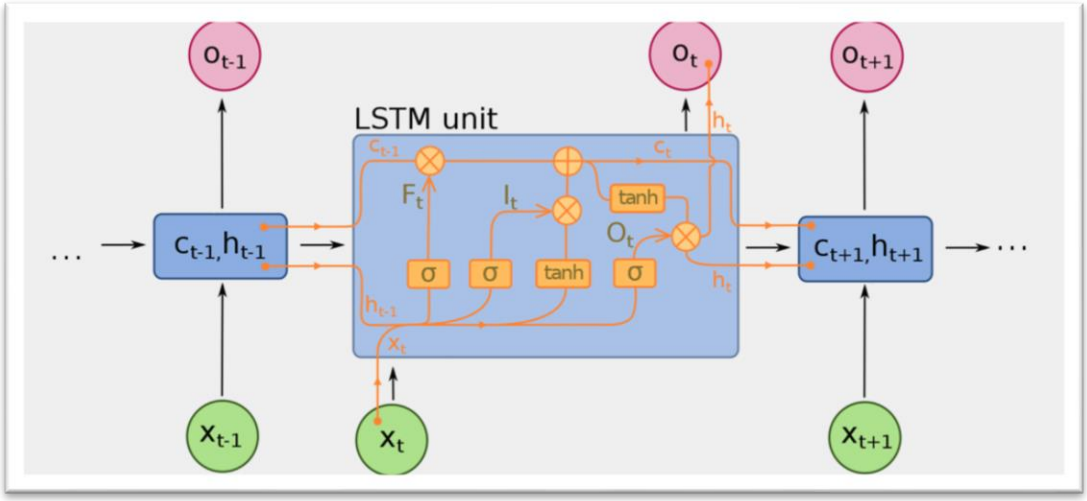


Figure 1: LSTM Model for Deep Learning (Rezaei et al., 2021)

From Figure (1), the LSTM model has the following properties:

- Forget gate (F_t) – the data from the existing cell is chosen by the F_t to be dropped, which is a mathematical sigmoid function (Sen & Mehtab, 2020).
- Input gate (I_t) – the data is chosen by the I_t to be combined and deposited in the existing cell gate, using a *tanh* function (Sen & Mehtab, 2020).
- Output gate (O_t) – O_t acts in controlling the production that runs to the adjacent cell (Agrawal et al., 2022).

LSTM is essentially a unique algorithm derived from RNN, that is efficient in long-term needs. LSTM contains modules that repeat in interacting layers (Mehtab et al., 2021).

Various research articles state the comparisons between Simple Moving Average (SMA), Exponential Moving Average (EMA) and LSTM, as depicted in the table below:

Model	**SMA**	**EMA**	**LSTM**
Monitoring Time (seconds)	6,083	4,910	3,731
(%) logs/MAPE	12.53	10.72	2.37
logs/RMSE	43.77	36.68	12.63
Monitoring/CPU	10.7	4	9.7
Monitoring/memory	13.71	13.71	13.71

Table 1: Comparison between Existing Models and LSTM (Selvamuthu et al., 2019)

From Table (1), the Central Processing Unit (CPU) and memory usage between EMA, SMA and LSTM are somewhat similar, however LSTM has a lesser monitoring time and more efficient RMSE and MAPE calculation method. (Kalyoncu et al., 2020) states that the methodology of using LSTM is in three steps:

1. The data that is omittable from the current cell must be settled. This is computed by the usage of a sigmoid function (Agrawal et al., 2022).
2. The second layer of the LSTM algorithm has two functions: sigmoid and tanh function. The sigmoid algorithm determines the value (0 or 1) to be passed to the input gate, whilst the tanh function provides weightage assigned to the values through to the input gate (-1 to 1) (Sen & Mehtab, 2020).
3. The output gate, which determines the final output of the LSTM model. Once the sigmoid function decides the values that goes to the output, the tanh function will then ensure that the values which fall within its spectrum (-1 to 1) is augmented by the yield of the sigmoid function (Rezaei et al., 2021).

(Selvamuthu et al., 2019) provides an example of the LSTM function used to predict the price of stock of Google, in the figure below:

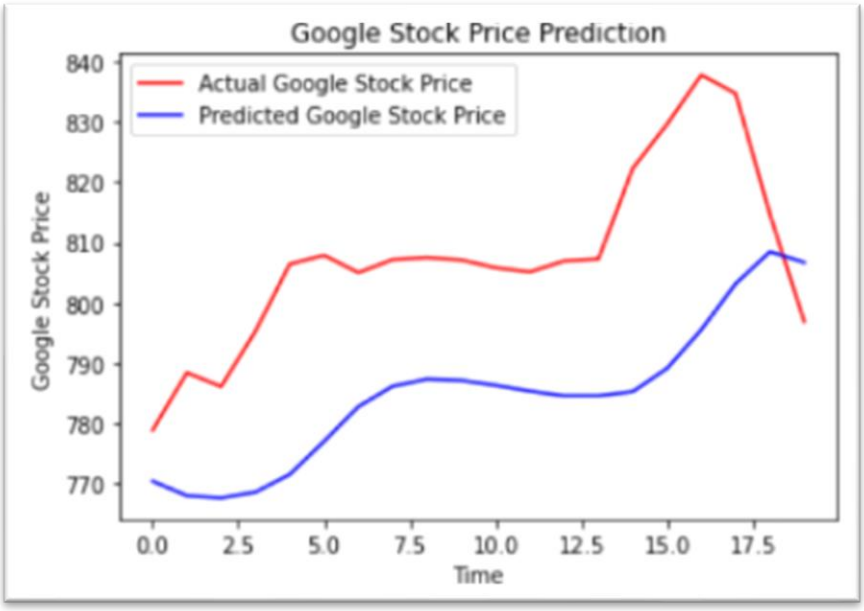


Figure 2: Google Stock Price Prediction using LSTM (Selvamuthu et al., 2019)

Figure (2) shows that the actual stock price versus the predicted stock price, using LSTM, has no significant difference. Whilst this model may not be optimum for daily traders (Nabipour et al., 2020), consumers willing to patiently trade in the stock market can still consider this solution a viable option (Rezaei et al., 2021).

The primary reason for using deep learning in forecasting stock market prices, is due to the pace and complexity of its algorithms in making predictions (Kalyoncu et al., 2020). Deep learning algorithms can examine the data for functions that are correlative (Shen & Shafiq, 2020), thus enabling quicker computations (Mehtab et al., 2021). Moreover, deep learning algorithms can work with various types of data, mainly unstructured data (Sen & Mehtab, 2020). Stock market data is mostly unstructured (Agrawal et al., 2022), with no pre-defined model and can sometimes be extracted in images or reports (Rezaei et al., 2021). Next, deep learning is effective at producing superior quality outcomes (Nunes et al., 2019), which is vital in predicting the stock market (Mehtab et al., 2021). The vast amounts of data that is available in the stock market, in terms of price changes and various other financial information (Chun et al., 2020), enables the usages of deep learning algorithms, as such algorithms work efficiently with greater data to provide a more accurate outcome (Sen & Mehtab, 2020). These factors ensure that deep learning algorithms are beneficial in predicting the outcome of a stock price (Kalyoncu et al., 2020).

Review of Existing Deep Learning Algorithms

According to (Mehtab et al., 2021), compilation of various research articles shows the frequency of each Deep Learning algorithms, depicted in Figure (3) below:

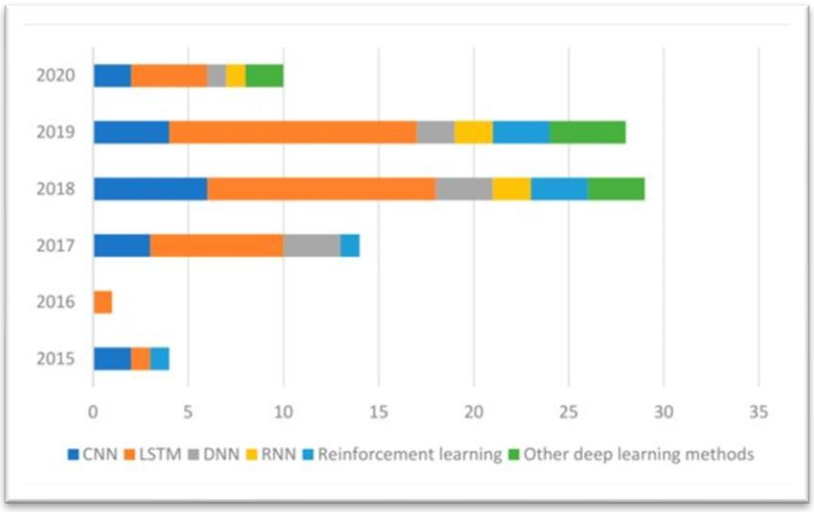


Figure 3: Research Articles Compilation of Various Deep Learning Models (Mehtab et al., 2021)

LSTM is the most widely used model, based on research articles compiled from 2015 until 2020. RNN and CNN are also widely used. LSTM has been used as early as 2015, and as for the year 2016, LSTM is widely used to predict trends in the stock market. However, as the years advance, various other models have come up, such as the Reinforcement learning model. CNN is still prominent in its usage for forecasting stock market, as depicted in Figure (3).

Another research article (Chun et al., 2020) compiles the average performance of Deep Learning models based on the share market in the United States of America, with error measures such as MAE, RMSE and MAPE, as shown in Figure (4) below:

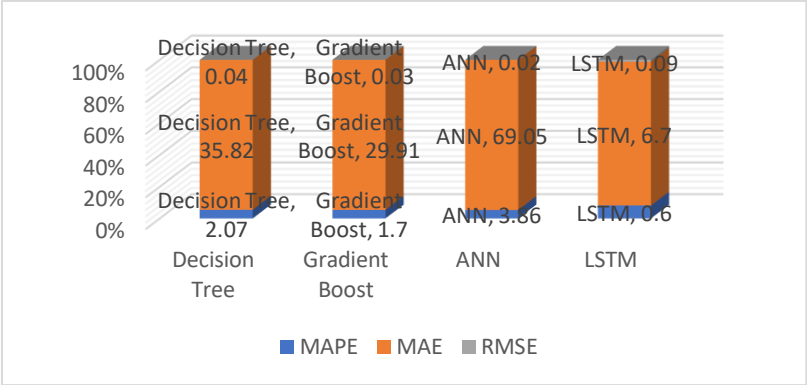


Figure 4: Error Measures of Deep Learning Models for Stock Market Prediction in the USA

From Figure (4), the error measures for LSTM are the least, compared to the other models. Only the models that were prominently used was selected for comparison in Figure (3). LSTM outperformed ANN, Decision Trees and Gradient Boost algorithms, to successfully predict the stock market for a diversified stock portfolio in the USA (Kalyoncu et al., 2020).

A survey that provides a comparative study of Deep Learning algorithms based on research articles for stock price forecasting, is shown in the table below:

Title	Authors	Year	Model Used
Stock Market Prediction using Machine Learning Algorithms	M. Misra, A. Yadav, H. Kaur	2018	SVM, Linear Regression, Principal Component Analysis
Stock Market Prediction using Machine Learning Techniques	N. Sirimevan, I. Mamalgaha, C. Jayasekara,	2019	Sentimental Analysis, RNN, LSTM
Survey of Stock Market Prediction using Machine Learning	A. Sharma, D. Bhuriya, U. Singh	2017	Linear Regression
Empirical Study on Stock Market Prediction using Machine Learning	R. Sable, S. Goel, P. Chatterjee	2019	LSTM, ARIMA
Predicting NASDAQ and NSE Stocks using Machine Learning Algorithms	A. Narote, K. Jadhav, J. Barot, S. Sawant	2020	LSTM, Linear Regression
A Machine Learning Model for Stock Market Prediction	Osman Hegazy, Omar S. Soliman and Mustafa Abdul Salam	2014	LSTM
Prediction of Stock Market using Machine Learning Algorithms	Neha Bhardwaj, MD Akil Ansari	2019	LSTM, Logistic Regression, Random Forest
Stock Market Prediction using Machine Learning	Ishita Parmar, Ridam Arora, Sheirsh Saxena, et al	2018	LSTM, Regression Based Model
Predicting Stock Market Trends Using Machine Learning and Deep Learning	M Nabipour, P. Nayyeri, H Jabani, Shahab S, A Mosavi	2020	LSTM, RNN, SVM, KNN
NSE Stock Market Prediction using Deep Learning Models	Gopalakrishnan E A, Hiransha M, Vijay K M, Soman K P	2018	LSTM, RNN, CNN
Stock Market Forecasting using Machine Learning: Today and Tomorrow	Sukhman Singh, Tarun Kumar Madan, Jitendra Kumar, Ashutosh Kumar Singh	2019	LSTM, Ensemble, ABS Regression

Table 2: Comparison of Various Deep Learning Models for Stock Market Prediction (Kalyoncu et al., 2020)

From Table (2), LSTM is the most widely used algorithm for Deep Learning models, based on the survey of various research articles over the past 7 years.

Hence, from Table (2), based on the study of current deep learning algorithms that have been in place for stock market predictions, the LSTM algorithm is the chosen model for this assignment. LSTM is beneficial in predicting stock markets due to the ability of this deep learning model to compute sequences (Nabipour et al., 2020), which is prominent for stock market prices (Kalyoncu et al., 2020). LSTM has a feedback loop, which creates what is essentially a short-term and long-term memory module (Mehtab et al., 2021). Data pertaining to the stock market are in the time-series format, and with the usage of LSTM memory modules, prior timestep data can be collected and trained for deep learning (Jiang, 2021). However, LSTM tends to overfit models, which is a drawback in terms of the accuracy of stock price prediction (Mehtab et al., 2021).

Discussion

As mentioned in the section above, LSTM is the chosen deep learning model for this assignment. The dataset chosen is the Stock Price of Apple Inc., from the USA (Mishra et al., 2021). This dataset contains the following variables, depicted in the table below:

Variable	Description
Close Price	Price of stock at closing time (5 pm)
Volume	The trade volume of the stock
Open	Opening price of stock (9 am)
High	The highest price value of stock for the day
Low	The lowest price value of stock for the day

Table 3: Apple Stock Price Dataset from 2010 – 2020 (Mishra et al., 2021)

Existing algorithms have been carried out in this dataset, and LSTM was previously used to predict the closing stock price of Google Inc (Ma et al., 2021). For this assignment, the purpose is to predict both the closing stock price and the high stock price of Apple Inc., for trading purposes. Accurately determining the high stock price is important for traders, as they use functions such as Take Profit and Stop/Loss during trading, which closely monitors the High price of said stock (Jiang, 2021).

The metrics to be evaluated are the Close Price and High variables, as this significantly contributes to the final stock price (Ma et al., 2021). Based on these metrics, the final prediction of both the Close and High prices are computed using LSTM algorithms, via Python programming (Long et al., 2020).

References

1. Hu, Z., Zhao, Y., & Khushi, M. (2021). A Survey of Forex and Stock Price Prediction Using Deep Learning. Applied System Innovation, 4(1), 9. https://doi.org/10.3390/asi4010009
2. Selvamuthu, D., Kumar, V., & Mishra, A. (2019). Indian stock market prediction using artificial neural networks on tick data. Financial Innovation, 5(1). https://doi.org/10.1186/s40854-019-0131-7
3. Jiang, W. (2021). Applications of deep learning in stock market prediction: Recent progress. Expert Systems with Applications, 184, 115537. https://doi.org/10.1016/j.eswa.2021.115537
4. Mishra, M., Patil, M., Raut, G., & Chaudhari, T. (2021). A SURVEY ON STOCK PRICE PREDICTION USING MACHINE LEARNING. International Journal of Engineering Applied Sciences and Technology, 5(11). https://doi.org/10.33564/ijeast.2021.v05i11.033
5. Long, J., Chen, Z., He, W., Wu, T., & Ren, J. (2020). An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market. Applied Soft Computing, 91, 106205. https://doi.org/10.1016/j.asoc.2020.106205
6. Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E., & S., S. (2020). Deep Learning for Stock Market Prediction. Entropy, 22(8), 840. https://doi.org/10.3390/e22080840
7. Shen, J., & Shafiq, M. O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. Journal of Big Data, 7(1). https://doi.org/10.1186/s40537-020-00333-6
8. Rezaei, H., Faaljou, H., & Mansourfar, G. (2021). Stock price prediction using deep learning and frequency decomposition. Expert Systems with Applications, 169, 114332. https://doi.org/10.1016/j.eswa.2020.114332
9. Li, Y., & Pan, Y. (2021). A novel ensemble deep learning model for stock prediction based on stock prices and news. International Journal of Data Science and Analytics. Published. https://doi.org/10.1007/s41060-021-00279-9
10. Ma, Y., Han, R., & Wang, W. (2021). Portfolio optimization with return prediction using deep learning and machine learning. Expert Systems with Applications, 165, 113973. https://doi.org/10.1016/j.eswa.2020.113973
11. Mehtab, S., Sen, J., & Dutta, A. (2021). Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models. Communications in Computer and Information Science, 88–106. https://doi.org/10.1007/978-981-16-0419-5_8
12. Sen, J., & Mehtab, S. (2020). A Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models. International Journal of Business Forecasting and Marketing Intelligence, 6(1), 272. https://doi.org/10.1504/ijbfmi.2020.10038524
13. Zhang, Y., Yan, B., & Aasma, M. (2020). A novel deep learning framework: Prediction and analysis of financial time series using CEEMD and LSTM. Expert Systems with Applications, 159, 113609. https://doi.org/10.1016/j.eswa.2020.113609
14. Nabipour, M., Nayyeri, P., Jabani, H., S., S., & Mosavi, A. (2020). Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data; a Comparative Analysis. IEEE Access, 8, 150199–150212. https://doi.org/10.1109/access.2020.3015966
15. Chun, J., Ahn, J., Kim, Y., & Lee, S. (2020). Using Deep Learning to Develop a Stock Price Prediction Model Based on Individual Investor Emotions. Journal of Behavioral Finance, 22(4), 480–489. https://doi.org/10.1080/15427560.2020.1821686
16. S. Kalyoncu, A. Jamil, E. Karataş, J. Rasheed, and C. Djeddi, “Stock Market Value Prediction using Deep Learning”, DataSCI, vol. 3, no. 2, pp. 10-14, Dec. 2020.
17. Agrawal, M., Kumar Shukla, P., Nair, R., Nayyar, A., & Masud, M. (2022). Stock Prediction Based on Technical Indicators Using Deep Learning Model. Computers, Materials & Continua, 70(1), 287–304. https://doi.org/10.32604/cmc.2022.014637
18. Nunes, M., Gerding, E., McGroarty, F., & Niranjan, M. (2019). The Memory Advantage of Long Short-Term Memory Networks for Bond Yield Forecasting. SSRN Electronic Journal. Published. https://doi.org/10.2139/ssrn.3415219

PART 2

Model Implementation

The dataset used for this assignment is titled “HistoricalQuotes” which is the dataset reflecting the stock price of Apple Inc. from the year 2010 until the year 2020. This dataset was procured from the following URL: https://www.kaggle.com/sarthak1799/apple-stock-prediction/data Once the dataset was uploaded onto Colab, data pre-processing was performed.

Firstly, the following packages were imported for data pre-processing:

```
pip install numpy
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (1.19.5)
```

```
import pandas as PD
import numpy as np
%matplotlib inline
import matplotlib. pyplot as plt
import matplotlib
```

```
!pip install -U scikit-learn scipy matplotlib
!pip install sklearn
```

```
import tensorflow as tf
from tensorflow import keras

from sklearn.preprocessing import MinMaxScaler
from keras. layers import LSTM, Dense, Dropout
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib. dates as mandates
from sklearn. preprocessing import MinMaxScaler
from sklearn import linear_model
from keras. models import Sequential
from keras. layers import Dense
import keras. backend as K
from keras. callbacks import EarlyStopping
#from keras. optimisers import Adam
from keras. models import load_model
from keras. layers import LSTM
from keras. utils.vis_utils import plot_model
```

As these packages have been installed, pre-processing of the dataset can begin.

```
#Get the Dataset
from google.colab import files
uploaded = files.upload()
```

Choose Files HistoricalQuotes.csv

- **HistoricalQuotes.csv**(application/vnd.ms-excel) - 145669 bytes, last modified: 2/29/2020 - 100% done

```
Saving HistoricalQuotes.csv to HistoricalQuotes.csv
```

The dataset (Kaggle, 2021) was uploaded onto the Colab drive.

Next, a quick look at the dataset (to see the first 5 columns of the dataset) is performed using the following code:

```
import pandas as pd
df = pd.read_csv('HistoricalQuotes.csv')
df.head() #Taking a quick look at the dataset (first 5 columns and rows)
```

The output would be:

	Date	Close/Last	Volume	Open	High	Low
0	02/28/2020	$273.36	106721200	$257.26	$278.41	$256.37
1	02/27/2020	$273.52	80151380	$281.1	$286	$272.96
2	02/26/2020	$292.65	49678430	$286.53	$297.88	$286.5
3	02/25/2020	$288.08	57668360	$300.95	$302.53	$286.13
4	02/24/2020	$298.18	55548830	$297.26	$304.18	$289.23

From the output, the target for prediction of Stock Price would be the Close/Last column. This is because, with the usage of LSTM, the stock price at the end of each day (which is the Close/Last price) can be predicted. The reason for predicting this price is because this value is pivotal for stock traders, as prediction of the closing price of the stock for one day gives an overall bearing for the opening price for the following day. Moreover, predicting this stock price is pivotal for traders to buy or sell stocks based on their target prices.

Next, checking the dataset for Null values is performed with the following code and output:

```
#Print the shape of Dataframe  and Check for Null Values
from pandas import Series, DataFrame
print('Dataframe Shape: ', df. shape) #To check shape of data
print('Null Value Present: ', df.isnull().values.any()) #To check if any null values present in data
```

```
Dataframe Shape:  (2518, 6)
Null Value Present:  False
```

Next, the information of the dataset is produced, as shown below:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2518 entries, 0 to 2517
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         2518 non-null   object
 1    Close/Last  2518 non-null   object
 2    Volume      2518 non-null   int64
 3    Open        2518 non-null   object
 4    High        2518 non-null   object
 5    Low         2518 non-null   object
dtypes: int64(1), object(5)
memory usage: 118.2+ KB
```

As can be seen above, the type for Close/Last is a non-null type. This is because of the $ sign in front of each value in the columns for Close/Last (and every other column). However, since the target is only Close/Last stock price, the $ for this variable will be removed with the following code:

```
df1 = df.iloc[:,1] #Selecting feature of Close/Last stoc
df1.head()
```

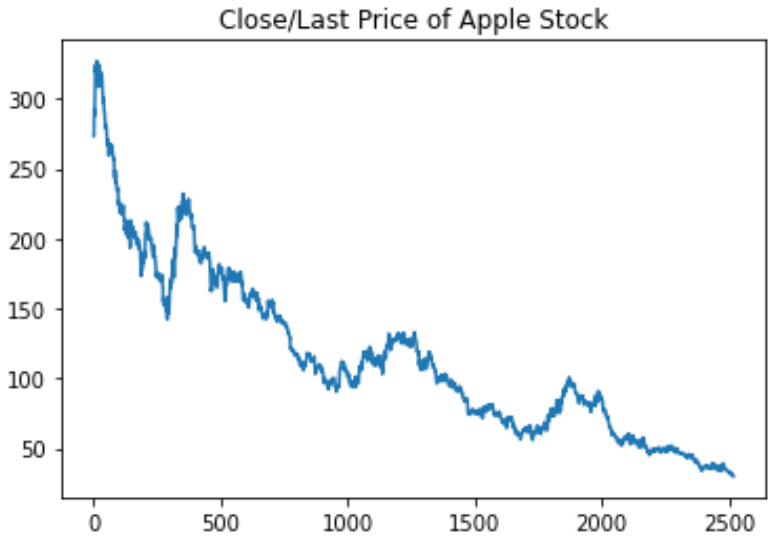
```
0     $273.36
1     $273.52
2     $292.65
3     $288.08
4     $298.18
Name:  Close/Last, dtype: object
```

```
df1 = df1.replace('[\$,]', '', regex=True).astype(float)
df1.head()
```

```
0    273.36
1    273.52
2    292.65
3    288.08
4    298.18
Name:  Close/Last, dtype: float64
```

Next, plotting of the graph for Close/Last price over the whole dataset (2500 rows of data from the year 2010 – 2020) is shown with the following code and output:

```
plt.plot(df1) #Plotting data using matplotlib library,
plt.title('Close/Last Price of Apple Stock')
```



The Close/Last price of stock for Apple declines with each day and year. This could be due to seasonal downtrend of these stock prices, over time. However, the purpose of this assignment is to predict stock price, so the focus will be towards achieving that objective.

Tuning/Validation

To tuning/validation of the dataset, the data must first be normalized. LSTM requires normalization of data, or any other Recurrent Neural Network for that matter. Using the MinMaxScaler package, a minimum and maximum scaled value of the dataset can be obtained. The values used for this assignment are the range of {0,1}.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0,1))
```

The shape of the new data frame is obtained with the following code:

```
df1.shape
```

```
(2518,)
```

The data frame is in the form of a vector. However, LSTM requires a 2-Dimensional array, using the MinMaxScaler. Hence, the data frame is converted into a 2-D array with the following code and output:

```
#df1 is a vector, based on the shape. MinMaxScaler howe
df1 = scaler.fit_transform(np.array(df1).reshape(-1,1))
print(df1)
print('\n df1 present shape : ', df1.shape)
```

```
[[8.18942624e-01]
 [8.19480684e-01]
 [8.83812549e-01]
 ...
 [2.30693463e-04]
 [0.00000000e+00]
 [6.72575693e-05]]

 df1 present shape :  (2518, 1)
```

The shape is now a 2-D array. Hence, the dataset can be split for testing and training purposes. Using a 70% - 30% split, for train – test, the following code and output is produced:

```
training_1 = int(len(df1)*0.7)
test_1 = len(df1) - training_1
print('Training Size : ',training_1)
print('Test Size : ',test_1)

data_train, data_test = df1[0:training_1,:],df1[training_1:len(df1),:]
print('Training Data Shape : ', data_train.shape)
print('Test Data Shape: ', data_test.shape)
```

```
Training Size :  1762
Test Size :  756
Training Data Shape :  (1762, 1)
Test Data Shape:  (756, 1)
```

The training dataset has a size of 1,762; the test dataset has a size of 756.

Once the splitting has been performed, the dataset is converted into series of layers that overlap. This is because Close/Last is a time-series dataset, which is univariate. The following code is needed to define the series of datasets that overlap:

```
def create_dataset(dataset, window=1):
    dataX, dataY= [], []
    for i in range(len(dataset)-window-1):
        a = dataset[i:(i+window),0]
        dataX.append(a)
        dataY.append(dataset[i+window,0])
    return np.array(dataX), np.array(dataY)
```

100 windows are used for this dataset, to create 100 layers of epoch later. The following code windows the dataset to 100:

```
window = 100
X_train, y_train = create_dataset(data_train, window=100)
X_test, y_test = create_dataset(data_test, window=100)

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(1661, 100) (1661,)
(655, 100) (655,)
```

The train and test shape for the X variable (independent variable) has 100 windows.

Since this dataset is in a 2-D array, it must be converted into 3-D for the purpose of applying LSTM. The code to convert 2-D into 3-D is shown below:

```
X_train = X_train.reshape(X_train.shape[0],X_train.shape[1],1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1],1)
```

Now that the training and testing dataset for the independent variable (Close/Last price) has been re-shaped, the LSTM algorithm can be performed:

```
model = Sequential()
model.add(LSTM(50, return_sequences = True, input_shape = (100,1)))
model.add(LSTM(50, return_sequences = True))
model.add(LSTM(50))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer = 'adam')
```

The model used for LSTM is Sequential, and the loss calculated is Mean Squared Error (MSE), which is pivotal for LSTM (as mentioned in Part 1).

The model.summary() codes show the following output:

```
model.summary() #The summary of the model with LSTM layers
```

```
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
_________________________________________________________________
```

Now, the data must be fit with the model, using Epoch training. 100 epochs are used to train the dataset, with a batch size of 64. Once this model is trained with 100 epochs, prediction of the value for Close/Last stock price can be performed.

The code for using 100 epochs are shown below:

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

```
Epoch 73/100
26/26 [==============================] - 5s 205ms/step - loss: 6.0318e-05 - val_loss: 1694821772566783342107
Epoch 74/100
26/26 [==============================] - 5s 198ms/step - loss: 6.0351e-05 - val_loss: 1694821772566783342107
Epoch 75/100
26/26 [==============================] - 5s 197ms/step - loss: 6.0404e-05 - val_loss: 1694821772566783342107
Epoch 76/100
26/26 [==============================] - 5s 191ms/step - loss: 6.0509e-05 - val_loss: 1694821772566783342107
Epoch 77/100
26/26 [==============================] - 5s 194ms/step - loss: 6.7906e-05 - val_loss: 1694821772566783342107
Epoch 78/100
26/26 [==============================] - 5s 198ms/step - loss: 7.2036e-05 - val_loss: 1694821772566783342107
Epoch 79/100
26/26 [==============================] - 5s 201ms/step - loss: 6.3978e-05 - val_loss: 1694821772566783342107
Epoch 80/100
26/26 [==============================] - 5s 201ms/step - loss: 6.6234e-05 - val_loss: 1694821772566783342107
Epoch 81/100
26/26 [==============================] - 5s 202ms/step - loss: 6.2408e-05 - val_loss: 1694821772566783342107
Epoch 82/100
26/26 [==============================] - 5s 206ms/step - loss: 7.3094e-05 - val_loss: 1694821772566783342107
Epoch 83/100
26/26 [==============================] - 5s 207ms/step - loss: 6.4236e-05 - val_loss: 1694821772566783342107
```

Training of the model is now complete. The model can now be predicted, but before performing that process, the predicted data for Close/Last must be inverse scaled, using scaler.inverse_transform:

```
pred_y = scaler.inverse_transform(model.predict(X_test))
pred_y[:5]
```

```
array([[94.291374],
       [93.87242 ],
       [93.2766  ],
       [93.63485 ],
       [95.11751 ]], dtype=float32)
```

The predicted values are of the range of 93 – 95, for the independent variable. Next, comparing these values to the test dataset, and by also inverse scaling the test dataset:

```
test_y = scaler.inverse_transform(y_test.reshape(-1,1))
```

+ Code + Text

```
test_y[:5]
```

```
array([[94.4728],
       [94.1985],
       [95.3007],
       [97.3314],
       [95.0257]])
```

The array for the test dataset is like the predicted dataset (from the first 5 values of prediction of the model). To further test the validity of these values, RMSE and R^2 values must be calculated:

```
import math
from sklearn.metrics import mean_squared_error

rmse_score = math.sqrt(mean_squared_error(test_y,pred_y))
print('Root Mean Squared Error(test) : ',rmse_score)
```

```
Root Mean Squared Error(test) :  1.2758585290133733
```

```
from sklearn.metrics import r2_score
print('R-squared Score  : ',r2_score(test_y,pred_y))
```

```
R-squared Score  :  0.9953253710579897
```

The RMSE score is 1.276, which indicates that the predicted values for Close/Last match the actual values in the dataset (test dataset). The R^2 score is 0.99, which indicates that the model prediction line closely fits the actual line of the test dataset.

Visualization and Critical Analysis

Data visualization is performed to show a singular graph of the test, training and the whole dataset:

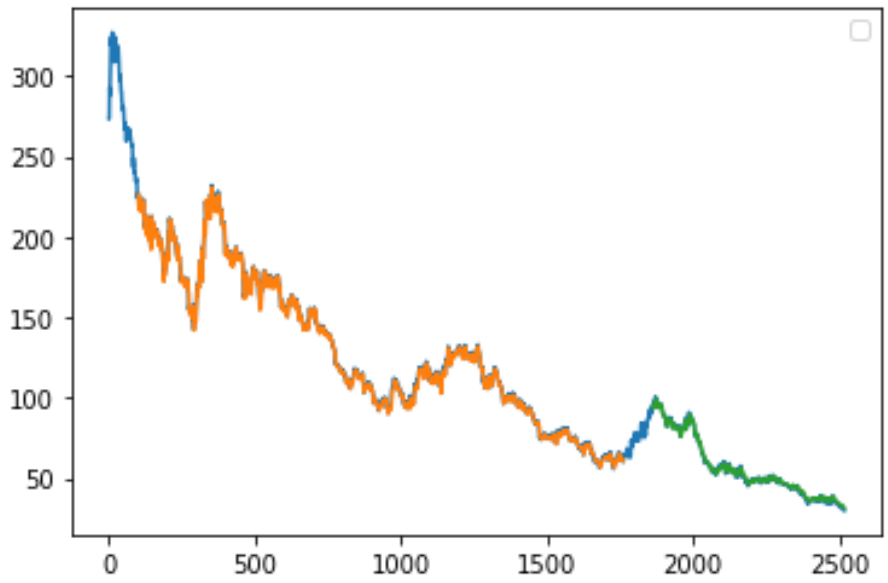
```
# shift train predictions for plotting
train_prediction = scaler.inverse_transform(model.predict(X_train))

look_back=100
trainPredict_Plot = np.empty_like(df1)
trainPredict_Plot[:, :] = np.nan
trainPredict_Plot[look_back:len(train_prediction)+look_back, :] = train_prediction

# shift test predictions for plotting
testPredict_Plot = np.empty_like(df1)
testPredict_Plot[:, :] = np.nan
testPredict_Plot[len(train_prediction)+(look_back*2)+1:len(df1)-1, :] = pred_y

# plotting of baseline and predictions of testing and training dataset
plt.plot(scaler.inverse_transform(df1))
plt.plot(trainPredict_Plot)
plt.plot(testPredict_Plot)
plt.legend()
plt.show()
```

The output of the graph is shown below:



Next, with the usage of LSTM, a 30-day prediction of the price for Close/Last is performed. This can be done by calculating the value for each day, and then calculating the values for 99 more days (total of 100 days). The iteration of 100 is pivotal for the prediction of the Close/Last values for a period of 30-day, to increase accuracy of the prediction. The following code is used to produce the iteration of 100:

```
x_input=data_test[len(data_test)-100:].reshape(1,-1)
x_input.shape
```

```
(1, 100)
```

```
input_temp=list(x_input)
input_temp=input_temp[0].tolist()
```

Input_temp are the temporary values in the dataset, for iteration of 100.

```
from numpy import array

lst_output=[]
n_steps=100
i=0
while(i<30):

    if(len(input_temp)>100):
        print(input_temp)
        x_input=np.array(input_temp[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        input_temp.extend(yhat[0].tolist())
        input_temp=input_temp[1:]
        print(input_temp)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        input_temp.extend(yhat[0].tolist())
        print(len(input_temp))
        lst_output.extend(yhat.tolist())
        i=i+1


print(lst_output)
```

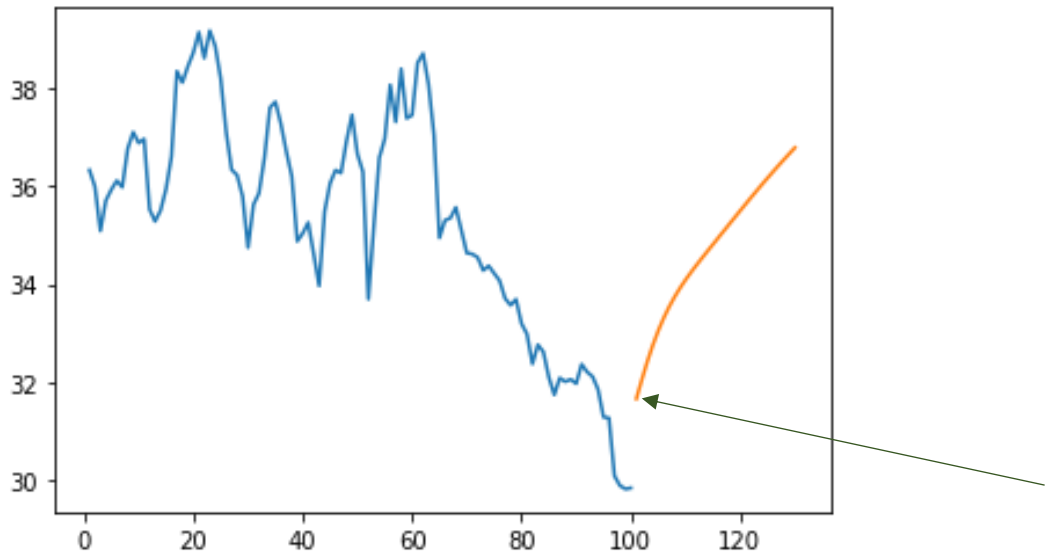
Next, 2 variables called new_day and pred_day are created. The new_day variable creates a whole new day of dataset, considering day 1 to 101 (100 days) of the previously calculated values, and the pred_day takes the following 30 days (101 to 131 days) for the predicted value of Close/Last for 30 days:

```
new_day=np.arange(1,101)
pred_day=np.arange(101,131)
```

Finally, the plotting of the graph showing the 30-day predicted values are:

```
plt.plot(new_day,scaler.inverse_transform(df1[len(df1)-100:]))
plt.plot(pred_day,scaler.inverse_transform(lst_output))
```

The output is shown below:



The output shows that the predicted values for 30-days match that of the new_day dataset, as shown by the yellow line for the predicted values. The predicted values follow the same trend as the new_day data trend. Hence, it can be concluded that the stacked LSTM model used for this assignment to predict the stock price of Apple Inc., can predict the trends of stock market, for any dataset.

References for Part 2

1. Stock Price Prediction And Forecasting Using Stacked LSTM- Deep Learning. (2020, May 25). [Video]. YouTube. https://www.youtube.com/watch?v=H6du_pfuznE&t=1902s

2. S. (2020, December 31). Apple_Stock_prediction. Kaggle. https://www.kaggle.com/sarthak1799/apple-stock-prediction/data

3. Sharma, P. (2021, October 13). Stock Market Prediction | Machine Learning for Stock Market Prediction. Analytics Vidhya. Retrieved December 2, 2021, from https://www.analyticsvidhya.com/blog/2021/10/machine-learning-for-stock-market-prediction-with-step-by-step-implementation/

4. Biswal, A. (2021, September 16). An Easy Guide to Stock Price Prediction Using Machine Learning. Simplilearn.Com. https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning