Stock Market Prediction using Machine Learning

**Submitted to-**

**REGex Software Services**

**(Submitted towards the partial fulfilment of the internship training during batch**

**Jul-Oct 2021)**

**Submitted by-**

**GROUP-13 :-**

1. **SURAJ KUMAR ROY**
2. **ARPITA MAHARATHA**
3. **VENU GOPAL REDDY**
4. **NILAVRA SARKAR**
5. **CHETAN SHUKLA**

**INTRODUCTION:**

Market prediction offers great profit avenues and is a fundamental stimulus for most researchers in this area. To predict the market, most researchers use either technical or fundamental analysis. Technical analysis focuses on analyzing the direction of prices to predict future prices, while fundamental analysis depends on analyzing unstructured textual information like financial news and earning reports. More and more valuable market information has now become publicly available online. This draws a picture of the significance of data mining strategies to extract significant information to analyze market behaviour. There have been attempts to predict stock prices using time series analysis algorithms, though they still cannot be used to place bets in the real market.In this paper, LSTM (Long Short Term Memory) model has been used which is an artificial recurrent neural network (RNN) architecture used in the field of deep learning.

**What is STOCK MARKET?:**

A stock market is a public market where you can buy and sell shares for publicly listed companies. The stocks, also known as equities, represent ownership in the company. The stock exchange is the mediator that allows the buying and selling of shares.

**Importance of Stock Market**

* Stock markets help companies to raise capital.
* It helps generate personal wealth.
* Stock markets serve as an indicator of the state of the economy.
* It is a widely used source for people to invest money in companies with high growth potential.

Stock price analysis has been a critical area of research and is one of the top applications of machine learning. In this paper we have discussed how to perform stock price prediction using [machine learning](https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article) and [deep learning techniques](https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-deep-learning). Here, we will use an LSTM network to train your model with stocks data.

**LSTM Recurrent Neural Network:**

**Long-Short-Term Memory** Recurrent Neural Network (RNN) belongs to the family of deep learning algorithms. It is a recurrent network because of the feedback connections in its architecture. It has an advantage over traditional neural networks due to its capability to process the entire sequence of data. Its architecture comprises the***cell***, ***input gate***, ***output gate*** and ***forget gate***.

The cell remembers values over arbitrary time intervals, and the three gates regulate the flow of information into and out of the cell. The cell of the model is responsible for keeping track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell, and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

However, there are some variants of the LSTM model such as Gated Recurrent Units (GRUs) that do not have the output gate. LSTM Networks are popularly used on time-series data for classification, processing, and making predictions. The reason for its popularity in time-series application is that there can be several lags of unknown duration between important events in a time series.

**Stock Prediction**:

Trying to predict how the securities exchange will work is one of the most difficult tasks. There are so many variables involved with the expectation – physical elements versus psychological factors, rational and irrational behaviour, and so on.All of these factors combine to make share costs unpredictable and difficult to predict with any degree of certainty.

In this task, the future stock prices are predicted using the LSTM Recurrent Neural Network. Our task is to predict stock prices for next few days, which is a time series problem. The LSTM model is very popular in time-series forecasting, and this is the reason why this model is chosen in this task. The historical prices are collected. We have used 35 years of historical price data, from 13.03.1986 to 03.05.2021.

This data set contains 1483 observations with 12 attributes. After preprocessing, only dates and OHLC (Open, High, Low, Close) columns, a total of 5 columns, are taken as these columns have main significance in the dataset. The LSTM model is trained on this entire dataset, and for the testing purpose, a new dataset is fetched for first 28 years. The stock prices for this new duration will be predicted by the already trained LSTM model, and the predicted prices will be plotted against the original prices to visualise the model’s accuracy.

**Data Flow Diagram:**

MSFT Dataset

**PREREQUISITE**

**Prerequisite:**

**Project Requirements:**

* Python Environment
* Anaconda
* Jupyter Notebook
* Python Libraries
  + Numpy
  + Pandas
  + Matplotlib
  + Tensorflow
  + Keras

**System Requirement**

* Windows 10 – 64 Bit
* AMD Ryzen 5 Gen 3rd
* RAM 8 GB

**Software Requirement Analysis:**

1. **Python:**

Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support functional programming and [aspect-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming). Pythonis [dynamically-typed](https://en.wikipedia.org/wiki/Type_system#DYNAMIC) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

1. **Anaconda:**

Anaconda is a [distribution](https://en.wikipedia.org/wiki/Software_distribution) of the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) and [R](https://en.wikipedia.org/wiki/R_(programming_language)) programming languages for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) ([data science](https://en.wikipedia.org/wiki/Data_science), [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications, large-scale [data processing](https://en.wikipedia.org/wiki/Data_processing), [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics), etc.), that aims to simplify [package management](https://en.wikipedia.org/wiki/Package_management) and [deployment](https://en.wikipedia.org/wiki/Deployment_environment). The distribution includes data-science packages suitable for [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [Linux](https://en.wikipedia.org/wiki/Linux), and [MacOS](https://en.wikipedia.org/wiki/MacOS).The big difference between conda and the [pip package manager](https://en.wikipedia.org/wiki/Pip_(package_manager)) is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

1. **Jupyter Notebook:**

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. In other words, Jupyter Notebook is an open-source, web-based IDE with deep cross-language integration that allows you to create and share documents containing live code, equations, visualizations, and narrative text.

1. **Numpy:**

**Numpy**is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.  
Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

1. **Pandas:**

**Pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the [Python](https://www.python.org/) programming language. Pandas is a Python package providing fast, flexible and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis is python.

1. **Matplotlib:**

Matplotlib is quite possibly the simplest way to plot data in Python. It is similar to plotting in MATLAB, allowing users full control over fonts, line styles, colours, and axes properties. This allows for complete customization and fine control over the aesthetics of each plot, albeit with a lot of additional lines of code.

1. **Tensorflow:**

TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

1. **Keras:**

Keras is one of the leading high-level neural networks APIs. It is written in Python and supports multiple back-end neural network computation engines.The biggest reasons to use Keras stem from its guiding principles, primarily the one about being user friendly. Beyond ease of learning and ease of model building, Keras offers the advantages of broad adoption, support for a wide range of production deployment options, integration with at least five back-end engines (TensorFlow, CNTK, Theano, MXNet, and PlaidML), and strong support for multiple GPUs and distributed training.

**Code Snippets and Outputs:**

1. **Importing Libraries**

*import numpy as np*

*import pandas as pd*

*from sklearn.preprocessing import MinMaxScaler*

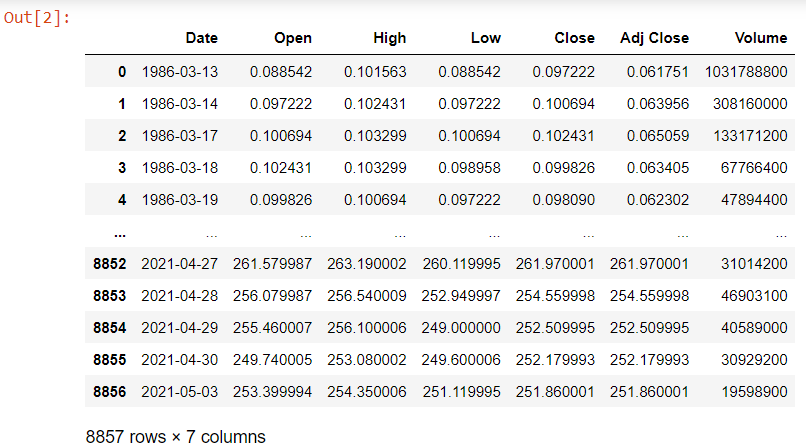
*from keras.models import Sequential*

*from keras.layers import Dense, LSTM*

*import matplotlib.pyplot as plt*

1. **Data Reading**

*df=pd.read\_csv('MSFT.csv')*



1. **Data Visualisation**

*plt.figure(figsize=(16,8))*

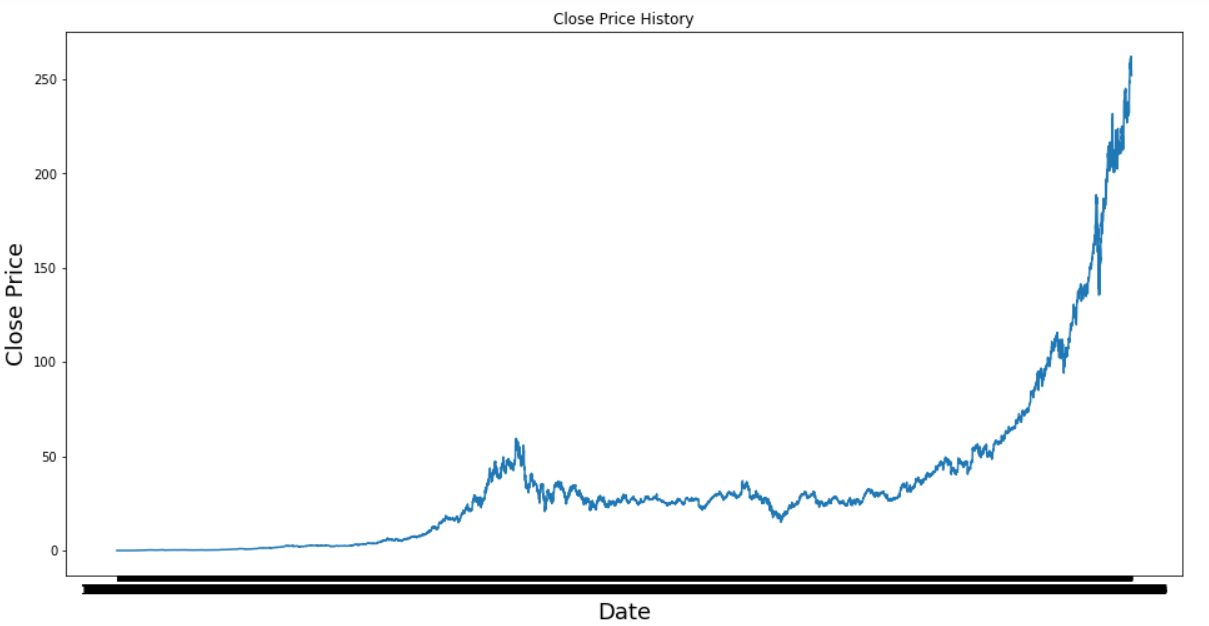
*plt.title("Close Price History")*

*plt.plot(df["Close"])*

*plt.xlabel('Date',fontsize=18)*

*plt.ylabel('Close Price',fontsize=18)*

*plt.show()*



1. **Data Scaling**

*scaler = MinMaxScaler(feature\_range=(0,1))*

*scaled\_data = scaler.fit\_transform(dataset)*

*scaled\_data*

*OUT:* array([[2.65159923e-05],

[3.97739884e-05],

[4.64067989e-05],

...,

[9.63876521e-01],

[9.62616393e-01],

[9.61394489e-01]])

1. **Train-Test Split**

*train\_data = scaled\_data[0:train\_len,:]*

*x\_train=[]*

*y\_train=[]*

*for i in range(60, len(train\_data)):*

*x\_train.append(train\_data[i-60:i,0])*

*y\_train.append(train\_data[i,0])*

*if i<=60:*

*print(x\_train)*

*print(y\_train)*

*print()*

**OUT:** [array([2.65159923e-05, 3.97739884e-05, 4.64067989e-05, 3.64594894e-05,

2.98304913e-05, 1.98869942e-05, 9.94349711e-06, 0.00000000e+00,

6.62899807e-06, 1.65724952e-05, 2.32014933e-05, 1.98869942e-05,

1.65724952e-05, 1.98869942e-05, 2.32014933e-05, 2.32014933e-05,

1.65724952e-05, 1.98869942e-05, 2.65159923e-05, 2.98304913e-05,

3.64594894e-05, 3.97739884e-05, 3.97739884e-05, 5.30357969e-05,

5.63502960e-05, 4.30922998e-05, 4.30922998e-05, 3.64594894e-05,

3.81167247e-05, 7.62372902e-05, 1.02757095e-04, 1.06071594e-04,

9.28097854e-05, 8.28662882e-05, 7.62372902e-05, 7.62372902e-05,

7.29227911e-05, 7.62372902e-05, 7.62372902e-05, 7.95517892e-05,

7.62372902e-05, 7.95517892e-05, 8.28662882e-05, 7.95517892e-05,

7.95517892e-05, 8.28662882e-05, 7.62372902e-05, 7.29227911e-05,

6.62937931e-05, 6.62937931e-05, 6.62937931e-05, 7.95517892e-05,

9.28097854e-05, 1.02757095e-04, 1.19329590e-04, 1.06071594e-04,

1.06071594e-04, 1.02757095e-04, 1.09386093e-04, 1.09386093e-04])]

[8.94952863201091e-05]

*test\_data = scaled\_data[train\_len-60:,:]*

*x\_test = []*

*y\_test = dataset[train\_len:, :]*

*for i in range(60, len(test\_data)):*

*x\_test.append(test\_data[i-60:i,0])*

1. **Model Training**

*model = Sequential()*

*model.add(LSTM(50, return\_sequences=True, input\_shape=(x\_train.shape[1],1)))*

*model.add(LSTM(50, return\_sequences=False))*

*model.add(Dense(25))*

*model.add(Dense(1))*

*model.compile(optimizer='adam', loss='mean\_squared\_error')*

*model.fit(x\_train, y\_train, batch\_size=64, epochs=1)*

1. **Testing and Error Evaluation**

*predictions = model.predict(x\_test)*

*predictions = scaler.inverse\_transform(predictions)*

*rmse = np.sqrt(np.mean(predictions-y\_test)\*\*2)*

*rmse*

OUT: 2.5262526238745013

1. **Prediction Visualisation**

*train = data[:train\_len]*

*valid = data[train\_len:]*

*valid['Predictions'] = predictions*

*plt.figure(figsize=(16,8))*

*plt.title('Model')*

*plt.xlabel('Date', fontsize=18)*

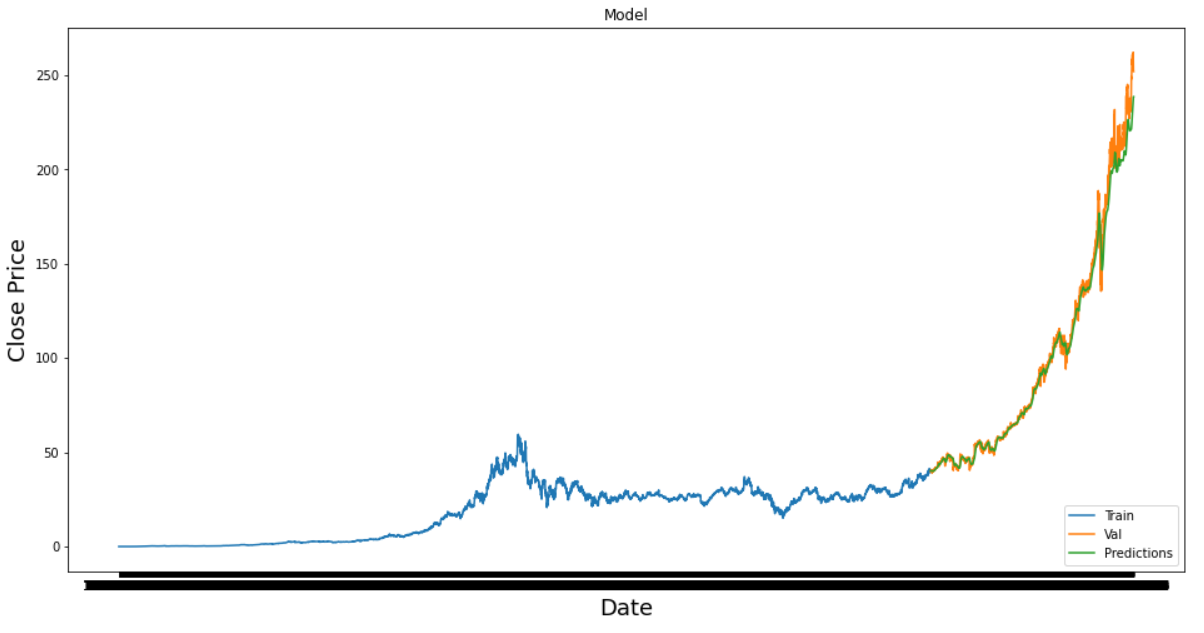
*plt.ylabel('Close Price',fontsize=18)*

*plt.plot(train['Close'])*

*plt.plot(valid[['Close','Predictions']])*

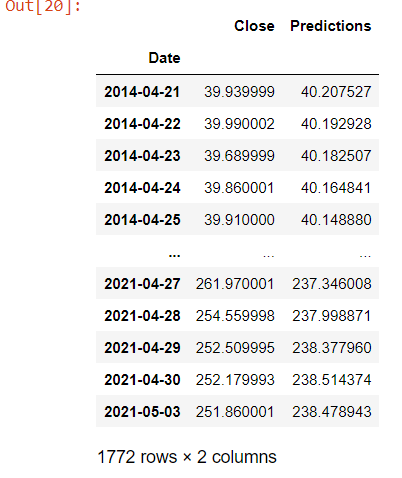
*plt.legend(['Train','Val','Predictions'], loc='lower right')*

*plt.show()*



1. **Comparing Real and Predicting Values**

*valid*



**Conclusion**

In Stock Market Prediction, we worked on predicting the future value of the financial stocks of a company taking its previous historical stock data into consideration. And as mentioned, the root mean square error has come out to be 2.526 which is actually a pretty good result. This conveys that the model has been trained efficiently which can also be visible in the comparison table mentioned above.

The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. Machine learning itself employs different models to make prediction easier and authentic. The paper focuses on the use of LSTM based Machine learning to predict stock values. Factors considered are open, close, low, high and volume.