



Stock prediction

[machine learning ::linear regression &
long short term memory]

BY::

:SHYAM.M [411621104050]

:TANUJ KUMAR [411621104052]

:POOVA RAGAN[411621104040]

: VADDIBOINA SANTHOSH [411621104054]

: SHIBYA VISVAISWARAN.S [411621104049]

Stock Market Prediction

Abstract: *The Stock Market is a challenging forum for investment and requires immense brainstorming before one shall put their hard earned money to work. This project aims at processing large volumes of data and running comprehensive regression algorithms on the dataset; that will predict the future value of a stock using the regression model with the highest accuracy. The purpose of this paper is to analyze the shortcomings of the current system and building a time-series model that would mitigate most of them by implementing more efficient algorithms. Using this model, anyone can monitor the preferred stock that they want to invest in; and maximize profit by purchasing volume at the lowest price and liquidating the stock when it's at its highest.*

Keywords: *Stock Market, Forecast, Regression, Time-Series Prediction.*

I. INTRODUCTION

Stocks form the corner-stone of any business portfolio and may be purchased privately, or from public forums. Any such transaction must conform to legal norms that have been established by the government; in order to prevent illegal practices. *"A stock (also known as "shares" or "equity") is a type of security that signifies proportionate ownership in the issuing corporation. This entitles the stockholder to that proportion of the corporation's assets and earnings."*

Historically, stocks have survived the relentless wrath of time, surpassing all its predecessors. Stocks can be bought at the stock exchange or from many online stock brokers.

Significant profit can be yielded from the successful prediction of a stock's future. *"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 efficient-market hypothesis indicates that the prices of stock reflect all information, available currently and hence, any changes in price that are not based on information that recently came into light, are consequently unpredictable. The people who oppose this hypothesis therefore, possess myriad methods and technologies which calculatedly allow them to gain information about the future price of stock. Stock

market price data is tediously voluminous and extremely volatile. Stock market trading is an extremely complicated and ever-changing system where people will either gain a fortune or lose their entire life savings.[11] In this work, an attempt to build a Time-Series prediction model, to predict stock prices.[11] This paper takes the current stock values from the data sets gathered. The data gathered is modelled into various sub parts or data sets which is used to train and test the algorithm. We use regression models in python or R to model the data. We run a comprehensive search algorithm on the data sets and create a summary table based on the output. We plot the values on a chart and apply regression and clustering techniques to find out the increase or decrease in price of that stock.

Based on the calculation, we extrapolate the current stock prices to generate a prediction after a given time. The model is developed using supervised machine learning algorithms. The output will be in graphical form and will change with change in dataset. We expect up to 66% in-sample accuracy and 35% out-of-sample accuracy using supervised machine learning algorithms on prediction model. This will enable the user to take better decisions while investing.

II. LITERATURE SURVEY

The Stock Market of a country is considered an accurate reflection of its economic prowess. Stock Market prices are ever-changing as they are prominently affected by the ebb and flow of finances throughout various economic domains. Prices of stock are invariably dependent on the demand and supply curve, that is a fundamental rule of economics. An increase in demand of a particular stock, will increase its price whereas a decrease in popularity of another, will reduce its price. Even though this fluctuation in stock prices is necessary to reap profits from investment that have been made, it is of paramount importance that we predict the future value of a stock so that loss on investments is mitigated. This review is done in order to predict the prices of stock, so as to make more informed investments.

Recently occurring trends in a market is studied and different types of machine learning classifiers and regression techniques are applied on them. Various approaches [Fig. 1] and results from past studies are weighted based on various parametrics; [Table 1] and then is displayed in a graphical format. The survey brings to light, different conventional approaches to stock market prediction, that has already been built. In addition to that, it discusses recent application of machine learning techniques along with strengths and weaknesses of each technique for effective prediction of stock prices, in the future.[12]

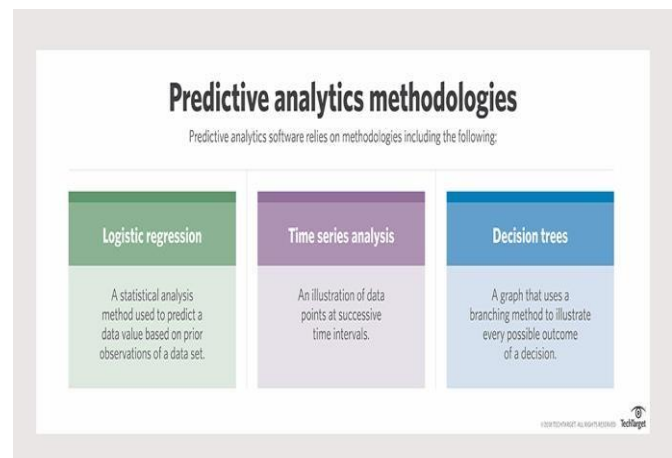


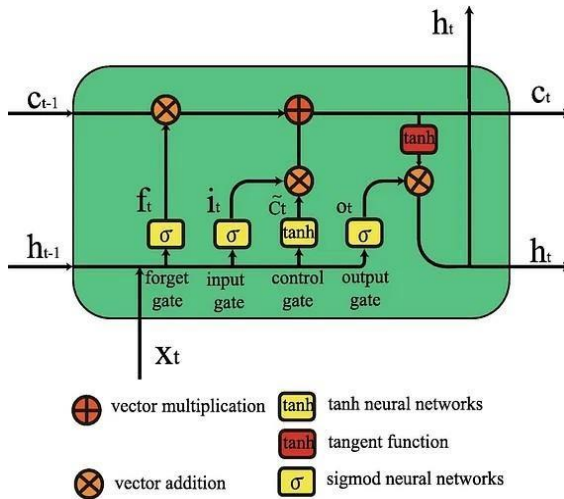
Fig. 1: Predictive Analysis Methodologies[13]

LSTM(LONG SHORT TERM MEMORY)

Understanding Long Short Term Memory Network for Stock Price Prediction

LSTM is a Recurrent Neural Network that works on data sequences, learning to retain only relevant information from a time window. New information the network learns is added to a “memory” that gets updated with each timestep based on how significant the new sample seems to the model. Over the years, LSTM has revolutionized speech and handwriting recognition, language understanding, forecasting, and several other applications that have become the new normal today.

A standard LSTM cell comprises of three gates: the input, output, and forget gate. These gates learn their weights and determine how much of the current data sample should be remembered and how much of the past learned content should be forgotten.



As seen in the equations below, i , f , and o represent the three gates: input, forget, and output. C is the cell state that preserves the learned data, which is given as output h . All of this is computed for each timestamp t , considering the learned data from timestamp $(t-1)$.

$$\begin{aligned} i_t &= \sigma(x_t U^i + h_{t-1} W^i) \\ f_t &= \sigma(x_t U^f + h_{t-1} W^f) \\ o_t &= \sigma(x_t U^o + h_{t-1} W^o) \\ \tilde{C}_t &= \tanh(x_t U^g + h_{t-1} W^g) \\ C_t &= \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \\ h_t &= \tanh(C_t) * o_t \end{aligned}$$

The forget gate decides what information and how much of it can be erased from the current cell state, while the input gate decides what will be added to the current cell state. The output gate, used in the final equation, controls the magnitude of output computed by the first two gates.

So, as opposed to standard feed-forward neural nets, LSTMs have the potential to remember or erase portions of the past data windows actively. Its feature of reading and training on windows (or timesteps) of data makes its training unique

Evaluating Prediction Performance for Stock Price Prediction

Before putting the algorithms into practice, let's clarify the metric to measure the performance of our models. Stock price prediction being a fundamental regression problem, we can use RMSE (Root Mean Squared Error) or MAPE (Mean Absolute Percentage Error) to measure how close or far off our price predictions are from the real world.

Looking closely at the formula of RMSE, we can see how we will be able to consider the difference (or error) between the actual (A_t) and predicted (F_t) price values for all N timestamps and get an absolute measure of error.

$$RMSE = \sqrt{\frac{1}{N} * \sum_{t=1}^N (A_t - F_t)^2}$$

On the other hand, MAPE looks at the error concerning the true value – it will measure relatively how far off the predicted values are from the truth instead of considering the actual difference. This is a good measure to keep the

error ranges in check if we deal with too large or small values. For instance, RMSE for values in the range of 10e6 might blow out of proportion, whereas MAPE will keep error in a fixed range

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

Stock Market

Prediction using

Machine Learning

Project Code

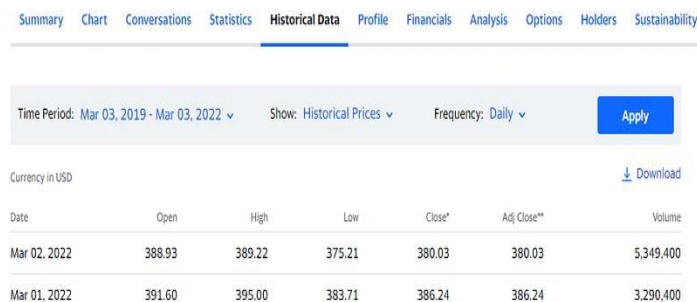
First, we will implement a simple LSTM network using [Keras](#) in Python. Let's take a look at the Stock Prediction using Machine Learning dataset. We can work on actual stock data from major public companies such as Facebook, Microsoft, or Apple by simply downloading the data from finance.yahoo.com.

Downloading the Stock Prices Dataset for Project on Machine Learning Price Prediction of Stocks

Go to finance.yahoo.com/ and search the company whose data you want to seek for stock price prediction. For our example, we will look at the Netflix (NFLX) stock over 3 years.

Going

to finance.yahoo.com/quote/NFLX/history?p=NFLX in the "Historical Data" section, we see the stock data listed each day. We can filter out the time for which we wish to analyze and download the CSV file using the download button on the right.



Date	Open	High	Low	Close*	Adj Close**	Volume
Mar 02, 2022	388.93	389.22	375.21	380.03	380.03	5,349,400
Mar 01, 2022	391.60	395.00	383.71	386.24	386.24	3,290,400

The download CSV file will contain the data for Open, High, Low, Close, Adj Close, Volume for each date, as shown in the image above.

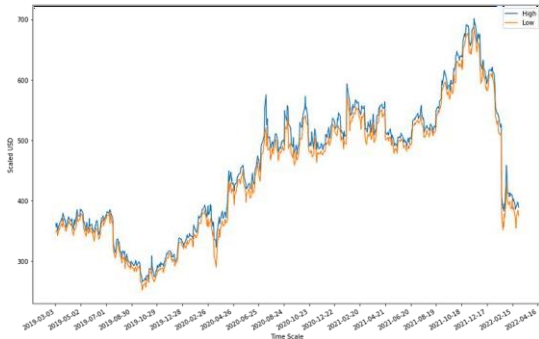
Loading the Stock Prices Dataset

Load the CSV file as a DataFrame using [Pandas](#). Since the data is indexed by date (each row represents data from a different date), we can also index our DataFrame by the date column. We have taken the data from March 2019 to March 2022. This will also challenge our model to work with the unpredictable changes caused by the COVID-19 pandemic.

```
1 import pandas as pd
2 stock_data = pd.read_csv('./NFLX.csv', index_col='Date')
3 stock_data.head()
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2019-03-04	359.720001	362.250000	348.040009	351.040009	351.040009	7487000
2019-03-05	351.459991	356.170013	348.250000	354.299988	354.299988	5937800
2019-03-06	353.600006	359.880005	351.700012	359.609985	359.609985	6211900
2019-03-07	360.160004	362.859985	350.500000	352.600006	352.600006	6151300
2019-03-08	345.750000	349.920013	342.470001	349.600006	349.600006	6898800

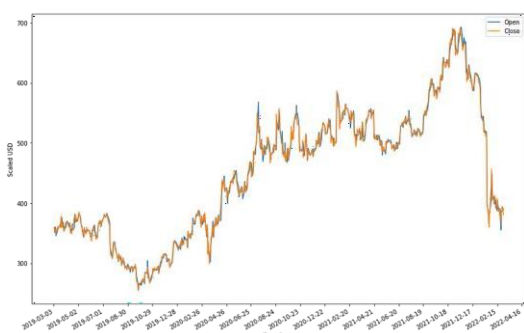
Plotting the High and Low points of Netflix stock over 3 years, we see the below graph.



As noticeable, around March 2020, we see a sudden drop in the price, after which it reports steady growth until recently.

It will be challenging for a model in the stock prediction using machine learning project to correctly estimate the rapid changes that we can see in March 2020 and February 2022. We will focus on evaluating the model performance in predicting the more recent values after training it on the past data.

Similarly, plotting the Open and Close value of the stock for each day gives equivalent observations.



The code for plotting these graphs is as shown below. We use matplotlib to plot the DataFrame columns directly against the Date index column. To make things flexible while plotting against dates, lines 6-8 convert our date strings into datetime format and plot them cleanly and legibly. The interval parameter in line 7 defines the interval in days between each tick on the date axis.

```
1 import matplotlib.dates as mdates
2 import matplotlib.pyplot as plt
3 import datetime as dt
4
5 plt.figure(figsize=(15,10))
6 plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
7 plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=60))
8 x_dates = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in stock_data.index.values]
9
10 plt.plot(x_dates, stock_data['High'], label='High')
11 plt.plot(x_dates, stock_data['Low'], label='Low')
12 plt.xlabel('Time Scale')
13 plt.ylabel('Scaled USD')
14 plt.legend()
15 plt.gcf().autofmt_xdate()
16 plt.show()
```

We will use the Open, High, and Low columns to predict the Closing value of the Netflix stock for the next day.

Importing the Libraries for Stock Price Prediction

Project

We will be building our [LSTM models](#) using Tensorflow Keras and preprocessing our stock prediction machine learning data using scikit-learn. These imports are used in different steps of the entire process, but it is good to club these statements together. Whenever we wish to import something new, just add the statement arbitrarily to the below group.

```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import numpy as np
4 from tensorflow.keras.models import Sequential
5 from tensorflow.keras.layers import Dense
6 from tensorflow.keras.layers import LSTM
7 from tensorflow.keras.layers import Dropout
8 from tensorflow.keras.layers import *
9 from tensorflow.keras.callbacks import EarlyStopping
10
11 from sklearn.preprocessing import MinMaxScaler, StandardScaler
12 from sklearn.metrics import mean_squared_error
13 from sklearn.metrics import mean_absolute_percentage_error
14 from sklearn.model_selection import train_test_split
15 from sklearn.model_selection import TimeSeriesSplit
16 from sklearn.metrics import mean_squared_error
```

Data Preprocessing for Stock Market Prediction using Machine Learning

As with any other machine learning model, it is always good to normalize or rescale the data within a fixed range when dealing with real data.

This will avoid features with larger numeric values to unjustly interfere and bias the model and help achieve rapid convergence in the machine learning stock prediction project.

First, we define the features and the target as discussed above.

```
1 target_y = stock_data['Close']
2 X_feat = stock_data.iloc[:,0:3]
```

Next, we use a StandardScaler to rescale our values between -1 and 1.

```
1 #Feature Scaling
2 sc = StandardScaler()
3 X_ft = sc.fit_transform(X_feat.values)
4 X_ft = pd.DataFrame(columns=X_feat.columns,
5                       data=X_ft,
6                       index=X_feat.index)
```

Scikit-learn also provides a popular MinMaxScaler preprocessing module. However, considering the context, stock prices might max out or minimise on different days, and using those values to influence others might not be great. The change in values from using either of these methods would not be much, so we stick to StandardScaler.

We have 757 data samples in the dataset.

So, the next step would be to split it into training and testing sets. As explained above, the training of an LSTM model requires a window or a timestep of data in each training step. For instance, the LSTM will take 10 data samples to predict the 10th one by weighing the first nine input samples in one step. So, we need a different approach than the train_test_split provided by scikit-learn.

Let's define a splitting function called `lstm_split()` which will make windows of size "n_steps" starting from the first sample of data and ending at n_steps'th sample (if n_steps=10, then the 10th sample) from the end. We understand the latter part because, for each time step, LSTM will take n_steps-1 samples for training and predict the last sample. Loss calculation is done based on the error in this prediction. So if n_steps=10, you cannot use the last 9 samples to predict anything because the "10th" data point for the current step does not exist in the dataset.

The function below takes the entire data and creates windows of size n_steps starting from the beginning. The target y will contain the target value corresponding to the n_steps'th index. So if n_steps is 10, the first element in X will have features from 10 data samples, and y will contain the target of the 10th data sample.

```
1 def lstm_split(data, n_steps):
2     X, y = [], []
3     for i in range(len(data)-n_steps+1):
4         X.append(data[i:i + n_steps, :-1])
5         y.append(data[i + n_steps-1, -1])
6
7     return np.array(X), np.array(y)
```

Train and Test Sets for Stock Price Prediction Project

We split our data into training and testing sets. Shuffling is not permitted in time-series datasets. In the beginning, we take two steps worth of past data to predict the current value. Thus, the model will look at yesterday's and today's values to predict today's closing price.

```
1 X1, y1 = lstm_split(stock_data_ft.values, n_steps=2)
2
3 train_split=0.8
4 split_idx = int(np.ceil(len(X1)*train_split))
5 date_index = stock_data_ft.index
6
7 X_train, X_test = X1[:split_idx], X1[split_idx:]
8 y_train, y_test = y1[:split_idx], y1[split_idx:]
9 X_train_date, X_test_date = date_index[:split_idx], date_index[split_idx:]
10
11 print(X1.shape, X_train.shape, X_test.shape, y_test.shape)
```

(755, 2, 3) (604, 2, 3) (151, 2, 3) (151,)

Note above that the size of X1 is n_steps less than that of the original dataset. As we explained above, you cannot use the last two samples of the original set during training or prediction as we do not have their corresponding ground truth values.

Stock Prediction Machine Learning Project- Building the LSTM model

We will use the Sequential and LSTM modules provided by Tensorflow Keras to build a simple, single-unit LSTM model.

```
1 lstm = Sequential()
2 lstm.add(LSTM(32, input_shape=(X_train.shape[1], X_train.shape[2]),
3     activation='relu', return_sequences=True))
4 lstm.add(Dense(1))
5 lstm.compile(loss='mean_squared_error', optimizer='adam')
6 lstm.summary()
```

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(None, 1, 32)	4608
dense_2 (Dense)	(None, 1, 1)	33
Total params: 4,641		
Trainable params: 4,641		
Non-trainable params: 0		

Now we can fit this simple model to the training data.

```
1 history=lstm.fit(X_train, y_train,  
2 | | | | | epochs=100, batch_size=4,  
3 | | | | | verbose=2, shuffle=False)
```

```
Epoch 1/100  
152/152 - 2s - loss: 0.5510 - 2s/epoch - 16ms/step  
Epoch 2/100  
152/152 - 1s - loss: 0.1016 - 829ms/epoch - 5ms/step  
Epoch 3/100  
152/152 - 0s - loss: 0.0124 - 447ms/epoch - 3ms/step  
Epoch 4/100  
152/152 - 0s - loss: 0.0122 - 313ms/epoch - 2ms/step
```

Given the simplicity of the model and the data, we note that the loss reduction stagnates after only 20 epochs. You can observe this by plotting the training loss against the number of epochs, and LSTM does not learn much after 10-20 epochs.

