## 

**TITLE**

**Stock Prediction using basic RNN**

**Submitted to: Mr. Sagar Pandey**

|  |  |  |
| --- | --- | --- |
| Name | Registration Number | Roll Number |
| Anuj Pundir | 11811882 | A26 |

**Table of contents**

[ Introduction: 4](#_Toc36060926)

[ Objective: 4](#_Toc36060927)

[ Motivation: 4](#_Toc36060928)

[ Implementation of the project 5](#_Toc36060929)

[ Output: 7](#_Toc36060930)

[ Scope: 7](#_Toc36060931)

[ Work Distribution: 8](#_Toc36060932)

[ Libraries Used: 8](#_Toc36060933)

[ GitHub link: 8](#_Toc36060934)

[ References: 8](#_Toc36060935)

**Abstract**

Long Short Term Memory cells are like mini neural networks designed to allow for memory in a larger neural network. This is achieved through the use of a recurrent node inside the LSTM cell. This node has an edge looping back on itself with a weight of one, meaning at every feedfoward iteration the cell can hold onto information from the previous step, as well as all previous steps. Since the looping connection’s weight is one, old memories won’t fade over time like they would in traditional RNNs. LTSMs and recurrent neural networks are as a result good at working with time series data thanks to their ability to remember the past. By storing some of the old state in these recurrent nodes, RNNs and LSTMs can reason about current information as well as information the network had seen one, ten or a thousand steps ago. Even better, I don’t have to write my own implementation of an LSTM cell; they’re a default layer in Tensorflow’s Keras.So I had my plan; to use LSTMs and Keras to predict the stock market.

# Introduction:

# One of the most prominent use cases of machine learning is “Fintech” (Financial Technology for those who aren't buzz-word aficionados); a large subset of which is in the stock market. Financial theorists, and data scientists for the better part of the last 50 years, have been employed to make sense of the marketplace in order to increase return on investment. However, due to the multidimensional nature of the problem, the scale of the system, and inherent variation with time, it has been an overwhelmingly tough challenge for humans to solve, even with the assistance of conventional data analytics tools. However, with the onset of recent advancements in machine learning applications, the field has been evolving to utilize non-deterministic solutions the “learn” what is going on in order to make more accurate predictions.

# Objective:

This project aims on simple stock price prediction model and exploring how “tuning” the model affects the results. This project is intended to be easy to follow, as it is an introduction. We will use LSTM and basic RNN for the completion of the project.

# Motivation:

The art of forecasting stock prices has been a difficult task for many of the researchers and analysts. In fact, investors are highly interested in the research area of stock price prediction. For a good and successful investment, many investors are keen on knowing the future situation of the stock market. Good and effective prediction systems for stock market help traders, investors, and analyst by providing supportive information like the future direction of the stock market. In this work, we present a recurrent neural network (RNN) and Long Short-Term Memory (LSTM) approach to predict stock market indices.

# Implementation of the project

To run the project make sure you run the following lines of code:

Following is the code:

* **import** **numpy** **as** **np**
* **import** **matplotlib.pyplot** **as** **plt**
* **import** **pandas** **as** **pd**
* **import** **tensorflow** **as** **tf**
* *# Importing the training set*
* dataset\_train = pd.read\_csv('/content/Google\_Stock\_Price\_Train.csv')
* training\_set = dataset\_train.iloc[:, 1:2].values
* *# Feature Scaling*
* **from** **sklearn.preprocessing** **import** MinMaxScaler
* sc = MinMaxScaler(feature\_range = (0, 1))
* training\_set\_scaled = sc.fit\_transform(training\_set)
* *# Creating a data structure with 60 timesteps and 1 output*
* X\_train = []
* y\_train = []
* **for** i **in** range(60, 1258):
* X\_train.append(training\_set\_scaled[i-60:i, 0])
* y\_train.append(training\_set\_scaled[i, 0])
* X\_train, y\_train = np.array(X\_train), np.array(y\_train)
* *# Reshaping*
* X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))
* *# Building the RNN*
* *# Initialising the RNN*
* regressor = tf.keras.models.Sequential()
* *# Adding the first LSTM layer and some Dropout regularisation*
* regressor.add(tf.keras.layers.LSTM(units = 50, return\_sequences = **True**, input\_shape = (X\_train.shape[1], 1)))
* regressor.add(tf.keras.layers.Dropout(0.2))
* *# Adding a second LSTM layer and some Dropout regularisation*
* regressor.add(tf.keras.layers.LSTM(units = 50, return\_sequences = **True**))
* regressor.add(tf.keras.layers.Dropout(0.2))
* *# Adding a third LSTM layer and some Dropout regularisation*
* regressor.add(tf.keras.layers.LSTM(units = 50, return\_sequences = **True**))
* regressor.add(tf.keras.layers.Dropout(0.2))
* *# Adding a fourth LSTM layer and some Dropout regularisation*
* regressor.add(tf.keras.layers.LSTM(units = 50))
* regressor.add(tf.keras.layers.Dropout(0.2))
* *# Adding the output layer*
* regressor.add(tf.keras.layers.Dense(units = 1))
* *# Compiling the RNN*
* regressor.compile(optimizer = 'adam', loss = 'mse',metrics=['accuracy'])
* *# Fitting the RNN to the Training set*
* regressor.fit(X\_train, y\_train, epochs = 100)
* dataset\_test = pd.read\_csv('/content/Google\_Stock\_Price\_Test.csv')
* real\_stock\_price = dataset\_test.iloc[:, 1:2].values
* *# Getting the predicted stock price of 2017*
* dataset\_total = pd.concat((dataset\_train['Open'], dataset\_test['Open']), axis = 0)
* inputs = dataset\_total[len(dataset\_total) - len(dataset\_test) - 60:].values
* inputs = inputs.reshape(-1,1)
* inputs = sc.transform(inputs)
* X\_test = []
* **for** i **in** range(60, 80):
* X\_test.append(inputs[i-60:i, 0])
* X\_test = np.array(X\_test)
* X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))
* predicted\_stock\_price = regressor.predict(X\_test)
* predicted\_stock\_price = sc.inverse\_transform(predicted\_stock\_price)
* *# Visualising the results*
* plt.plot(real\_stock\_price, color = 'red', label = 'Real Google Stock Price')
* plt.plot(predicted\_stock\_price, color = 'blue', label = 'Predicted Google Stock Price')
* plt.title('Google Stock Price Prediction')
* plt.xlabel('Time')
* plt.ylabel('Google Stock Price')
* plt.legend()
* plt.show()

# Output:



# Scope:

There are a lot of complicated financial indicators and also the fluctuation of the stock market is highly violent. However, as the technology is getting advanced, the opportunity to gain a steady fortune from the stock market is increased and it also helps experts to find out the most informative indicators to make a better prediction. The prediction of the market value is of great importance to help in maximizing the profit of stock option purchase while keeping the risk low.

# Libraries and Dataset Used:

1. Numpy
2. Matplotlib
3. Pandas
4. Tensorflow
5. Keras
6. Google stock history
7. LSTM

# GitHub link:

# <https://github.com/AnujPundir29/AI-Project--Stock-Prediction-using-basic-RNN>

# References:

<https://en.wikipedia.org/wiki/Stock_market_prediction>

<https://keras.io/layers/recurrent/#rnn>

<https://blog.usejournal.com/stock-market-prediction-by-recurrent-neural-network-on-lstm-model-56de700bff68>