Google Stock Analysis Using Machine Learning

Peiyong Lin
Oakville Trafalgar High School, Oakville L6J 3L6, Canada.

Abstract: Stock price predictions have been a big obstacle for people to tackle from the start of stock trading. One of the biggest benefits of stock price prediction is that people can use this data to get an advantage in trading by knowing the price before the other people which significantly boosts the possibility of a positive investment. But because stock price is determined by many complicated factors and some are involved in real life situations such as a break out of war or an epidemic. It is really hard to predict the actual price of the stock. In this paper, it will talk about modern machine learning algorithms that will help people to predict the stock price. It will give a model of futute price but it will not guarantee to match with the real life performance of the stock due to many unpredictable factors.

Keywords: Stock Price Prediction; Modern Machine Learning Algorithms

Machine Learning Model Introduction

LSTM(Long short-term memory): LSTM is an artificial neural network that is great to analyze and predict based on time series data. It is widely used in stock prediction and it is the most cited neural network in the 20th century. It is developed from RNNs and it is able to deal with vanishing gradient problems that can happen with RNNs.

Google Inc Stock Prediction Using LSTM Network

Key steps:

- Collecting the data
- Visualize the data
- Train the machine learning model
- Visualize the final prediction

Step 1 - Collecting the data

Yahoo finance: 5 year worth of stock data coming from Yahoo finance on Google Inc.

Step 2 - Visualize the data

Using Python's matplotlib library, we are able to visualize the high and low of google's stock data and open close data shown below

Step 3 - Training the machine learning model

Import the necessary library needed to do the job.

```
import math
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

Get the closing price from *stock_data* and convert them into a number series and scale the data using *MinMaxScaler*

```
close_prices = stock_data['Close']
values = close_prices.values
training_data_len = math.ceil(len(values)* 0.8)

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(values.reshape(-1,1))
```

Then create the train_data and split the data into *x_train* and *y_train* and convert it into numpy arrays using *np.array()* and reshape the *x_train* array into a three dimensional array using *np.reshape()*

```
train_data = scaled_data[0: training_data_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])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

Then build the LSTM model by first creating a sequential model using *keras.Sequential()* and adding four layers into the model. The first two layers will be LSTM layer and the third and fourth layer will be Densely connected layers

```
model = keras.Sequential()
model.add(layers.LSTM(100, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(layers.LSTM(100, return_sequences=False))
model.add(layers.Dense(25))
model.add(layers.Dense(1))
model.summary()
```

Train the model 3 times and gets the rmse using *numpy* and the error is about *2.37*

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

```
model.fit(x_train, y_train, batch_size= 1, epochs=3)
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
rmse = np.sqrt(np.mean(predictions - y_test)**2)
```

Doing the same process but train 100 times (RMSE: 0.0732652621)

Step 4 - Get the Accuracy of the prediction

Loss function: Root mean squared error function to predict the difference between the result and the actual value. After 3 epochs, the error is around 2.7 and after 100 epochs, the error is around 0.07. This method will always produce a positive result and a small value means a high accuracy, so in this case. It dropped from 2.7 to 0.07 which is a 2.63 difference which is proven by the graphs above.

$$RMSD(\theta) = \sqrt{MSE(\theta)} = \sqrt{E((\Theta_2 - \theta_1)^2)}$$

Figure 2 – epoch 100

Figure 3 – epoch 20

Conclusion

The model works exceptionally well with predicting stock prices but one of the major drawbacks of this neural network and all other neural networks that predict stuff faces is people can only backtest the historical data but the price movement does not necessarily follow the historical trend in various unforeseen circumstances. As stated in the Abstract, real life performance of a stock is dependent on many factors that are impossible to predict which makes predicting real life scenarios exceptionally difficult and LSTM can give a certain hits about the performance of the stock but does not necessarily represent the real performance.

References

[1] Brownlee, J. (2017, May 24). *A Gentle Introduction to Long Short-Term Memory Networks by the Experts*. Machine Learning Mastery. Retrieved August 17, 2022, from https:// machinelearningmastery. com/gentle-introduction-long-short-term-memory-networks-experts/.

[2] ProjectPro. (2022, August 21). *Stock Price Prediction using Machine Learning with Source Code*. Retrieved August 13, 2022, from https:// www. projectpro.io/ article/stock -price-prediction- using-machine-learning-project/571.

[3] Saha S. (2021, December 7). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.Medium. Retrieved September 18,2022,from https://towardsdatascience.com/a- comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

About the author: Lin Peiyong (2006.06), male, Han nationality,Beijing native,student,high school degree,Oakville Trafalgar High School,Research direction: Economics, modern machine learning algorithms.