Cixin Gao, Lize Chen, Ziyao Tang

Professor Kevin Gold

CDS DS110

9 December 2022

Use Machine Learning To Predict Stocks

Introduction:

We believe there are more or less people around you who like to invest in stocks. Maybe it is because there is no need to socialize and one can make money at home. Maybe it is because one want to take the fate of making money into his or her own hands and test the guess in the future. Maybe it is because one loves to seek excitement and keep the expectation of profit and loss. No one wants to lose money in investing in stocks. Everyone wants to make a little money from it, and some want to make a lot of money. In order to make money from investing in stocks, everyone tries to analyze the trends of the stocks.

Unfortunately, it is very difficult for individuals to figure out an accurate result and they basically rely on guesses and previous experience. However, if we have machine learning, will this matter be much less complicated? If machine learning can really help people predict stock movements, does that mean throwing all kinds of data collected into a model to build a model to predict stock prices, turning on the computer every day to run through the model, predicting the price of the next trading day, deciding how to trade today, and then the money will “walk” into one’s pockets? Based on this conjecture, we set “Use Machine Learning To Predict Stocks” as our topic.

Methodology:

- **Stock Choosing**

At first, we chose Twitter - a platform that makes it easy for people to get the latest news - considering that people will follow the application they use the most. Just as we were about to start our research, we got the news that Twitter's stock was closed. So, we reselected a new stock, Meta Platforms, Inc. (META), which is Meta's products and services include Facebook, Messenger, Facebook Watch, and Meta Portal. It has also acquired Oculus, Giphy, Mapillary, Kustomer, Presize and has a 9.99% stake in Jio Platforms.

- **Data Selection and Cleaning**

We used Yahoo Finance, which is a platform that provides historical stock data. (https://finance.yahoo.com/quote/META/history?p=META) We installed the yfinance package to download history data from the website to python. We chose to download the stock from 1, 1, 2019 to 11, 1, 2022 as the data that would be trained and tested. To directly create a table with an index of the date, we downloaded columns of the open price, highest price, lowest price, close price, adjusted close price, and volume of the stock each day, and deleted the name of the stock. For the machine learning, we only selected "Open", "High", "Low", "Close", and "Volume" columns, rounded every value into 2 decimals, and filled the black space with the nearest data on above.

For the following prediction, we decided to predict the close price, by training the stock's data from 01/01/2019 to 12/31/2021 in both methods, then testing the model using data from 01/01/2022 to 10/31/2022. After plotting the prediction and the real history data, we will compare two plots that are plotted by two methods we use and decide which method is better for META stock.

- **KNN Regressor**

The first method we used was KNN Regressor, which is a non-parametric method that approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighborhood.

We first used all the columns to make the prediction. We changed the type of the date from date time to float. Then, we separated the table into two groups for training and testing. Clearing each table for training and testing, we imported several sklearn packages that needed to be used to scale the data size from 0 to 1 by MinMaxScalar, and trained those data by KNN Regressor. By using Gridsearch, we tried to find the best parameter for KNN from 2 to 9. Last we fitted the model and made predictions for the testing part, and plotted the graph for the inversely scaled predictions and the original data.

- **LSTM**

The second method we use is LSTM, which is an artificial neural network used in the fields of artificial intelligence and deep learning.

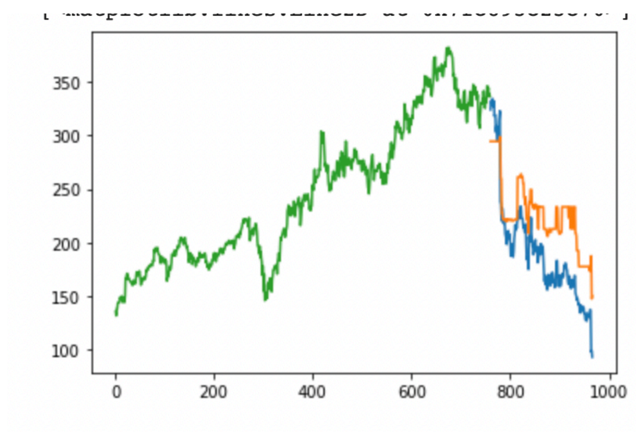
For the values preparation, we first installed and imported tensorflow, keras packages, and sklearns related packages. Then we only selected the “Close” column and the “Date” index for later learning by building up a new dataframe table. By taking the values from the table to make an array, we separated those values into two sets for training and testing based on the separation we decided above. By MinMaxScalar, we scaled the values from 0 to 1 and fitted those values.

Second, We grouped 60 values as a set and put it into X_train and Y_train, then reshaped the X_train. By using Sequential, we built up a network model that used to train the data. We added two LSTM layers of the network with the first layer setting the return_sequence to true, so that the output of the layer will be another sequence of the same length. Then we added a densely connected neural network layer with 1 network unit to specify the output. We set 'mean_squared_error' for loss function, and adopted 'adam' for optimizer. Then, we trained the model by fitting it with the training set with batch_size of 1, 3 epochs, and 2 verbose as parameters. During the process, we adjusted several parameters, like the layers of the model, dense and the units. The parameters above were the best from 8 sets we tried by using an enumeration method. Based on the technical reasons, we could not test large parameter sets.

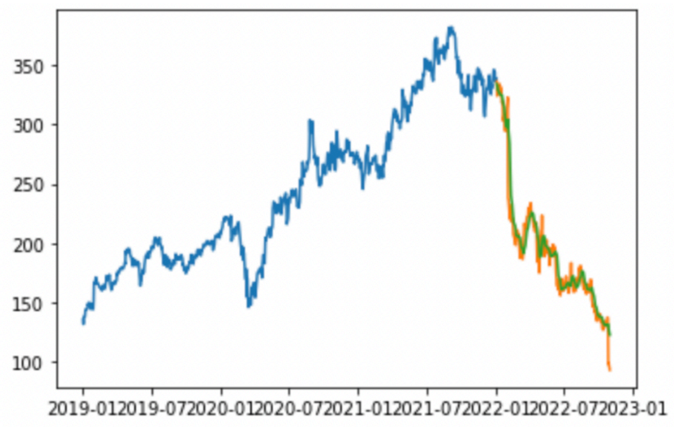
Last, we redid the same data preparation part and used the model to predict the testing part. By fitting the model and inversely scaling the values, we plotted the prediction part with the real data.

Results:

KNN-Regressor



LSTM:



After comparing these two plotted graphs and methods, we are able to find out that LSTM prediction works better than K-NN prediction. From the graph created through our program, it is not hard to see that LSTM method has a better fit. Furthermore, we considered the RMSE result as a further statistical values to prove that LSTM is better.

Conclusions:

To sum up, we could only say that the LSTM is more reliable than the K-NN estimation, but we cannot say that we could completely rely on the LSTM estimation results to buy stocks. Machine learning is certainly useful, but we need to assume that both historical data and future data obey the same distribution. This is what allows us to estimate results that are closer to what might happen in the future. Also, machine learning requires the model to be invariant, so machine learning is more effective in the area of high frequency trading because the model is more likely to remain constant in the short term. The results we estimate for this project are only idealized, but in reality there are a variety of unpredictable factors that can affect the daily stock movements. Machine learning is just a reference, it is impossible to predict exactly what will happen in the future. In other words, if machine learning could perfectly predict where the data will go in the future, it would be the standard answer that everyone thinks, while it is actually not that amazing. There is no such thing as pie in the sky, and if you rely too much on machine learning, not only will it be less helpful in buying stocks, you may lose money significantly.