McNeil, Christopher
Harda, Yazid
Adams, Ryan

Stocking on Data

GITHUB: **HTTPS://GITHUB.COM/RWADAMS/STOCKING-ON-DATA**

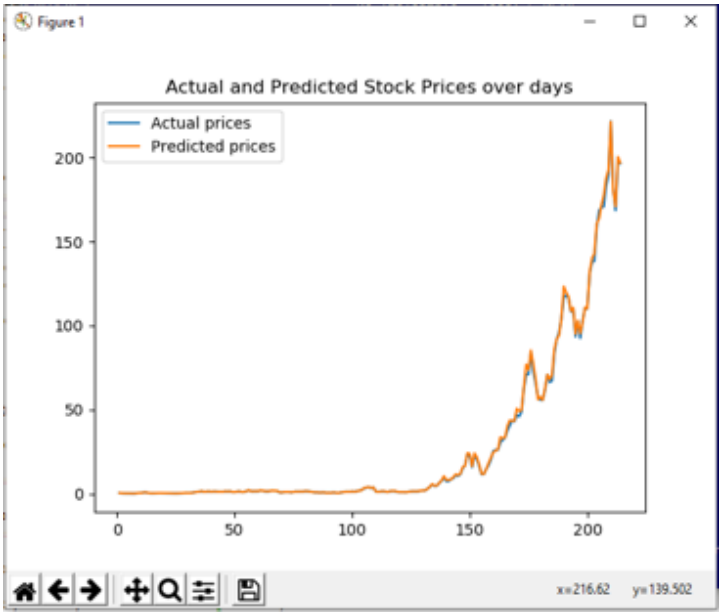
#1. Provide a description of your tool/project. What does it do? Why is it special? Is there anything out there like it? Why is the problem challenging?

For our project, our goal was to be able to predict the stock market and whether it would go up or down the next day, based upon the last ten days of stock prices. Going into this project we knew that predicting day to day fluctuations is generally considered impossible by people in finance, however we wanted to see if we could create a prediction that was better than purely guessing what a stock would do. To this end we were somewhat successful with our predictions with around 60% prediction accuracy. This shows that there is some connection between the past price and future price on a day to day level, but the connection is minimal.

Our project started with developing a script that if given an NYSE ticker, it would receive that ticker's data and create a new data set. Each dataset contained a number of ten-day periods spread across the history of the entire stock's existence. Each ten day period in the dataset, also had the average price of the period, the max price in the period, and min price of the period added. We would then look at the 11^{th} day and added a 1 if the stock went up, or a 0 if the stock stayed the same or decreased. For a company that has existed about 35 years, the created training dataset would be about ~400 ten-day periods, with a test dataset of about ~200 ten-day periods. We then used TensorFlow to create a four-layer neural network that would train the datasets that were made.

This set up worked well for when only a single stock or handful of stocks were being tested. We tried using this same methodology for testing against the whole of the NASDAQ-100. For this, the training dataset created was ~28,000 ten-day periods, and the test dataset was ~14,000 ten-day periods. This massive leap in the size of the datasets severely impacted the ability to train and test. Each epoch would take several minutes for calculating and return either very limited improvements in accuracy or no improvement in accuracy.

After the failure of testing on a large scale, we decided to return to focusing on a single stock at a time, but instead of just predicting whether the stock would go up or down, predict to what magnitude the stock would change. We used the same datasets for this prediction, except that instead of a 1 or 0, we added the actual price of the stock on the 11th day. When we tested this, we managed to correctly predict the price with an average difference of only 40 to 75 cents depending upon the stock being tested.



ACTUAL VS PREDICTED PRICE OF APPL STOCKS

If we were to continue work on this in the future, there are several things we would look to either change or expand further upon in the future. Accuracy could most likely be improved upon by adding additional data. Such data could include volatility measures, moving averages, or using the relative strength index. However much of this was not included due to much of this data being focused on computational financial analysis rather than machine learning. Another thing we might change is what data we train the neural network on. Currently we use an even spread across the entirety of a stock's history. This may lead to a skewing of data on account of general inflation. Another possible issue might be the presence of recessions that lead to outliers in the training data that don't classify correctly and decrease accuracy. Despite being only mildly successful in creating predictions, this project did provide a great opportunity into learning TensorFlow, and taught a good deal about stock market and finances.

#2. Software Requirements - Do I need to install software/packages?

We sat up an Ubuntu VM with all the packages needed for running the project. The packages are as follow: tensorflow, yfinance, numpy, pandas and sklearn.

The virtual machine is hosted on: https://drive.google.com/open?id=1mNb09CyBLiwGZ1vtS9aRa9sj3Af_1cBp

#3. Tool/Project Installation Instructions

After downloading the VM image, you need to install VirtualBox. To run the VM, import appliances in the file menu and select the one proper one. That should take care of installing the Ubuntu VM on your machine. The password to the VM is: 'root'

#4. Instructions for Generating Results

First of all, change to the working directory by typing:

cd Desktop/Stocking-on-Data/

To run the binary classifier, generate the data using the command:

python3 stock_data_script.py

Then,

python3 main.py

To pull the data for the price predictor type:

python3 stock_data_script_price.py

And display the predictions using:

python3 python3 regression_main.py

To test the algorithm with other tickers change the 'tickers' variable in either 'stock_data_script.py' or 'stock_data_script_price.py' to "GOOGL", "MSFT" or any of the tickers in the file 'nasdaq_tickers.txt' and rerun the script using:

python3 [stock_data_script.py or stock_data_script_price.py]

Change the name of the ticker in the scripts 'main.py' and 'stock_data_script_price.py' as well, specifically in both variables 'traindata' and 'testdata' and rerun the classifier or predictors accordingly.