Mario Bitar, Zi Chao Zhang

Alexandre Pronine, Andrew Stewart, Jonathon Sumner

Introduction to Computer Programming

1 June 2020

Stock Market Prediction for the S&P 500 in 2017

CONTEXT

Since the advent of the stock market, people have been trying to predict the market, and reading into the technicality of the mysterious system, in hopes of making money. However, these people all came to a single conclusion, that the market is unpredictable. For our project, we wanted to see how close we can predict the 2017 stock prices of the S&P 500, using only past data from the years prior (2013-2016). While seeming relatively trivial, we decided to figure out a way to fit an exponential to the stock data from scratch through an iterative process which is not as simple nor straightforward as it seems.

DESCRIPTION OF MODEL

If we zoom out of the S&P 500 chart all the way to its beginnings in 1950 as illustrated in Figure 1, the overall trend of the graph is to the upside and forming something like an exponential function (Harwood). Within those years, there are periods of strong economic growth and economic crashes causing the S&P 500 to trace higher or lower, but the overall exponentially growing trend is what allows us to confidently use a computational approach to predict future stock prices of the S&P 500.

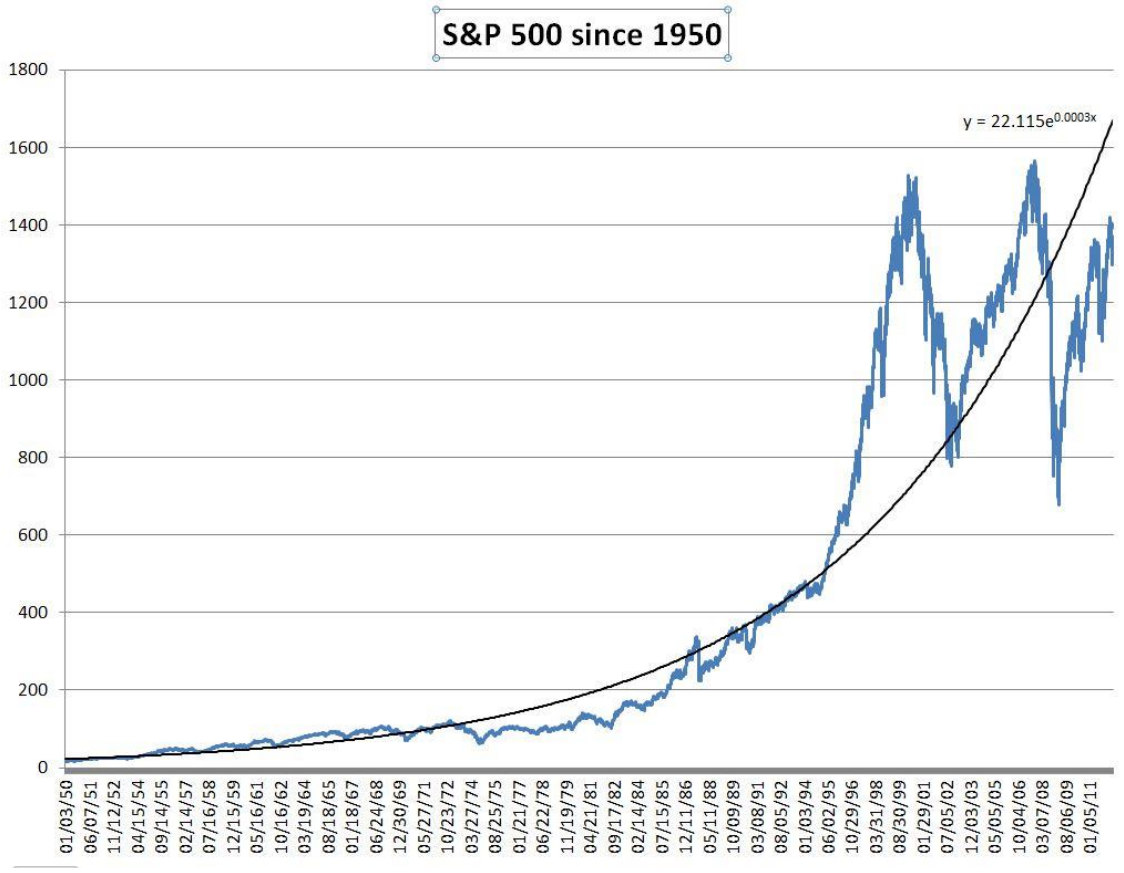


Figure 1. S&P 500 historical chart shows an overall exponential growth since 1950.

The exponential model that we will use to predict the prices of S&P 500 stocks in 2017 is based on the exponential function formula

$$f(x) = a \cdot e^{bx} + c \tag{1}$$

where a, b and c are parameters of the function, calculating the stock price $f(x)$ on a given day (x). All three parameters affect the shape of the exponential curve. In our scenario of predicting future stock prices, it is only appropriate to use positive values for these three parameters. Parameter a determines the vertical stretch or compression of the curve. For a > 1, As the parameter increases, the function curves more intensely upwards (vertical stretch), whereas for 0 < a < 1, the curve compresses (Figure 2). Parameter b determines the inclination of the curve, and since it multiplies with the independent variable in the exponent, a slight change of parameter b causes the curve to incline greatly. For that reason, we use small decimal values for this parameter. As this value gets closer to zero, e^{bx} gets closer to 1, which results in a horizontal line. Parameter C vertically shifts the graph (transition in the y-axis) (Lumen Learning).

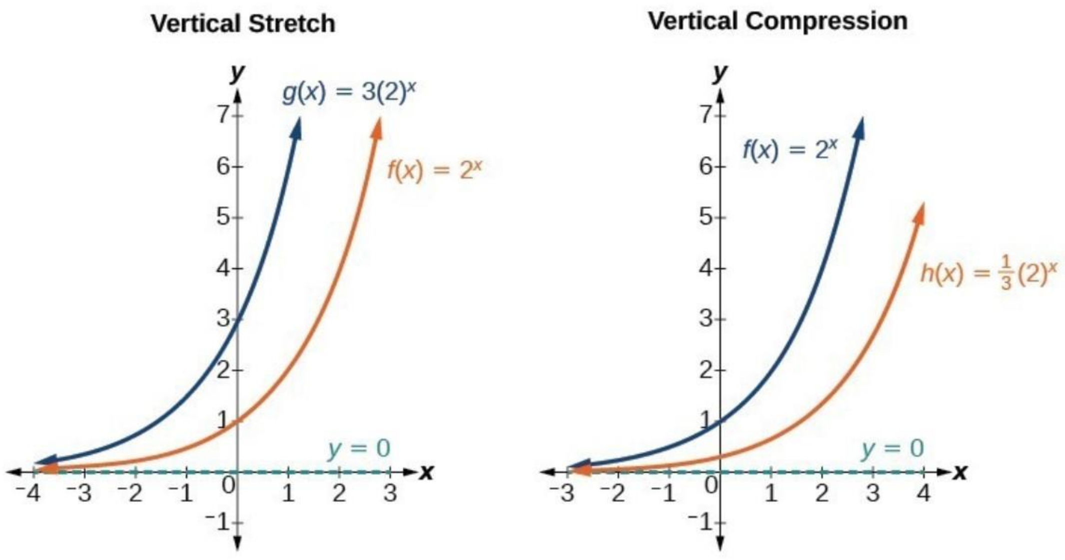


Figure 2. Vertical stretch and compression of an exponential function.

The model used to predict future prices of a S&P 500 stock is an iterative process. By modifying the three parameters and iterating through values of a specific range for each parameter, we can fit an exponential curve tailored for the respective stock. By iterating through random values of each parameter in a desired range, the model keeps the best parameter values. It adjusts values of each parameter depending on whether the predicted values overshoot or undershoot the data. We used the mean square error (MSE), as shown in Equation 2,

$$\text{Cost or MSE} = \sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n} \tag{2}$$

where $\hat{y}$ is the predicted value and $y$is the actual value, as an indicator of how close our predicted values are to the actual values of a stock price in 2017. The smaller the MSE, the better our prediction. Therefore, in an iterative model like ours, the more iterations performed, the higher the chance of obtaining a smaller MSE. A similar iterative process model used to predict the movement of the stock market has been done before, but instead of predicting the future stock price, an iterative calibration was used to predict a future market crash (Pele). Perhaps, the main constraint for this type of model is that it can only predict prices for stocks with exponential growth. In a case where a stock grows linearly, has little to no movement, or decreases, this model would nevertheless try to fit an exponential. Therefore, it is only a great model for the more innovative and well managed companies of the S&P 500, stocks like Amazon or Nvidia for instance (Figure 3).

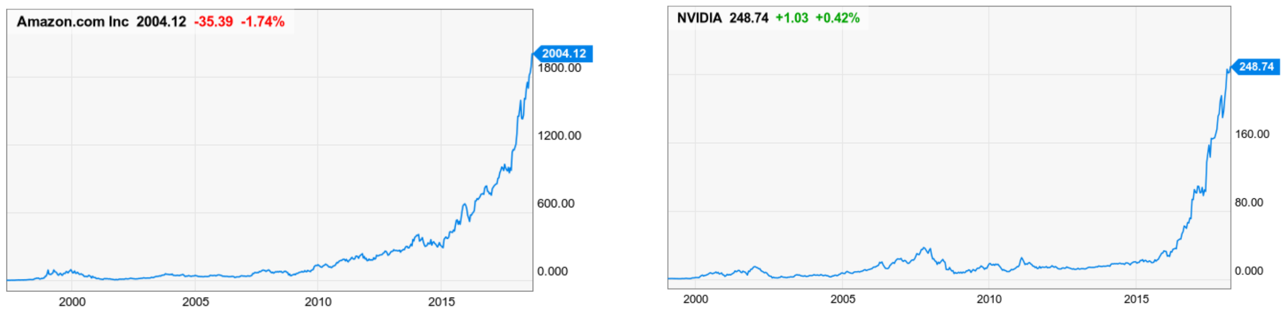


Figure 3. Amazon (AMZN) and NVIDIA (NVDA) stock chart demonstrating exponential growth.

DESCRIPTION OF COMPUTATIONAL APPROACH

The mathematical approach to the prediction of a S&P 500 stock price was accomplished using the iterative model described in the "Description of Model" section. We imported data of all individual stocks of the S&P 500 from the start of 2013 to the end of 2017. By only using data prior to the year 2017, we predicted prices of a specific stock in 2017 and compared it to the actual prices using the MSE. In our model, we used the “close price” of a stock on a given day as the dependent variable. Our mathematical model consists of 5 main functions:

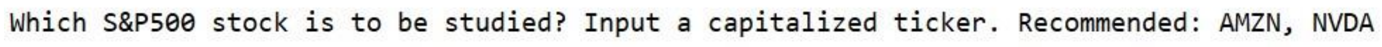
1. The Exponential Function
2. The Cost Function (MSE)
3. The Function of Greater-Lesser
4. The Normalized Root Mean Square Error (NRMSE)
5. The Iterative

The Exponential Function, in the form previously described in Equation 1, takes in the date of each day from 2013 to 2016 (excluding weekends and holidays since the Stock Market is closed on those days) as the independent variable and the three guess parameters (a, b, c). It then returns the predicted price of the stock.

The Cost Function returns a number (MSE, referred to as “cost” in our model) used to quantify the error between the predicted price and the actual price of our model. The smaller the cost, the closer the prediction is to the actual price of the stock. This cost function is very common in such processes and is simply the squared value of the average of predicted values minus the actual values. Squaring eliminates negatives.

The Function of Greater-Lesser returns a positive or negative number, telling us whether the prediction was superior or inferior to the actual price of the stock on average. It is coded similarly to the cost function but does not incorporate the squaring of the difference allowing for an either negative or positive average value. The sign of this output is especially important when determining whether the model should increase or decrease the parameters during the iteration step.

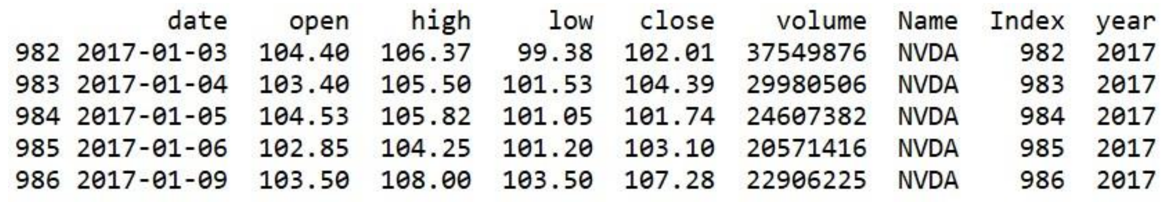
Finally, the iterative part of the code is the most important. It starts off by generating a random (within a range) list of values for parameters a, b, c. This component of the code will compute and return a cost for a specific stock in the S&P demanded by the user, which will give an indication of how good the predictions were for 2017 (Figure 4).



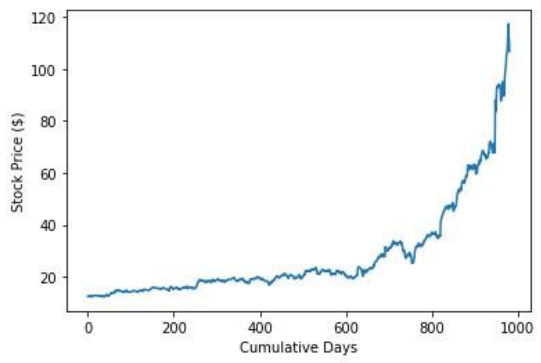
```
Which S&P500 stock is to be studied? Input a capitalized ticker. Recommended: AMZN, NVDA
```

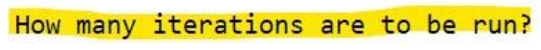
Figure 4. User input about which stock of the S&P 500 is to be studied.

Then, for each time the cost function is run, the greater-lesser function also runs to determine whether the predicted price at that instant overshot or undershot. For each parameter, starting with parameter b, it will slightly increase parameter b if greater-lesser is negative and slightly decrease parameter b if greater-lesser is positive. It then computes the cost. If this latter cost is



```
           date    open    high     low   close    volume  Name  Index  year
982  2017-01-03  104.40  106.37   99.38  102.01  37549876  NVDA    982  2017
983  2017-01-04  103.40  105.50  101.53  104.39  29980506  NVDA    983  2017
984  2017-01-05  104.53  105.82  101.05  101.74  24607382  NVDA    984  2017
985  2017-01-06  102.85  104.25  101.20  103.10  20571416  NVDA    985  2017
986  2017-01-09  103.50  108.00  103.50  107.28  22906225  NVDA    986  2017
```

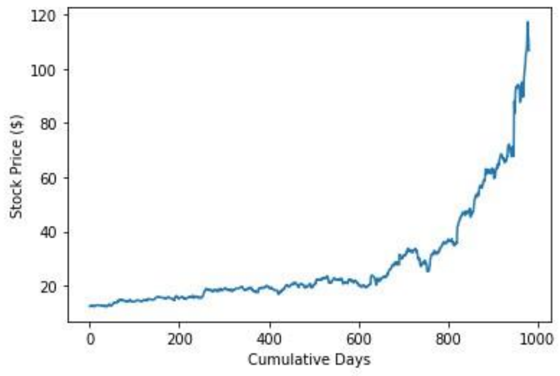




```
How many iterations are to be run?
```

less than the former, the process repeats. This goes on until the cost produced by decreasing or increasing b is no longer less than the previous. This process is repeated in turn for parameters a and then c. The order of iteration for these three parameters (in order of b→a→c) generated the best cost and was decided after multiple trials of all possible orders. When all of this is done, we get the best cost the algorithm could find for the randomly generated (within a range) set of guess values for each parameter. Depending how many iterations are to be run (the aforementioned total process from parameters b→a→c counts as a single iteration), the process will be repeated with a new set of randomly generated guesses (Figure 5). The algorithm stores and compares the optimized cost of each iteration and keeps the best cost and the optimal parameters. Considering the nature of the hard stop once the modification of a parameter is no longer fruitful, it is necessary to incorporate more than 1 iteration with variable initial guesses to avoid local minima influence.

```
          date    open    high     low   close    volume  Name  Index  year
982 2017-01-03  104.40  106.37   99.38  102.01  37549876  NVDA    982  2017
983 2017-01-04  103.40  105.50  101.53  104.39  29980506  NVDA    983  2017
984 2017-01-05  104.53  105.82  101.05  101.74  24607382  NVDA    984  2017
985 2017-01-06  102.85  104.25  101.20  103.10  20571416  NVDA    985  2017
986 2017-01-09  103.50  108.00  103.50  107.28  22906225  NVDA    986  2017
```



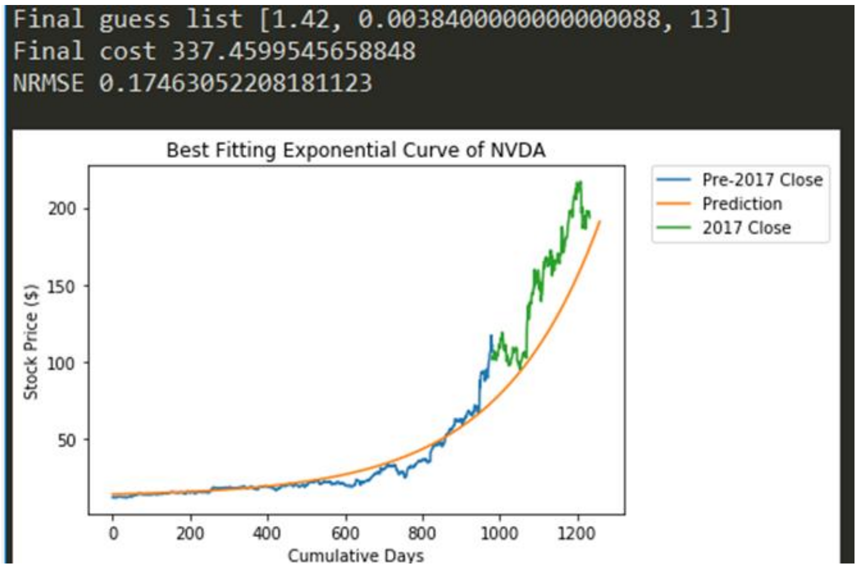
```
How many iterations are to be run?
```

Figure 5. For NVIDIA stock, the computational model asks the user how many iterations are to be run.

After all iterations, the predicted stock prices, in the form of an exponential function, are graphed alongside the actual prices, and the model prints out the optimal cost and its respective parameters as illustrated in Figure 8.

RESULTS

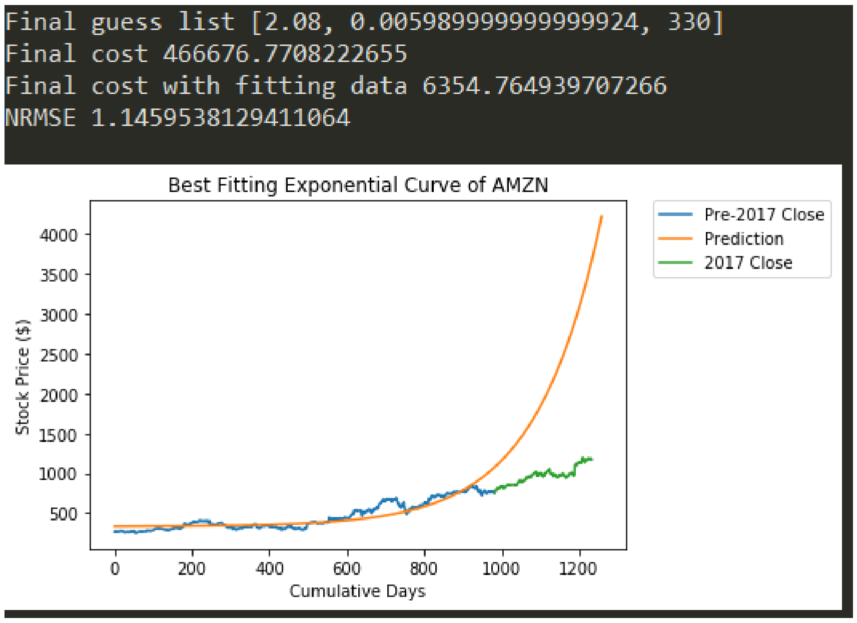
As mentioned in the "Description of Model", this iterative approach of trying to fit an exponential function in order to predict stock prices only works well with certain stocks of exponential growth, notably Amazon (AMZN) and NVIDIA (NVDA) among others. Therefore, we tested how good our model was using these two tickers. For NVDA, after running 5 iterations, the best prediction had a cost value of 337.46 (this cost is calculated using the fitting data AND the test data of 2017) with parameters a, b and c respectively being equal to [1.42 , 0.00384, and 13] (Figure 6).



```
Final guess list [1.42, 0.0038400000000000088, 13]
Final cost 337.4599545658848
NRMSE 0.17463052208181123
```

Figure 6. For NVIDIA stock, the computational model outputs the cost and its corresponding parameters after 10 iterations.

As for AMZN, since the values were larger and it took more time to run a single iteration, we only ran 3 iterations. As a result, the best prediction, while having a very impressive fitting data cost of 6354.76, had a tremendous final cost of 8466676.77 due to the fit only being adequate for the studied years of 2013-2016 along with parameters values of [2.08, 0.00599, 330] for a, b and c respectively (Figure 7).



```
Final guess list [2.08, 0.005989999999999924, 330]
Final cost 466676.7708222655
Final cost with fitting data 6354.764939707266
NRMSE 1.1459538129411064
```

Figure 7. For AMZN stock, the computational model outputs the cost and its corresponding parameters after 3 iterations.

A better way of evaluating the costs of both tickers is by looking at its normalized root mean square error (NRMSE), by taking the root of the MSE and dividing that by the range of stock prices. In that case, NVDA has a NRMSE of about 0.1746 and AMZN has a NRMSE of about 1.1459.

We also tested our mathematical model on stocks that have non-exponential growth. For ticker OXY (Occidental Petroleum), which decreased in price and whose movement did not fit a particular pattern from 2013 to 2017 as shown in Figure 8, it was impossible to fit an exponential curve. The prediction had a cost of fitting data of 626.45 and a NRMSE (of the fitting data only, from a previous version of the code) of 0.5492 all of which are much worse when the 2017 test data is included in the calculation.

```
guess list [2.4800000000000004, 0.0032900000000000078, 63]
cost 626.449528322496
NRMSE 0.549242347528089
```

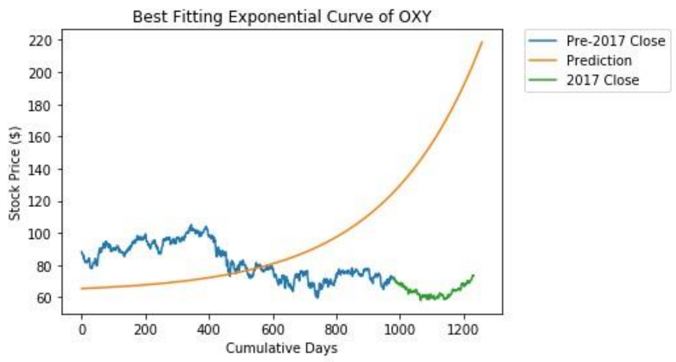


Figure 8. For OXY stock, the computational model outputs the cost and its corresponding parameters after 3 iterations.

Conclusion

Overall, our iterative model of attempting to predict stock prices of S&P 500 stocks in 2017 can generate great results but is only limited to the stocks that have an exponential growth from 2013 to 2017. Even in the aforementioned case, stocks who initially seem greatly exponential but later lessen their exponentiality like Amazon will find great fits to their fitting data, but this fit might be non-representative of future price. These issues, while important, lie in the concept of curve fitting with respect to stock data rather than in our specific model as they are intrinsic to curve fitting no matter the algorithm. To solve them, one would need a model more complex than curve fitting such as the GAM (Generalized Additive Model), a model that we initially were interested in but whose math quickly proved to be beyond our current knowledge.

Works Cited

Lumen Learning. “Stretching, Compressing, or Reflecting an Exponential Function”. College Algebra, module 11. https://courses.lumenlearning.com/waymakercollegealgebra/chapter/stretch-or-compress-an-exponential-function/

Harwood, Vance. “Why 18.5 is the right PE ratio for the S&P 500”. Six Figure Investing, 10 March, 2017, https://sixfigureinvesting.com/2012/05/why-18-5-is-the-right-pe-ratio-for-the-sp-500/.

Pele, Daniel, and Miruna Mazurencu-Marinescu. “Modelling stock market crashes: the case of Bucharest Stock Exchange.” Procedia - Social and Behavioral Sciences 58, pp. 533-542, 2012. https://pdf.sciencedirectassets.com/277811.