Schulich School of Business

TEAM 007

PREDICTION GURUS

MMAI 5300 Final Group Assignment

Insights on Linear Regression Applications for Stock Market Predictions

Submitted on 2021/04/09
by **Team 007**:
Awmnah Asad
Najih Shabir
Ricardo Luhm
Serif Ersegun Kocoglu

1. Executive summary

Predicting stock prices in advance is vital for speculators, investors, and business people. Predicting the stock prices is essential not only for investors who buy and sell for speculative purposes but also for financial managers aiming to maximize firm value and the whole economy. The ultimate goal in this market is to determine the variables that affect the markets and their effect on stock values in real-time and make an investment decision.

Linear Regression is a powerful tool to analyze data trends and even predict the outcomes of a given set of features. Several linear regression approaches can provide solutions with slightly different processing times and forecasting precision. Stock Market prediction is one example of an analysis that can benefit from Linear Regression. The purpose of this report is to create four other simulations to evaluate the performance of Linear Regression on Random Generated Datasets before moving on to the actual Stock Market predictions.

Applying Linear Regression in the simulations allowed us to find which approaches had the best results given the shape (rows AND columns) of the input matrix and the best processing times. As for the real data, we found that mixing stock market prices and volume without any data preprocessing, such as converting input data into logarithmic scales before applying linear regression, improves forecasting precision. However, the stock prices range from zero to thousands, while volume ranges from millions to billions of transactions. This dramatically affects the predictions of unnormalized data.

The stock market prices were trained separately from the stock volume, and the effects of logarithm preprocessing were also evaluated to address these issues. Although the overall error was greatly reduced, it was not enough to provide price predictions. That might happen when you feed the model with the entire time range to learn to forecast, making it difficult to predict price spikes or trade volumes.

Based on these findings, we conclude that linear regression can detect if a stock market is on the rise or going down, but we could not achieve an excellent exact price prediction. Our recommendation for better modelling for stock market predictions is to make further investigations to explore smaller time samples' effects by applying techniques such as ARIMA, Multilayer Perceptron, and XGBoost.

2. Simulations

a. Scenario 1

The purpose of the simulation is to understand the numerical implications for a random squared matrix with values ranging from -1 to 1. Python libraries such as Numpy, Sympy were used to convert Algebra formulations into the code (N*umerical_Analysis_Sim_1.ipynb*). The following equations were used in this simulations:

$$Ax = b \;\; => \;\; Given \text{ “}A\text{” } and \text{ “}b_{true}\text{” } are\ known$$

$$find \;\; "x" \;\; where\ A^{-1} * Ax = A^{-1} * b$$

$$x_{ext} = A^{-1} * b$$

$$compare \;\; "x_{ext}" \;\; with\ "x_{true}"$$

The algorithm produces the following graphs as outputs in Figure 1.

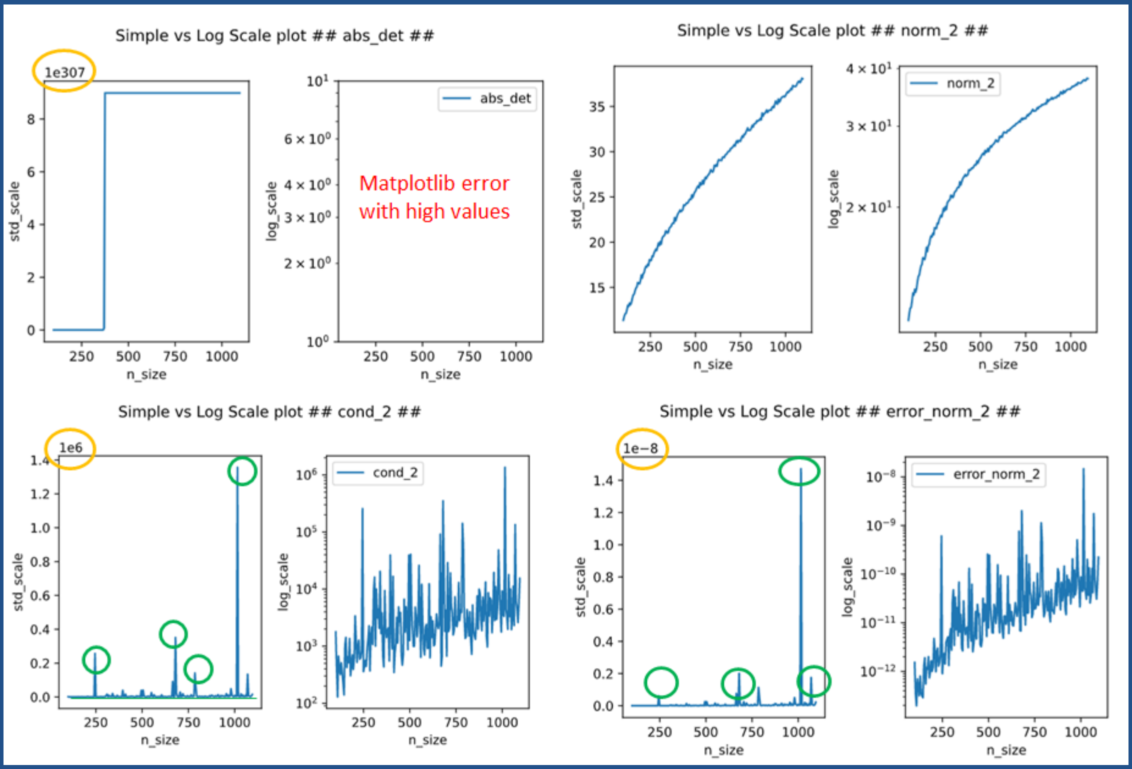


Figure 1. Matrix Determinant (top-left), Norm(top-right), Condition(Bottom-Left) and Error (Bottom-Right) using normal and log scales.

We can identify that the determinant goes to infinity after matrix size increases around 300-350. Condition has a linear behaviour but there are some spikes in the graph where the values increases almost ten thousand. Those spikes are noises and they do not represent the model as a whole. The same happens for the norms of the error, but they are extremely low.

The time for each operation was also monitored. The graph below shows that all time operations are on an exponential increase as the matrix size increases. This helps us estimate what would be the time required for operations with larger dataset and setup a virtual machine able to compute the given data accordingly.

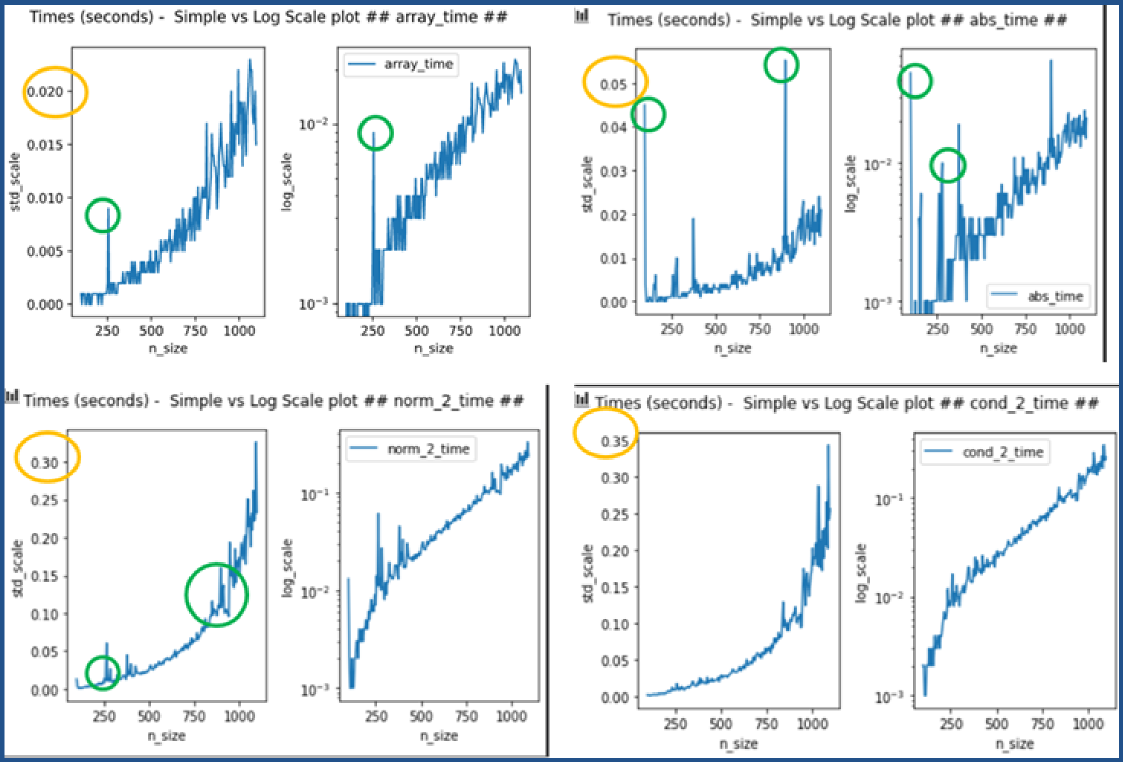


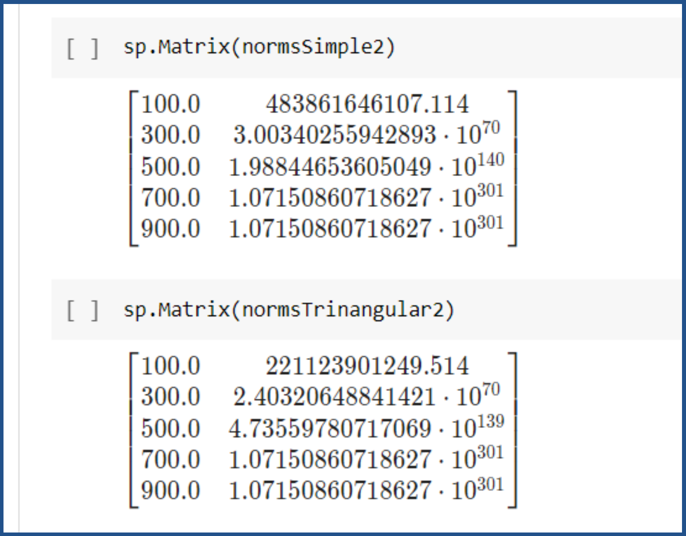
Figure 2. Matrix Determinant (top-left), Norm(top-right), Condition(Bottom-Left) and Error (Bottom-Right) using normal and log scales.

Figure 2 also shows some spikes circles in green, which must be related to random noize as well. Generating the array is the fastest operation, while the norm and cond require more time to be calculated.

b. Scenario 2

Simulation Scenario 2 is required to solve the triangular system by upper triangular matrix and to check the norm of the vector error. Then the norm2 values (error simple and error tri) were plotted against the matrix size *“n”*, on both regular and log scales. It was evaluated what were the changes in the outputs as *“n”* increases. After that the running times of the functions were plotted. The main goal of this simulation is to check matrix complexity, precision, and time to see the behaviour of the algebra solutions. Below are the results of the code (N*umerical_Analysis_Sim_2.ipynb*).

The code output shows that the norm values go to infinity around “n = 500” and “n=600”. The output belows shows the approximation that was done in the code to replace infinity by the maximum value accepted by a float 64.



```
[ ] sp.Matrix(normsSimple2)
```

$$\begin{bmatrix} 100.0 & 483861646107.114 \\ 300.0 & 3.00340255942893 \cdot 10^{70} \\ 500.0 & 1.98844653605049 \cdot 10^{140} \\ 700.0 & 1.07150860718627 \cdot 10^{301} \\ 900.0 & 1.07150860718627 \cdot 10^{301} \end{bmatrix}$$

```
[ ] sp.Matrix(normsTrinangular2)
```

$$\begin{bmatrix} 100.0 & 221123901249.514 \\ 300.0 & 2.40320648841421 \cdot 10^{70} \\ 500.0 & 4.73559780717069 \cdot 10^{139} \\ 700.0 & 1.07150860718627 \cdot 10^{301} \\ 900.0 & 1.07150860718627 \cdot 10^{301} \end{bmatrix}$$

By observing the standard and log scale, norm2 simple error and norm 2 triangular error, it was identified that they are almost identical. To better visualize both graphs the norm triangular value was divided by 0.99 just to show the line for the visualization, however both lines overlap when considering true values.

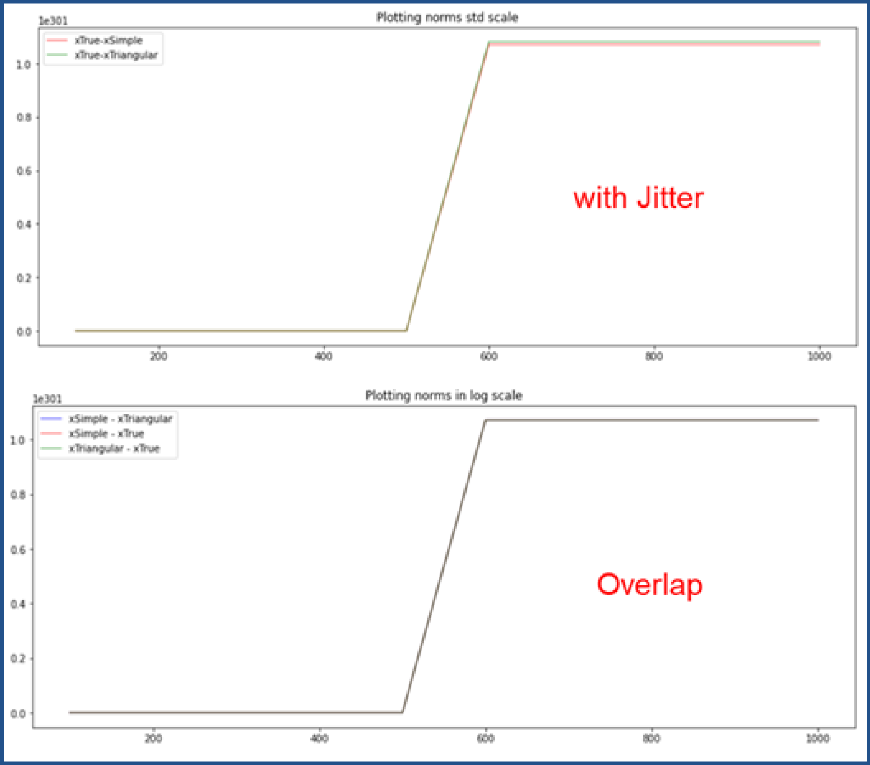


Figure3. Overlapping results solved with jitter.

Figure 3 shows the results considering the standard scale, for the abs determinant, as n increases. The abs determinant quickly approaches 1 where it stays at 1 no matter how much n increases. For conditions, conditions raise from 0 to 2 as n goes from 0 to 400, and then drops when n is 600, and then rises to 5 when n is 800 and then decreases to 1. For norm, n goes from 0 to 700,000 as n goes from 0 to 1000.

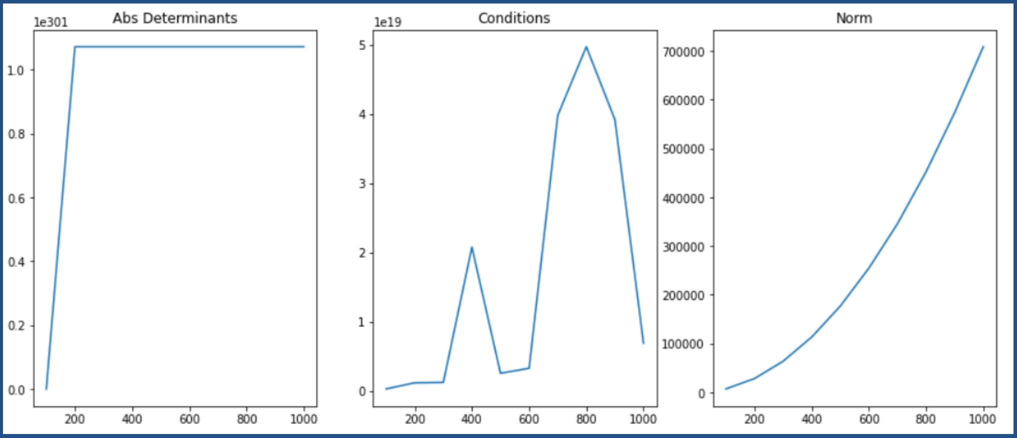


Figure4. Simulation 2 results with standard scale

For the log scale, for the abs determinant, as n increases, the abs determinant quickly approaches 700 as n reaches 300 and remains at 700 as n reaches 1000. For conditions, conditions raise from 42 to 47 as n goes from 100 to 400, and then drops to 42.5when n is 700, and then rises to 44.5 when n approaches 90. For norm log, n goes from 9 to 13 as n goes from 100 to 900.

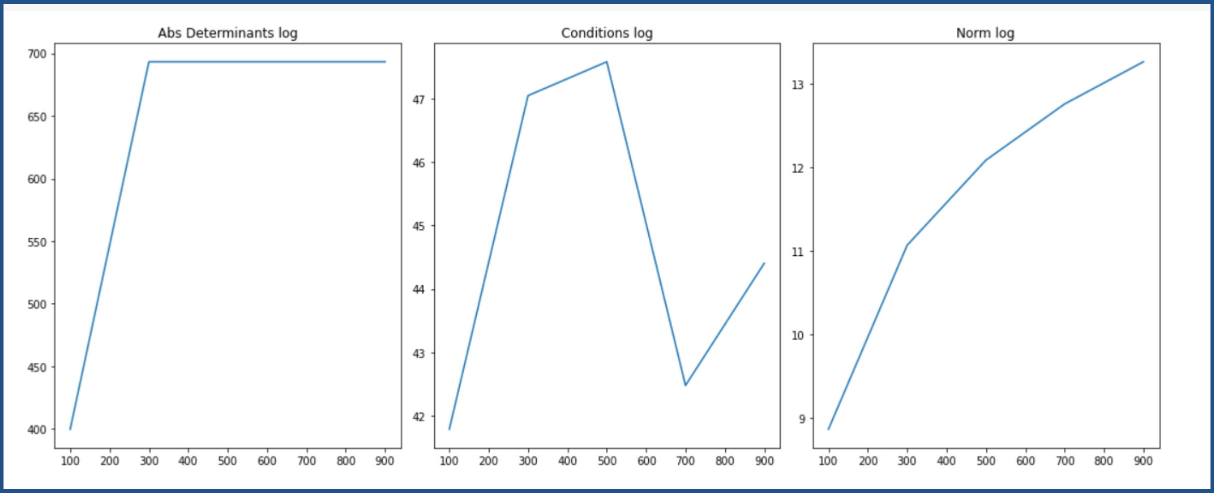


Figure 5. Simulation 2 results with log scale

It can be also noticed that in the log scale, the triangular error's (green line) time increases from step 1 to 4 but then decreases.It wase also noticed that in the log scale, the simple error takes less time than the triangular error from steps 1 to 4, however after step 8, the simple error (orange line) starts to take more time than the triangular error.

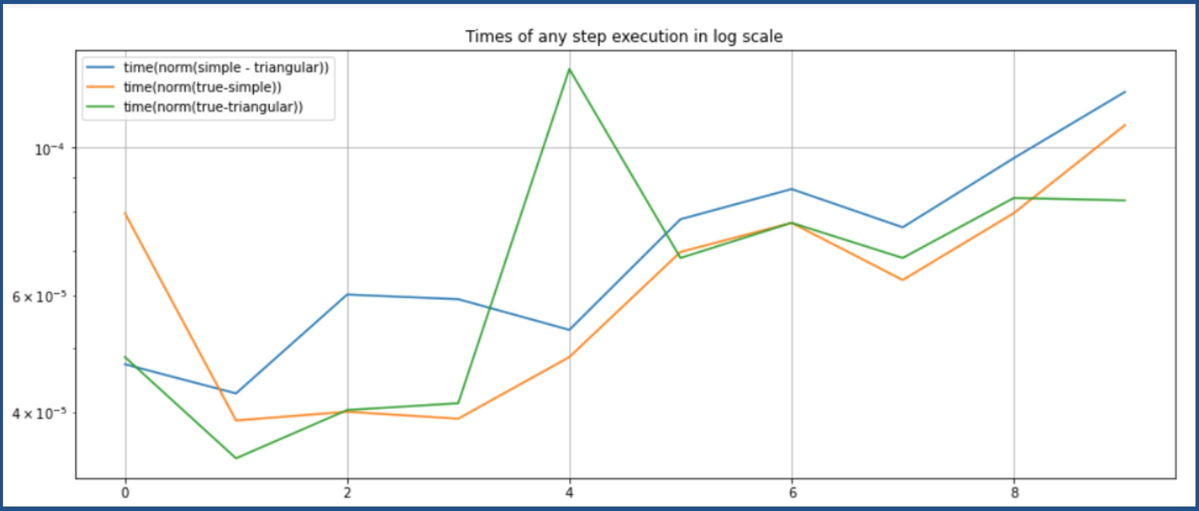


Figure 6. Time comparison between algorithm operations - Log Scale

The results of the time plot on standard scale shows that the norm 2 simple error takes a lot more time than the norm 2 triangular error line after the first step.

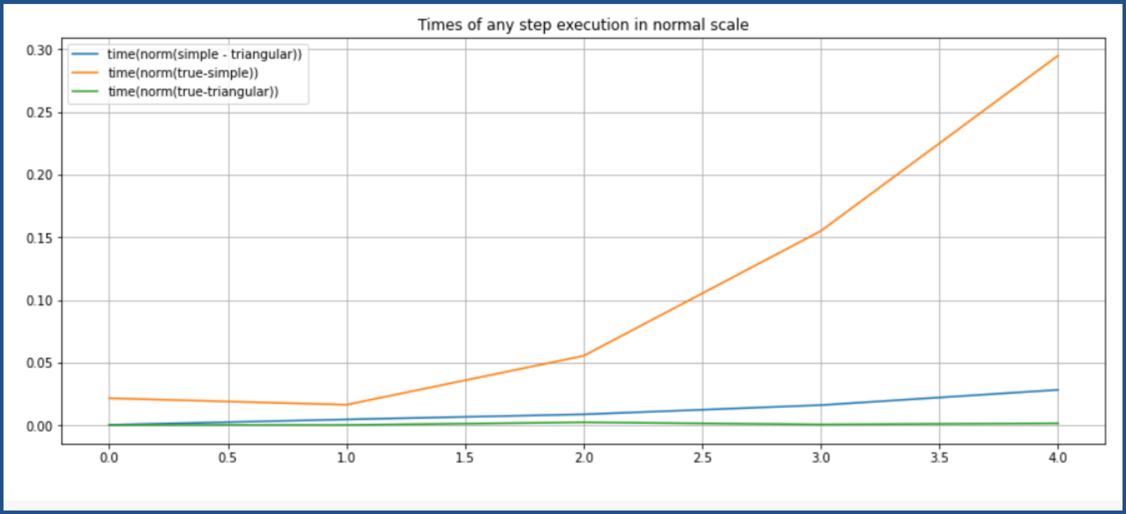


Figure 7. Time comparison between algorithm operations - Standard Scale

c. Scenario 3

The third scenario simulates how QR decomposition works. First we generate an upper triangular array (R_true) and make sure that its diagonal has only positive values. Then we create an orthogonal array(Q_true) from the same size and the upper triangle. The final array "A" is retrieved by multiplying the orthogonal array by the upper triangle. To evaluate the precision from the QR decomposition, we use "A" and retrieve the "Q_estimated" and "R_estimated". Then we compare the true values with the estimated values. Figure 8 shows the error and the time required to generate "A" and the time to decompose "A".

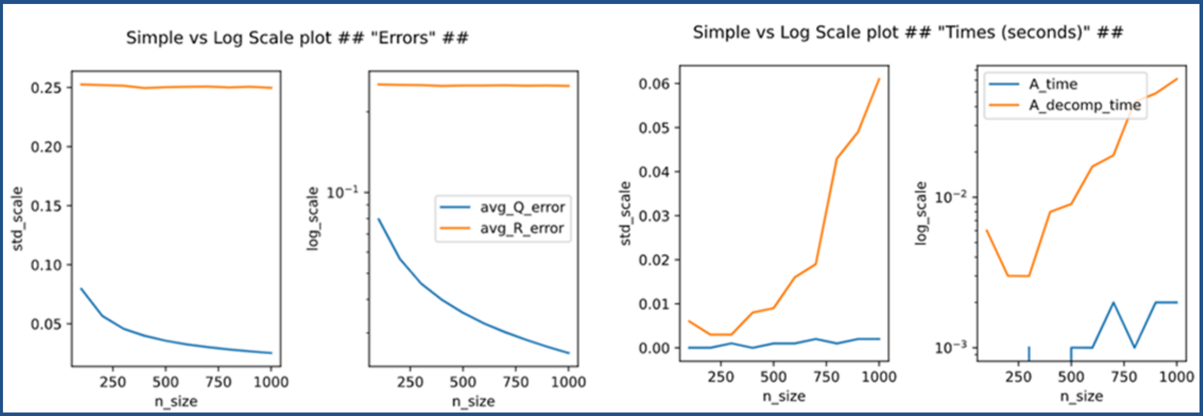


Figure 8. QR decomposition Error and time comparison.

The error of the "R" component is constant which means there is a bias that can be applied to retrieve the correct output. The error of the "Q" component is slightly lower and it decreases with the size of the matrix. The time to generate "A" has a very small slope while the decomposition time has an exponential behavior when the size of "A" increases.

d. Scenario 4

Simulation scenario 4 looks at normal equations with and without QR decomposition. First a matrix A (m by n) was randomly generated. Next, three different operations were performed: pseudoinverse of matrix A (A_psinv), decomposition of matrix A (qr_decomposition), and simplified pseudoinverse of matrix A (A_qrpsinv). The pseudoinverse computes the least squares solution to a system of linear equations that does not have a solution. In addition, pseudoinverse can also be used to find the minimum norm solution when there are multiple solutions in a system of linear equations. We then looked at the computation time of the three different operations as the matrix length and width increases, which is shown in figure 9 and figure 10. Finally, we also look at the difference between the two methods to calculate the inverse of matrix A, which is shown in figure 11.

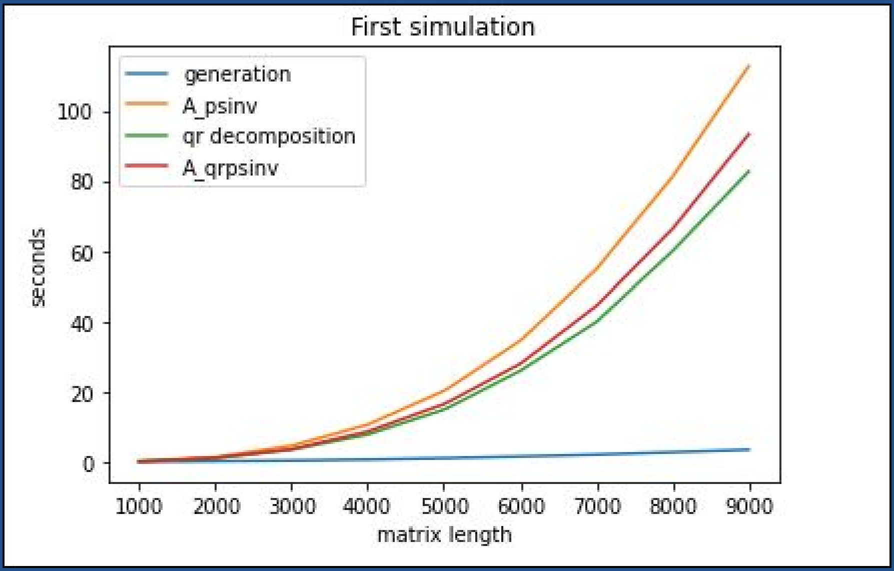


Figure 9. Computation time of the 3 different operations by matrix length

Figure 9 shows a graph of the time it takes to compute the three different operations: pseudoinverse, decomposition, simplified pseudoinverse. The graph shows us that as the matrix length increases the time it takes to compute the three different operations increases exponentially. Furthermore, it is evident from the graph that computing pseudoinverse takes the longest, followed by the simplified pseudoinverse and finally the QR decomposition.

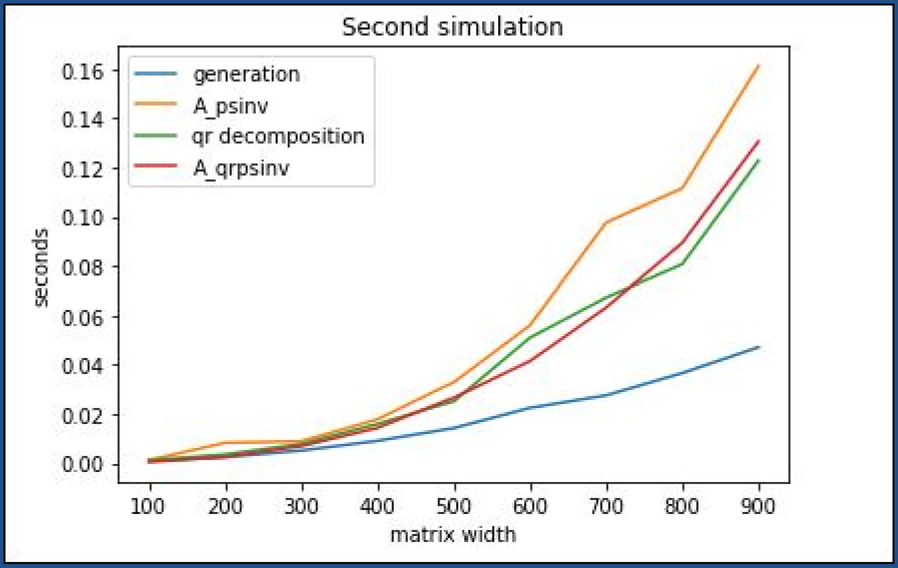


Figure 10. Computation time of the 3 different operations by matrix width

Figure 10 shows a second simulation of the time it takes to compute the three different operations: pseudoinverse, decomposition, simplified pseudoinverse. However, this time we look at the time it takes for the computation of those operations as the matrix width increases. Similar to simulation 1, the time increases as the matrix width increases. But the increase is not exponential as shown in the previous figure. Furthermore, the order of computation time for the different operations is similar to simulation 1 with the pseudoinverses taking the longest, followed by the simplified pseudoinverse and then the QR decomposition.

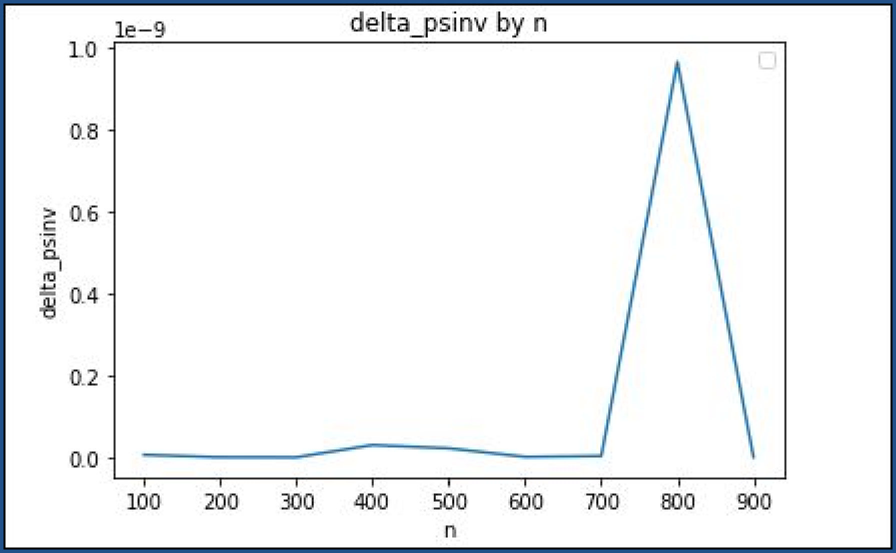


Figure 11. Calculated Delta(Error) values of pseudo inverse

Figure 11 plots the delta_psinv by the matrix size. The delta_psinv is the error or difference between solving the linear system with pseudoinverse and qr pseudoinverse. As we can see from the graph, the result of the two methods to calculate the inverse from matrix A is very small since the scale is 1e-9. Thus, we can either use pseudoinverse or simplified (qr) pseudoinverse to calculate the inverse of A.

3. Real Data and Our Methodology

The Stock Price dataset used had daily stock price closing and trading volume data for 20 companies between 2005 and 2020.

Since the 2020 COVID-19 effect is also included in our data set, we considered the excessive increase in some stocks this year as "force-majeure". In other words, we evaluated the impact of this unprecedented year on the aggregate trend when creating our LR model.

Tableau was used to extract visual insights and some preliminary analysis with actual data. By doing this, it was possible to understand whether we can do regression analysis or not. Figure 12 shows a regression for Apple stock prices considering.

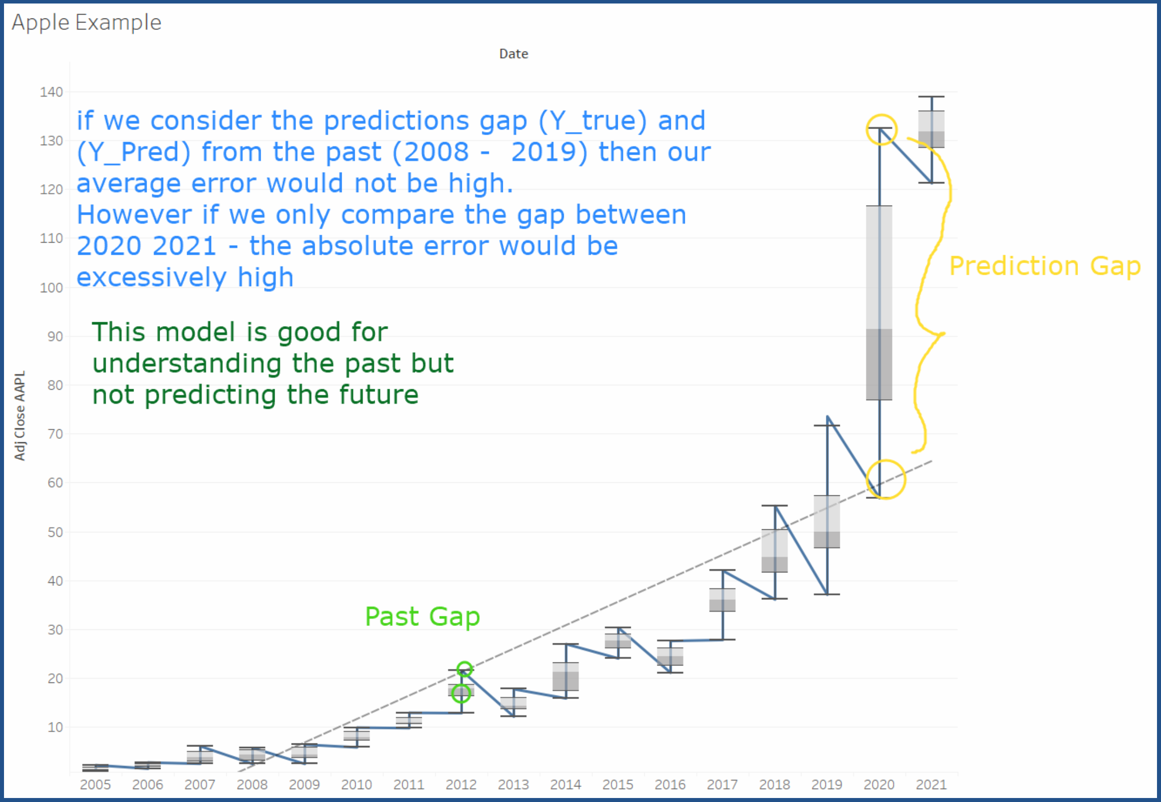


Figure 12. Linear regression of a single stock example.

This analysis looked at the development of the annual average stock price and daily transaction volume data between 2005 and 2021. The purpose of this stage is to find and decide the best method for prediction for this analysis. To evaluate the prediction efficiency the MAE (Mean Absolute Error) was used.

- By looking at Absolute Error from Raw Data, it was found that the Error was more than 1 trillion. We found that Error after 0.2 was extremely high.
- Then, when we did Narrow View, we found that eight errors occurred between 0 and 0.2.
- When we applied Log to Wide View, we found that our Regression model performed 10,000 times better.
- When we applied Very Narrow View to Log Data, we found that our Regression Model got to a much better point.

Based on these results, we developed a model hypothesis as:

"Separating the Adj Close and Volume dataset and applying regression to two different datasets will increase our prediction"

Figure 13 shows the comparison of four approaches: training with: raw price and volume, log price and volume, raw price and log price.

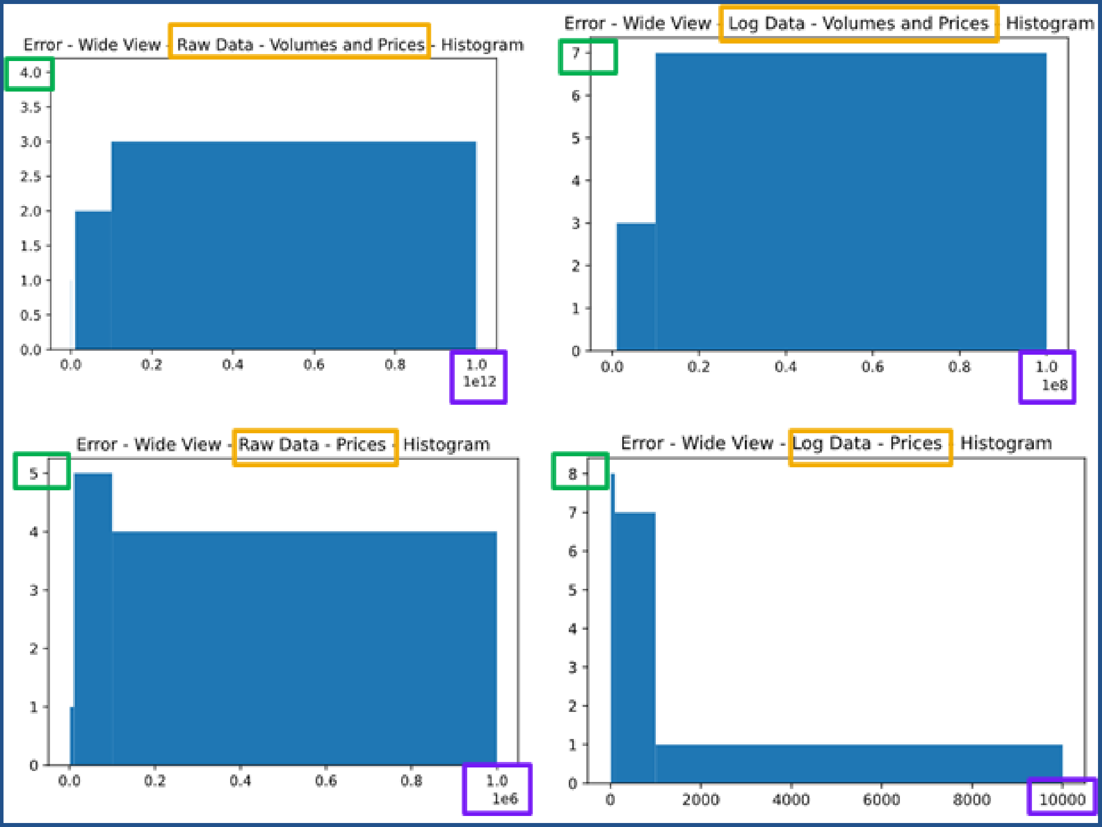


Figure 13. Linear regression and preprocessing steps VS prediction error - Wide View

Applying log before finding the regression coefficient greatly reduces the overall error, but not enough to provide an accurate forecast. Using price data only results improving even further, both to raw data and log data. However forecasting error is still high. The following figure shows why the forecasting error is still high.

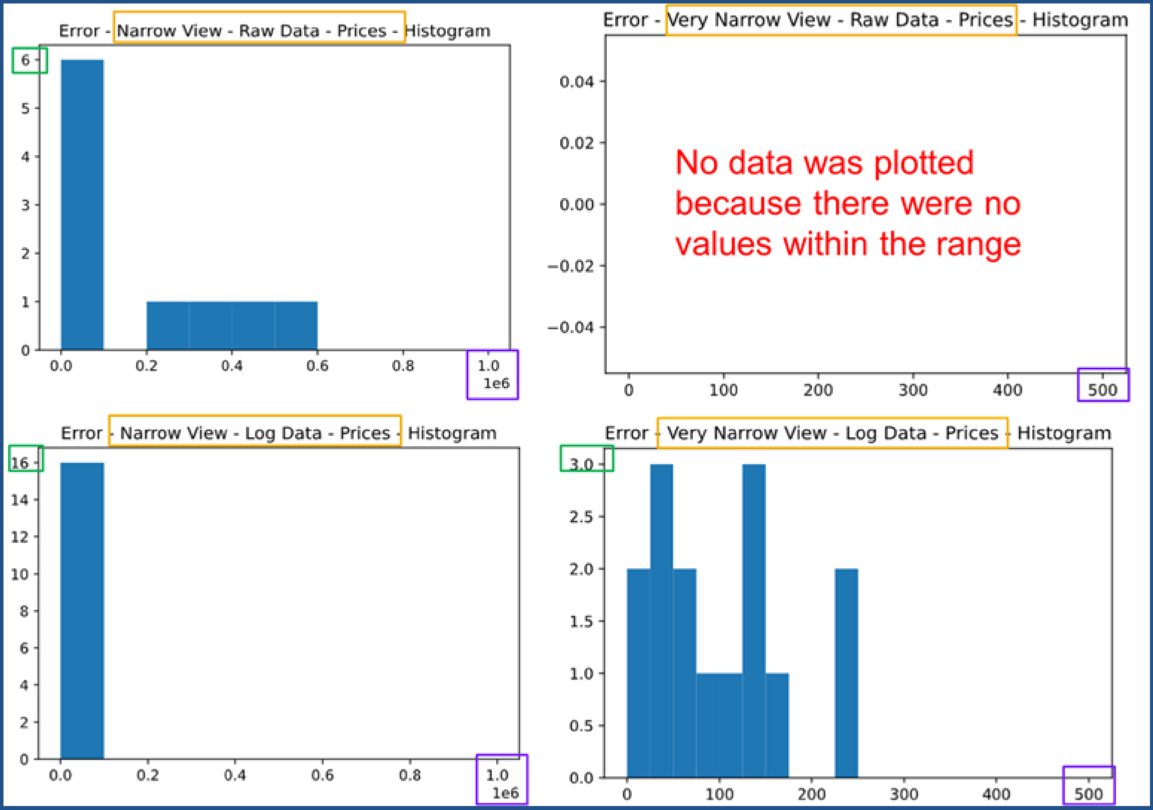


Figure 14. Linear regression and preprocessing steps VS prediction error - Narrow View

Because we thought Volume was behaving differently than Stock Price.

- Our Raw Data content
    - Stock Price
    - Transaction Volume

The diagram in figure 15 shows how the problem was approached in this report, with the Solution Pipeline and the errors for each attempt.

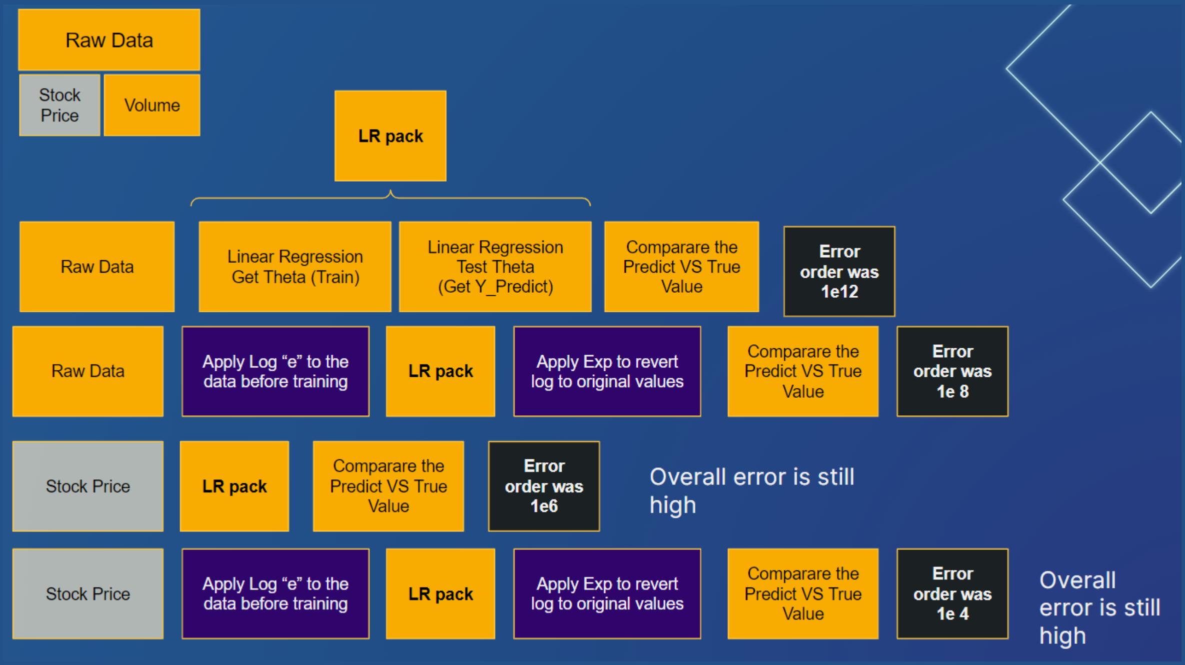

Raw Data
Stock Price
Volume
LR pack
Raw Data
Linear Regression Get Theta (Train)
Linear Regression Test Theta (Get Y_Predict)
Comparare the Predict VS True Value
Error order was 1e12
Raw Data
Apply Log "e" to the data before training
LR pack
Apply Exp to revert log to original values
Comparare the Predict VS True Value
Error order was 1e 8
Stock Price
LR pack
Comparare the Predict VS True Value
Error order was 1e6
Overall error is still high
Stock Price
Apply Log "e" to the data before training
LR pack
Apply Exp to revert log to original values
Comparare the Predict VS True Value
Error order was 1e 4
Overall error is still high

Figure 15. Linear regression and preprocessing steps VS prediction error.

First stage
- With Linear Regression, we found Theta and trained it.
- Then, we applied Linear Regression with Test Theta to find Y_Predict.
- Comparing the Y_Predict we found with the actual value, we encountered a 1e12 level error

Second stage
- We applied Log to the data before training.
- After applying LR, we applied Exp to revert the Log to original values.
- As in the first step, we compared the Y_Predict we found with the actual value.
- This time, we encountered a 1e8 level error.

Finally
- We applied LR for the stock price data set
- We compared the Y_Predict we found with the actual value.
- This time, we encountered a 1e6 level error.

4. Conclusions and Next Steps

Traditional statistical methods were initially used to estimate stock prices. Some of these methods are prediction studies based on time series models such as AR, MA and ARIMA. However, these models require statistical assumptions about the data set. In other words, to use the methods, some conditions must be met in the data set. For this reason, their forecast performance is limited.

Our analysis in this report is prepared with a data set consisting of two 16-year parameters belonging to 20 companies. In other words, it does not include all the parameters needed to predict the future value of stocks. For this reason, we used the LR (Linear Regression) model and we tried to answer the following question in the analysis we made only with the variables of daily closing values and transaction volumes.

"Can we apply Linear Regression to the Stock Market?"

It depends. If you use a narrow range of prices, the model starts behaving like ARIMA - Auto-Regressive Integrated Moving Average. Also, Linear Regression can be used to get a simple insight if the stock prices are increasing or not, but their prediction error is not as accurate as when compared with getting the actual price for the next day. Figure 16 shows how to improve predictions using linear regression using a narrower time range.

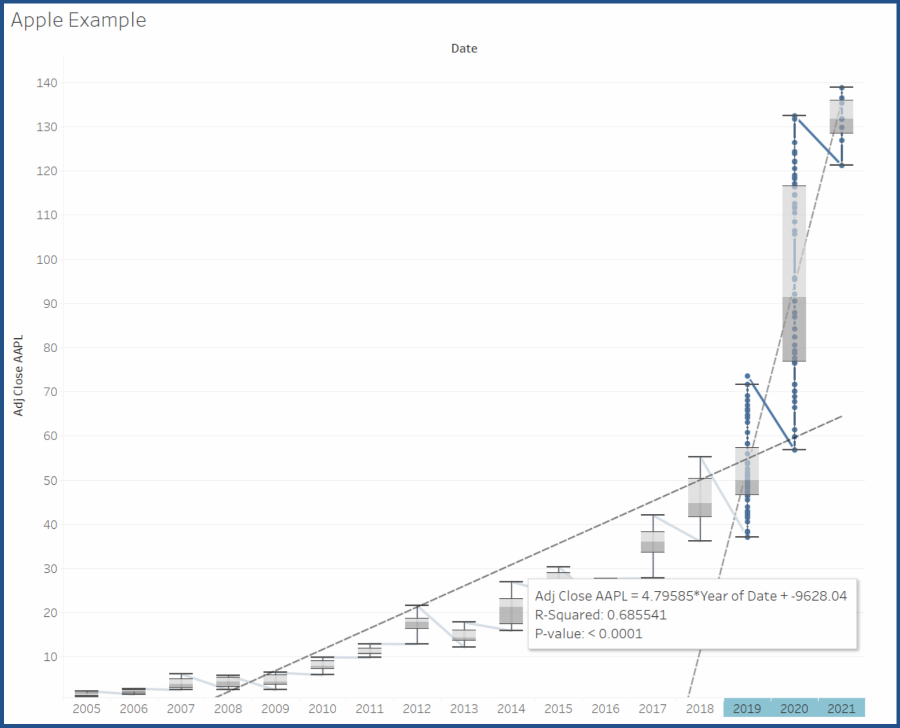


Figure 16. Linear regression and preprocessing steps VS prediction error.

As shown in the stages of our Solution Pipeline (Figure 15), we cannot say that the model is very satisfactory even at the lowest error level we can reach. We have determined that the error level can go down to 1e6 in the most successful LR model we have achieved.

At the end of this analysis, we made the following resolutions about our model:

1. Our model is good but not the best.
2. This is because the capacity to understand the past is high, but predicting the future is low.

Next Steps

Traditional methods developed have been partially successful due to the non-linear and complicated performance of stock prices. Artificial neural networks proposition is a relatively new, active and promising area in predicting stock price behavior.

Today, artificial neural networks are popularly applied to many financial problems such as stock market index prediction, bankruptcy forecast or corporate bond classification.

Based on the results we have obtained from this study and our research, we propose the following steps.

1. Testing New Models:
   To test the following Machine Learning models by adding stock opening, highest value, lowest value parameters to the parameters:

   - Last Value
   - Moving Average
   - Extreme Gradient Boosting (XGBoost)
   - Long Short Term Memory (LSTM)

2. Developing an Artificial Neural Network Model:
   To work on promising Artificial Neural Networks models tested for more than 20 years in the world by adding new parameters to the data set.

One of the first studies carried out is Kimoto et al. (1990)[1] on the Tokyo stock market index. Later, Kamijo and Tanikawa (1990)[2] used recurrent neural networks, and Ahmadi (1990)[3] used the backpropagation network. Yoon and Swales (1991)[4] studied with both quantitative and qualitative data. Mizuno et al. (1998)[5], on the other hand, predicted the buy and sell signals of the Tokyo stock market with 63% accuracy using artificial neural networks.

[1] "Stock market prediction system with modular neural ... - IEEE Xplore." https://ieeexplore.ieee.org/document/5726498. Accessed 9 Apr. 2021.
[2] "Stock price pattern recognition-a recurrent neural ... - IEEE Xplore." https://ieeexplore.ieee.org/document/5726532. Accessed 9 Apr. 2021.
[3] "Testability of the arbitrage pricing theory by neural ... - IEEE Xplore." https://ieeexplore.ieee.org/document/5726558. Accessed 9 Apr. 2021.
[4] "Predicting stock price performance: a neural network ... - IEEE Xplore." https://ieeexplore.ieee.org/document/184055. Accessed 9 Apr. 2021.
[5] "Application Of Neural Network To Technical Analysis Of Stock ...." 12 Jul. 2001, http://nas.uhcl.edu/boetticher/ML_DataMining/Kimoto.pdf. Accessed 9 Apr. 2021.

The prediction accuracy is not very good due to the complexity of the data and the local convergence of the algorithms used. Sexton et al. (1998)[6] suggested that momentum (speed factor), restarting the training process at random points, and restructuring the network structure can solve the problems. They state the direction to be followed as the use of global search algorithms for the network's weight vector.

[6] "Toward global optimization of neural networks: A comparison of the ...." https://www.sciencedirect.com/science/article/abs/pii/S0167923697000407. Accessed 9 Apr. 2021.