Silicon Valley Technology Companies Stock Price Prediction - Project Report

Group3 - Ruixiang An, Xinze Li, Yifeng Xie, Ruize Xu

Western University - MEng of Electrical and Computer Engineering

Abstract

The objective of this project is to train several deep learning models to predict the daily closing stock prices of three big IT companies in silicon valley including Google, Yahoo and Apple. It is the information age nowadays, and stock prices of these technology companies can be affected by more factors. Moreover, their fluctuation can reflect or serve as a signal to the global economic situation in multiple ways.

I. Introduction

A. Problem Context:
   In this project, we are going to deal with the data of stock prices of several technology companies.

B. Problem Definition:
   We will predict the daily closing stock prices of Google, Yahoo and Apple in a range of about 347 days in 2016 and 2017, and then try to find the patterns behind them. The data we are using are from Kaggle. It collects historical data from 2010-2017 for the companies we will research on.

C. The Importance of the Problem:
   The stock owners will know when to sell and buy their stocks to gain profits once there is a prediction result for them. Besides, we will know more about the patterns and regularity between time and the stock prices.

D. Solution Overview:
   This project uses three deep learning methods to train models to predict the stock prices. They are LSTM, SARIMA and KNN Regression. All the models are implemented with Python.

E. Report Organization:
   Section I, outlines the detailed question, dataset we are using for this project, the main reason why we need to predict stock prices of these companies. Section II outlines the algorithms and accuracy measures we are using for this project. Section III shows the

process of how we will train the models one by one, including the data preparation, tuning process, validation process and so on. Section IV shows the result we get from the trained models. Then section IV concludes our final result and the suggestions we can give to stock traders. Finally, section VI lists the references and the links we use during the project.

II. Background

A. Algorithms Introduction:

a) LSTM: LSTM stands for Long short-term memory is based on recurrent neural network (RNN) architecture. LSTM networks are suitable in many different jobs like classifying, processing, and sequence prediction. The challenge we are dealing with is predicting the future closing price for a stock which is just like making a prediction based on time series data a.k.a sequence prediction problem.LSTM has a property exceeding the standard feed-forward neural networks and RNN, which is feedback connections. From the image2.1, we can see how feed-forward neural networks like RNN works when generating the output.

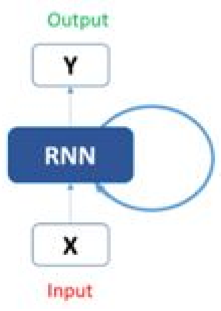


Image 2.1

The architecture of LSTM consists of multiple units including cell, input gate, output gate, forget gate. The cell is responsible for tracking the dependencies between the elements in the input sequence. Input gates have control of what value can get into the cell, forget gates have control of the value can be left in the cell or not, output gates have control of which value can be used to generate the output. Image 2.2 shows how LSTM works with different gates and cells.

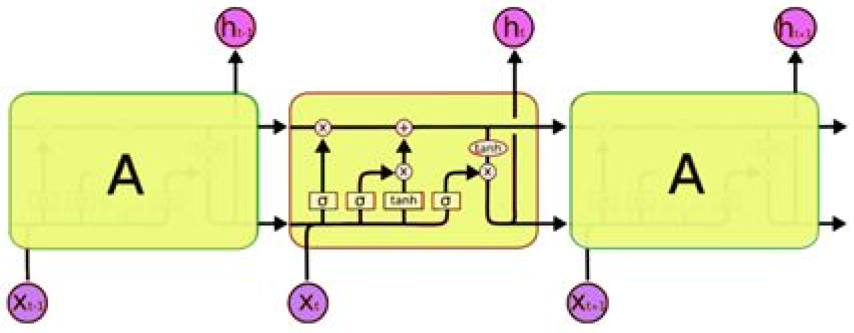


Image 2.2

The architecture of stack LSTM is shown in Image 2.3. A stacked LSTM architecture can be seen as a combination of multiple LSTM layers. The above LSTM layer can produce a sequence of output data to the LSTM layer below. In other words, the stack LSTM can provide one output for each time step instead of output for all the time steps.

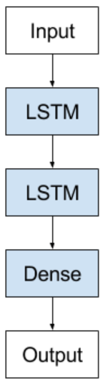


Image 2.3

b) SARIMA: Seasonal Autoregressive Integrated Moving average(SARIMA), is an extension of the ARIMA model that explicitly supports on the time series data with seasonality. Autoregressive integrated moving average, or ARIMA, is one of the most commonly used time series prediction algorithms. ARIMA model is a subset of linear regression models that uses the historical data of the target value to predict its future values. It is specified by three ordered parameters p, d, q, where p is the lag order of the autoregressive model (AR), d is the degree of differencing, and q is the order of moving average (MV). The equitation is shown as below.

$$ARIMA(p,d,q) \qquad (1)$$

As we know, ARIMA expects the stationary data series. But what if a time series data with a repeating cycle, or even the seasonality is unknown? ARIMA requires the procedures to remove the seasonal components. But with SARIMA, or seasonal ARIMA, which is an extension of ARIMA that includes four additional seasonal terms in the ARIMA, we can directly control its seasonality. SARIMA denotes as follow:

$$ARIMA(p,d,q)(P,D,Q)m \qquad (2)$$

The upper-case P, D, Q has exactly the same meanings as lower-case p, d, q in the ARIMA model, but we find optimal P, D, Q values in a single seasonal period. Why is seasonality important? We know Black Friday and Christmas day every year since it is fixed so that we could prepare the holiday related items in advance. In the business world, companies could prepare and adjust their resources to make their business succeed during the holiday period. In real life, we do not always know the future or seasonality of the things that we are working on, especially in the stock market. But we could look past the cyclicality and if we know the periodical fluctuation of the stocks we are holding, we have more chances to gain profits from it. As a result, SARIMA is extended from ARIMA with seasonality features to make a better prediction on time series.

c) KNN Regression: The model used is K-nearest neighbor regression. Being similar to KNN classification, the output is calculated based on a number of nearest neighbors. Similar to KNN classification, it averages the observations in the same neighborhood and the size of this neighborhood depends on the value K. Changes in K value may affect how the model reacts to each neighborhood. The goal is to select the K that minimizes the errors.

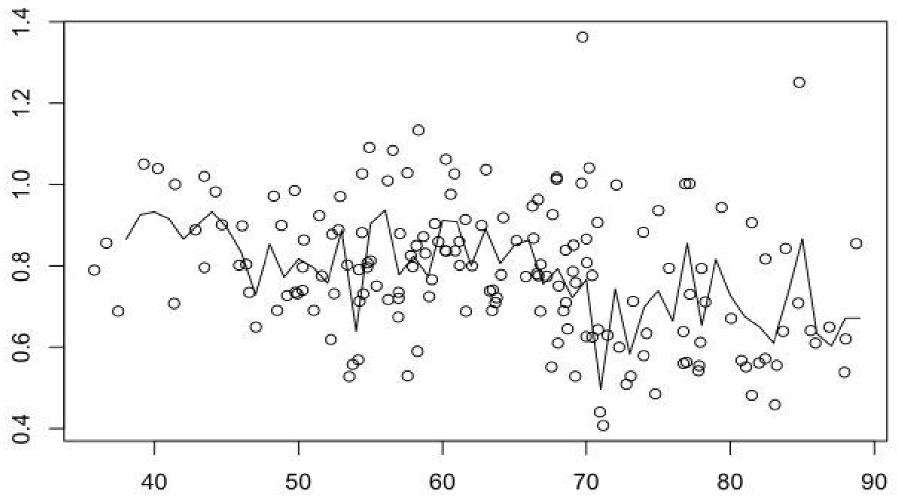


Image 2.4

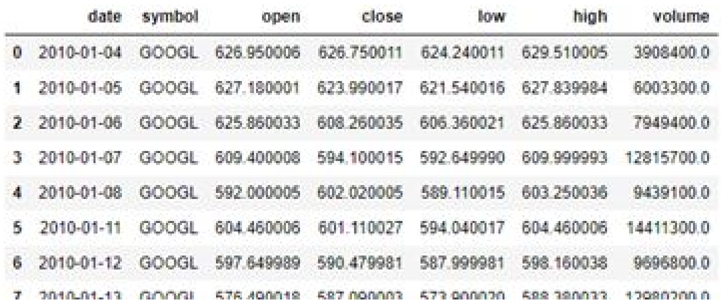
B. Accuracy Measures:

We plan to use mean squared error(MSE) and root mean squared error(RMSE) as the accuracy measures for all of the three models. MSE measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. RMSE is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed.

III. Process

A. Data Preprocessing:

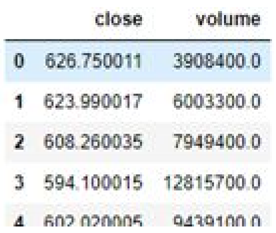
We are considering predicting the future closing price for a stock so the data will be used as close and volume in our dataset. Here is the dataset for google stock as an example. Which is shown in image 3.1.



	date	symbol	open	close	low	high	volume
0	2010-01-04	GOOGL	626.950006	626.750011	624.240011	629.510005	3908400.0
1	2010-01-05	GOOGL	627.180001	623.990017	621.540016	627.839984	6003300.0
2	2010-01-06	GOOGL	625.860033	608.260035	606.360021	625.860033	7949400.0
3	2010-01-07	GOOGL	609.400008	594.100015	592.649990	609.999993	12815700.0
4	2010-01-08	GOOGL	592.000005	602.020005	589.110015	603.250036	9439100.0
5	2010-01-11	GOOGL	604.460006	601.110027	594.040017	604.460006	14411300.0
6	2010-01-12	GOOGL	597.649989	590.479981	587.999981	598.160038	9696800.0
7	2010-01-13	GOOGL	576.490018	587.090003	573.900020	588.380033	12980200.0

Image 3.1

Here is the data that we will use, which is shown in image 3.2.



	close	volume
0	626.750011	3908400.0
1	623.990017	6003300.0
2	608.260035	7949400.0
3	594.100015	12815700.0
4	602.020005	9439100.0

Image 3.2

B. Models Training, Tuning and Validation:

a) LSTM:

1) Training: The code we used is based on Keras deep learning API. We need to know that the input shape of LSTM is always a three-dimensional array. The first dimension is batch size which can be explained as the number of samples in the two-dimensional database. The second dimension is time steps which can stand for the time we want to use to go back. The third dimension is the input dimension which can represent the number of units in one input sequence. For the output data dimension, it depends on another argument called return sequences. This argument can determine the output whether return at each time step or the final time step. In conclusion, the output could be a two-dimensional array or three-dimensional array based on return sequence argument. There is another concept called validation set used when processing LSTM. It can be considered as a part of the training set and used to evaluate loss and metrics at each epoch. Image 3.3 indicates a brief description of the constructure.

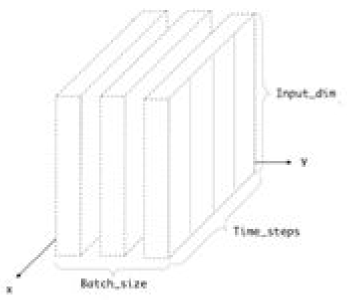


Image 3.3

2) Tuning: Hyperparameter can be thinking like a configuration for a model that value cannot be estimated from data. It works like a setting of an algorithm and it can be changed to optimize the performance and prevent overfitting. This is the principle of what we are trying to do with hyperparameter tuning. There are two commonly used methods to deal with hyperparameter tuning. One is grid search and the others one called random search. For the project we are working on we decide to use grid search. The idea of how grid search works are basically first defining a subset of the candidate values for each hyperparameter. Second, training all the possible combinations. Third, each possible model got evaluated on the validation set. Finally, we sort out the test result for each combination and use the best combination as the input. Another important technique was used since we are working on grid search called early stopping and callbacks. Because we need to evaluate all the combinations for all the hyperparameters we set before, it is going to take too long to get the final evaluation. The technique of early stopping and callbacks is to track some measure in this case we are tracking the validation loss. Once the stopping criterion is satisfied, we stop the training process, write it into a file and jump to another combination.

   Here is how we do the tuning process, we first define a set of hyperparameters which include an additional layer flag, number of neurons, number of batch size and the dropout point. The layer flag indicates True and False. Number of neurons include 128 and 256. batch sizes include 32, 64 and 128. Dropout points have value 0.1, 0.2 and 0.3. The layer flag can be used to determine if the model needs two or three LSTM layers. The dropout is a technique used to tackle overfitting, which is the fraction of the neurons to drop. Then we build the model and test it with each combination. Then we will try to get abbest number of combinations N. After the Nth combination has been made we show the combination with the highest score. Then we train our model using the best combination. And make predictions with the model we had built.

3) Validation: For the task we are working on, we use 25 days as the time step in LSTM. Each stock has 1762 lines of data in the dataset after the data normalization, and the training set and test set divide to 80% to 20%. So, the output days will be 347. These are the days that we are predicting in LSTM.

b) SARIMA:

1) Training: Our goal is to build a SARIMA model with the optimal values of the hyperparameters to make the prediction. Beyond the basic data preparing and cleaning process, we first applied the AUTO ARIMA model and seasonal decomposition analysis to determine the optimal values of the related hyperparameters. Eventually, we could complete the prediction and evaluate the results. The process is shown as image 3.4 below.

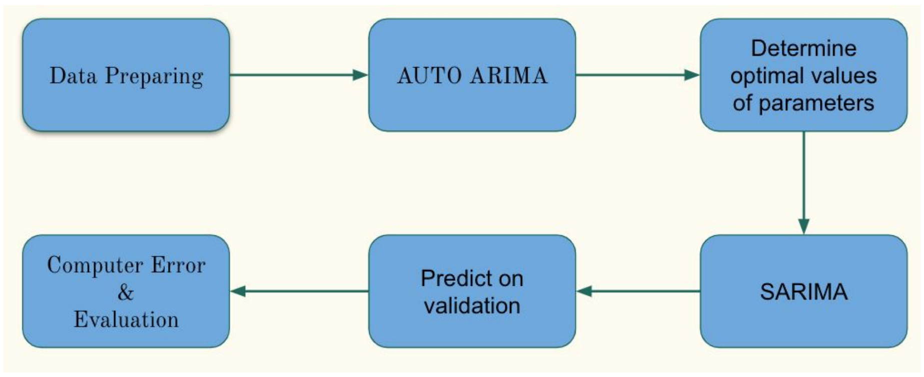
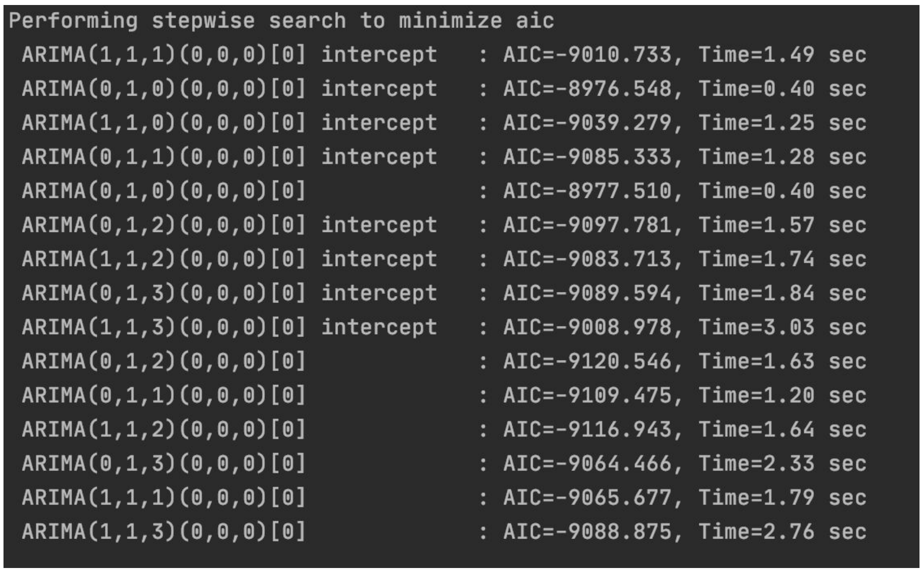

Data Preparing
AUTO ARIMA
Determine optimal values of parameters
SARIMA
Predict on validation
Computer Error & Evaluation

Image 3.4

It is essential to confirm the values of the hyperparameters p, d and q to build ARIMA models. One of the manual computation ways is to analyze the autocorrelation function (ACF) and Partial autocorrelation function (PACF) to find the most optimal value of p and q, and apply differencing and stationary analysis to get the value of d. However, AUTO ARIMA provides an automatic way to get the values of the optimal hyperparameters. AUTO ARIMA applies different tests, such as AIC and BIC, to determine the differencing order d. As a result, we are able to define a range of p, d, q to fit the ARIMA model to find the optimal values of the hyperparameters. Additionally, AIC and BIC are very complex statistics models, and since we only need them to choose the optimal values, we only need to know that the lower AIC or BIC is better. Therefore, AUTO ARIMA is ready to build.

2) Tuning: Initially, we have defined a large range of p, d, q, such as from 0 to 100. However, the returned results are always between 1 to 5 on different stocks. Therefore, after trying many ranges, we thought to set the maximum d range to 6, and set 0 to 12 for both p and q are reasonable and safe ranges. As the ranges and methods have been confirmed, we implemented the model by using python pmarima libraries to build the AUTO ARIMA model along with the parameter values. The automatic process of AUTO ARIMA using Yahoo stock (YHOO) is shown as the below image 3.5.



```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-9010.733, Time=1.49 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-8976.548, Time=0.40 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-9039.279, Time=1.25 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-9085.333, Time=1.28 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-8977.510, Time=0.40 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-9097.781, Time=1.57 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-9083.713, Time=1.74 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=-9089.594, Time=1.84 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=-9008.978, Time=3.03 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=-9120.546, Time=1.63 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=-9109.475, Time=1.20 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=-9116.943, Time=1.64 sec
 ARIMA(0,1,3)(0,0,0)[0]             : AIC=-9064.466, Time=2.33 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=-9065.677, Time=1.79 sec
 ARIMA(1,1,3)(0,0,0)[0]             : AIC=-9088.875, Time=2.76 sec
```

Image 3.5

From the sample picture, we can use 0, 1, 2 as p,d,q values since the AIC is -9120.546, which is the lowest in all the combinations.
We could apply the same AUTO ARIMA model as before once we clarify the seasonality of the series. As a consequence, the key is to find a single period and the seasonality of the data series. The only remaining hyperparameter m denotes the number of time steps or observations, which could be used to reflect a single period. In order to find a reasonable m, we have decomposed the data series. There are two decomposition models, additive model and multiplicative model, and we have used statsmodels API of python to implement the decomposition process. The equation of AAPL stock is shown below.

Additive Model

Y[t] = T[t] + S[t] + e[t]   (3)

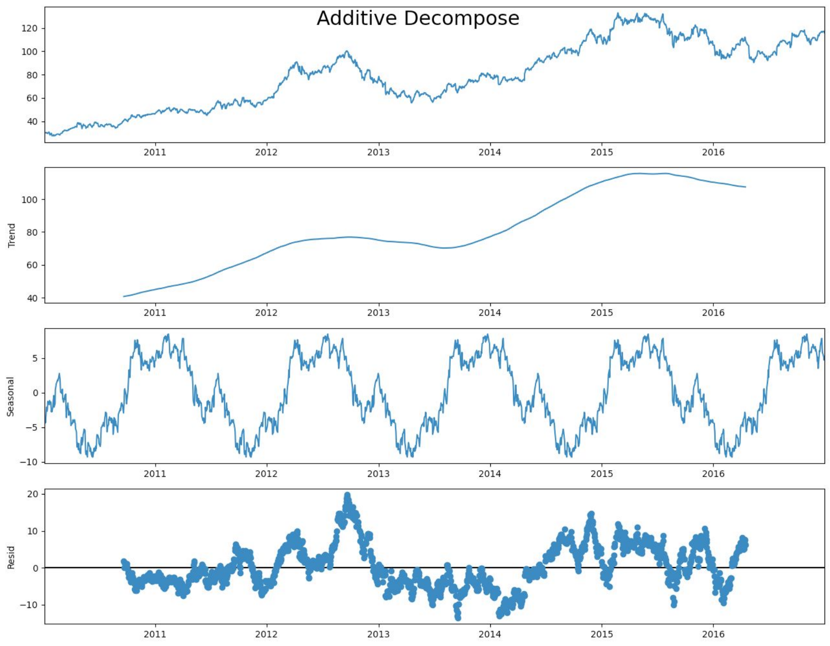


Image 3.6

Multiplicative Model

Y[t] = T[t] S[t] e[t]   (4)

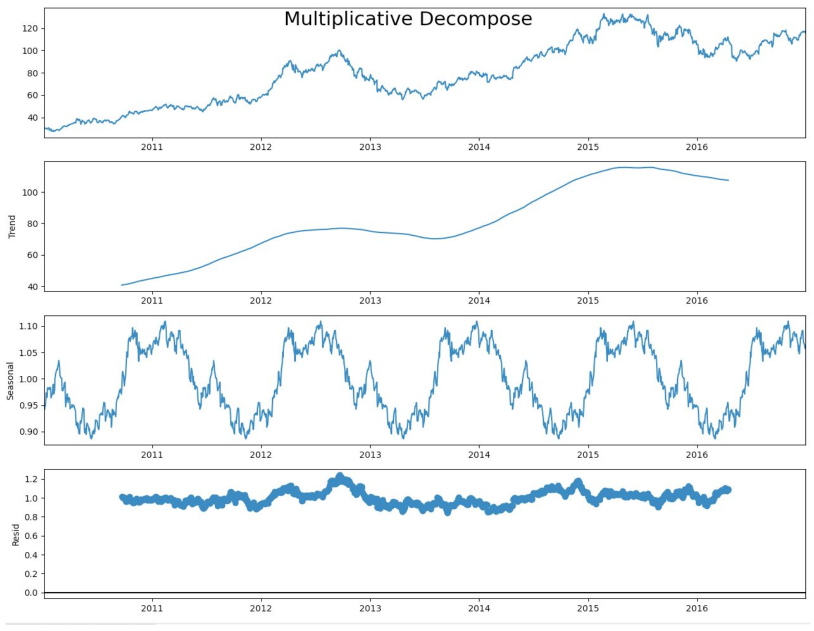


Image 3.7

From the sample figures, both models have similar trend and seasonal plots, but the residuals are different. From the residual plot, we could conclude that the multiplicative is more suitable for our data. Besides, the seasonality plots

explicitly show a 12-month period cycle, so we used 12 as the value of m. Thus, an m of 12 for monthly data suggests a yearly seasonal cycle.
Since the period of seasonality m is confirmed, we applied the same process as previous AUTO ARIMA but in a single period to define the most optimal P, D, Q values. At this point, we have the optimal values of all seven hyperparameters, and therefore the SARIMA model is built.

3) Validation: Since we are predicting 1-year(2016) stock prices, 253 days due to the source dataset, we split the training set and test set by a ratio of 85% to 15%. And the time step in Sarima is set to be 1-week after compared to 1-month and 1-day.

c) KNN Regression:

1) Training: As it is shown below in Image 3.6, the general process consists of five steps. The final goal is to determine the best size of neighborhood that minimizes the errors. The prices are normalized using the default Minmax scaler, and x-axis is the number of days.

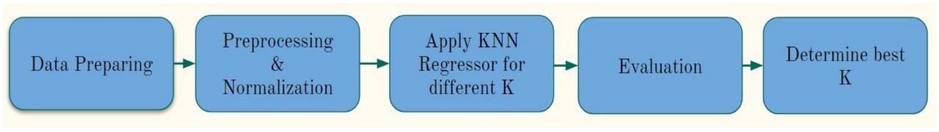

Data Preparing
Preprocessing & Normalization
Apply KNN Regressor for different K
Evaluation
Determine best K

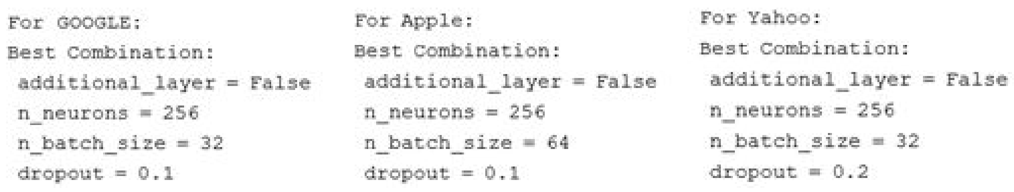
Image 3.6

2) Tuning: To get the best results for all three stocks, KNN regressor model is applied in a loop where K iterates from 1 to 150. Both MSE and Root MSE are calculated during the process. The best result is the K with the smallest error, and it will be stored and plotted.
3) Validation: The model predicts the stock prices of the next 10 days. The x-axis starts from day 0 to 1762, so the predictions are up to day 1763. The training set and testing set ratio is 8 to 2.

IV. Results

A. LSTM

After tuning the model. The best combination for Google stocks, Apple stocks and Yahoo are shown in the image 5.1.



```
For GOOGLE:                      For Apple:                       For Yahoo:
Best Combination:                Best Combination:                Best Combination:
 additional_layer = False         additional_layer = False         additional_layer = False
 n_neurons = 256                  n_neurons = 256                  n_neurons = 256
 n_batch_size = 32                n_batch_size = 64                n_batch_size = 32
 dropout = 0.1                    dropout = 0.1                    dropout = 0.2
```

Image 5.1

Image 5.2, 5.3 and 5.4 separately shows the results of our LSTM about the 3 stocks. They all predict the closing prices for the future 347 days since January 1, 2016.

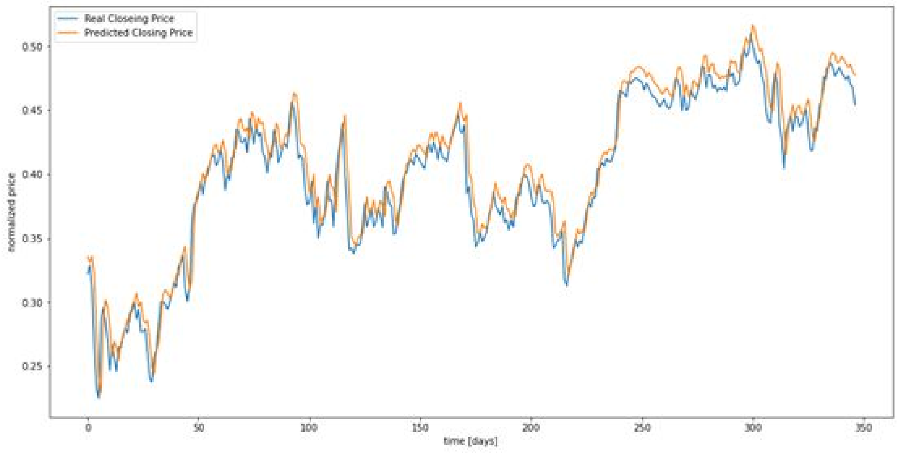


Image 5.2 LSTM RESULT OF GOOGLE

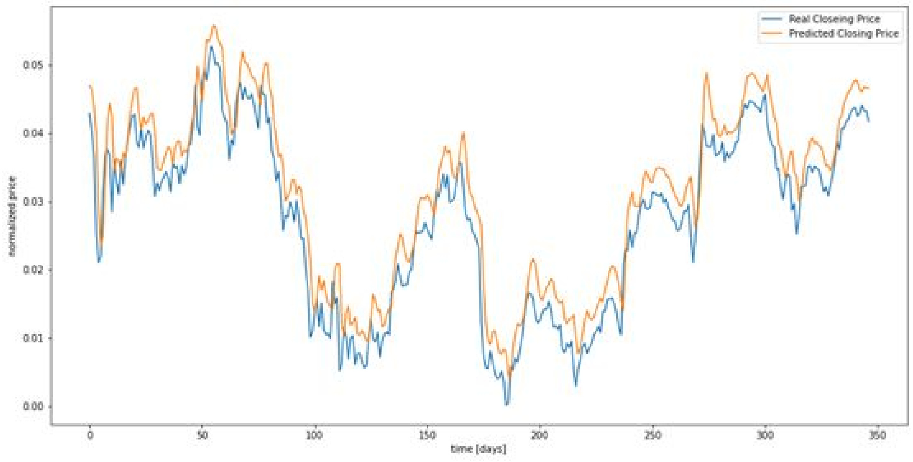


Image 5.3 LSTM RESULT OF APPLE

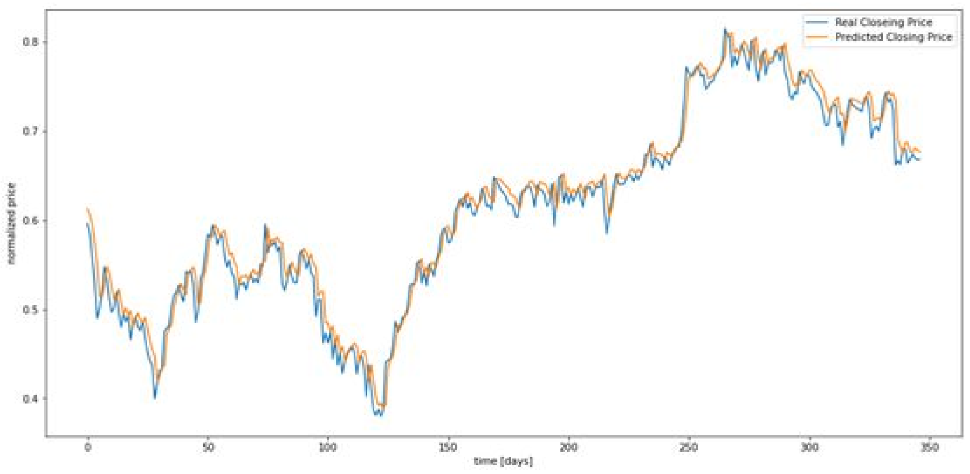


Image 5.4 LSTM RESULT OF YAHOO

The table 5.1 gives the accuracy result of all the three stocks.

	GOOGLE	APPLE	YAHOO
MSE	0.00023	0.0009	0.00033
RMSE	0.02	0.02	0.01

Table 5.1

The result plots show that our model works very well for predictions of all the three stocks, especially for Google and Yahoo.

B. SARIMA

As we repeat the previous process for three different stocks Google, Apple, and Yahoo, the prediction plots with the optimal values of the hyperparameters of SARIMA and error measures are shown below.

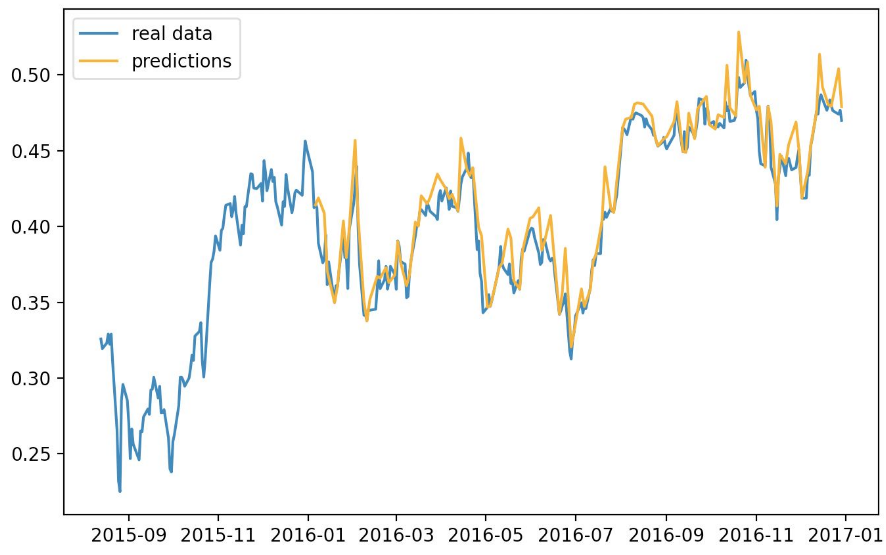


Image 5.5 SARIMA RESULT OF GOOGLE

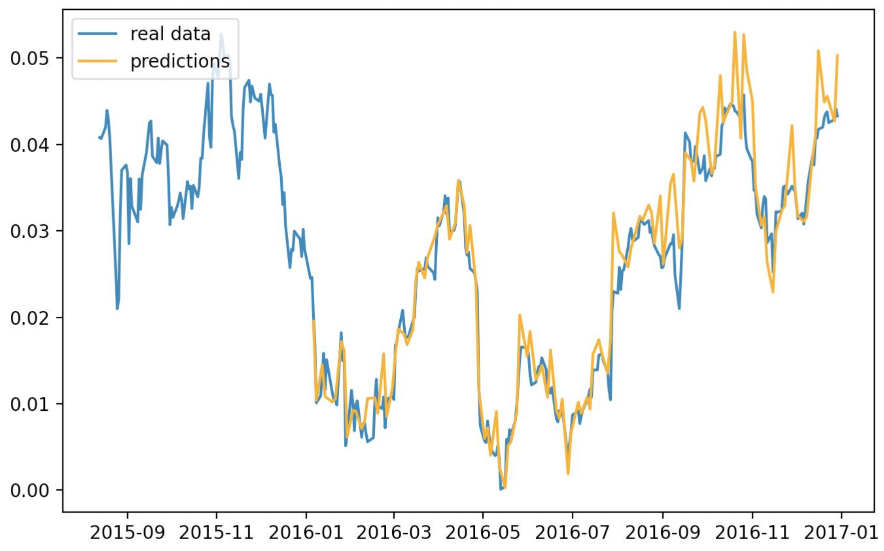


Image 5.6 SARIMA RESULT OF APPLE

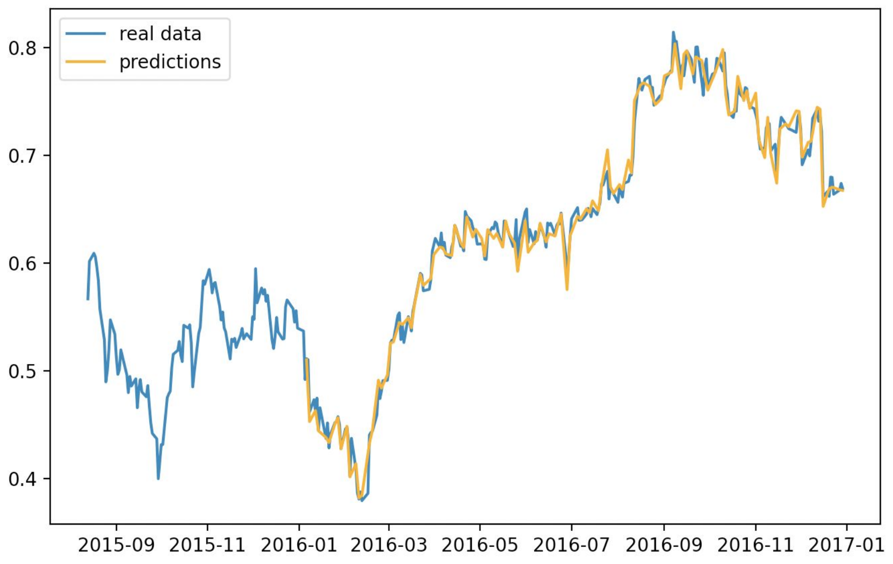


Image 5.7 SARIMA RESULT OF YAHOO

The table 5.2 gives the accuracy result of all the three stocks.

	GOOGLE	APPLE	YAHOO
MSE	0.000169	0.000136	0.00018
RMSE	0.013	0.0116	0.012

Table 5.2

The blue curve is the historical data actual data, the orange curve is the predicted data. We can observe from the plots that the predictions and the real data are almost overlapped at the beginning and then go varies. As the prediction time increases, the error increases. Therefore, based on our experiment, we can conclude that the SARIMA model is suitable for stock price prediction, especially in short term forecasting.

C. KNN Regression

Here are the plots showing the results of the KNN model. As you may notice, there is a significant drop in both Google and Apple's plots, while the above two methodologies do not have that drop. The reason is mainly due to that the above two methodologies used adjusted data for splits from the raw data, while this methodology used the raw data, since we would like to explore if there is any difference between the prediction outcomes of these datasets. We found that most of the trends in the plots remain the same, but the fluctuation becomes larger, which might cause decrease in accuracy.

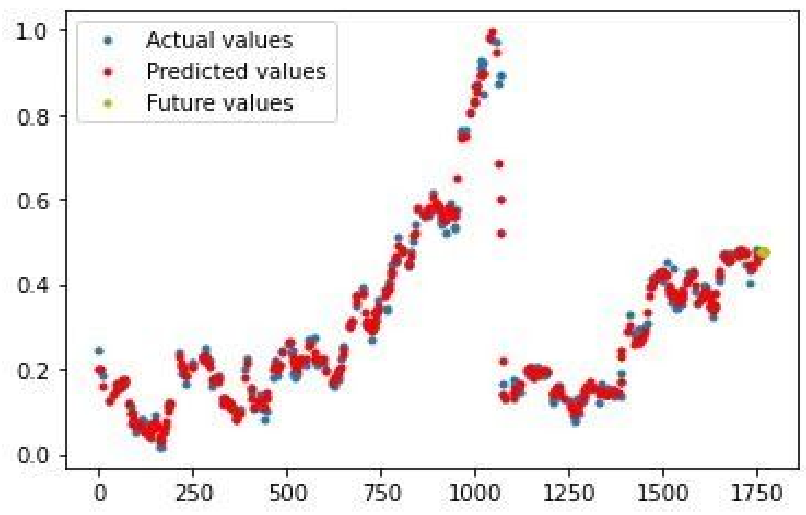


Image 5.8 KNN RESULT OF GOOGLE

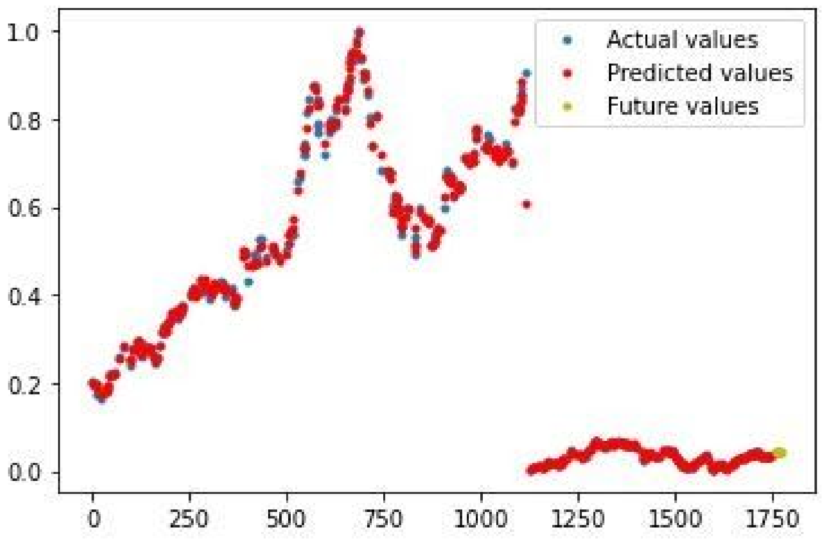


Image 5.9 KNN RESULT OF APPLE

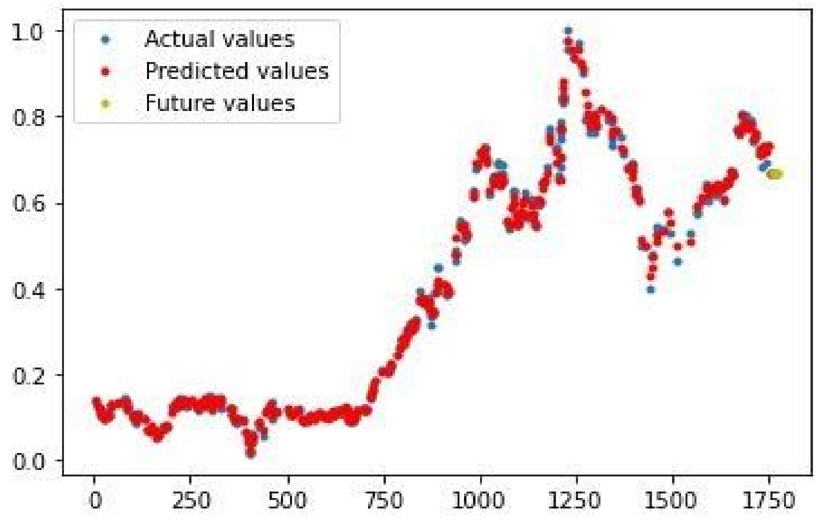


Image 5.10 KNN RESULT OF YAHOO

The table 5.3 gives the accuracy result and the value of K for all the three stocks.

	GOOGLE	APPLE	YAHOO
MSE	0.00099	0.00034	0.00014
RMSE	0.03142	0.01867	0.01196
K	10	3	3

Table 5.3

As it is shown, for Google, the best result is produced when K is 10, and for both Apple and Yahoo, the best K is 3. Yahoo has the lowest error among all three mostly because there are no drastic price changes that happened within a few days. The future values are calculated based on previous neighboring points. It can show a trend of the future stock prices.

V. Conclusions

In this project, we successfully implemented LSTM, SARIMA and KNN to predict the daily closing prices of three big technology companies in silicon valley - Google, Apple and Yahoo in a year since January 1, 2016.

After all the training and analyze, here are what we can conclude from the results.
LSTM is good for long term predictions, and it can perform much better if there are few trend changes, we think this is a good methodology for those companies that have founded for a long time or have huge assets. Because these kinds of companies tend to be more mature,

and it is easier for people to predict the trends. Traders do not need to worry too much about the trend fluctuations.

SARIMA is good for short term predictions, it can increase errors over the long term. We can see from the image 5.5, 5.6 and 5.7 that at the very beginning of the testing set, the orange line and blue line seems to be very close. However, as the time goes on, they start to separate from each other. If traders are only seeking for a very close range of time in the future, say 1 week or 1 month, we would suggest this model.

KNN regression has a relatively low accuracy, although it can still reach an accuracy which does not look bad, we still suggest that only use it for predicting general trendession since we have better options for predicting the detailed prices.

VI. References

New York Stock Exchange, S&P 500 companies historical prices with fundamental data accessed on Nov 20, 2020, from https://www.kaggle.com/dgawlik/nyse

Brownlee, J. (2019, August 14). Stacked Long Short-Term Memory Networks. Retrieved November 30, 2020, from https://machinelearningmastery.com/stacked-long-short-term-memory-networks/

Brownlee, J. (2020, August 25). Use Early Stopping to Halt the Training of Neural Networks At the Right Time. Retrieved November 30, 2020, from https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

Long short-term memory. (2020, November 10). Retrieved November 30, 2020, from https://en.wikipedia.org/wiki/Long_short-term_memory

Nabi, J. (2019, March 16). Hyper-parameter Tuning Techniques in Deep Learning. Retrieved November 30, 2020, from https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8

Understanding LSTM Networks. (n.d.). Retrieved November 30, 2020, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Verma, S. (2020, August 29). Understanding Input and Output shapes in LSTM: Keras. Retrieved November 30, 2020, from https://shiva-verma.medium.com/understanding-input-and-output-shape-in-lstm-keras-c501ee95c65e

Worcester, P. (2019, June 06). A Comparison of Grid Search and Randomized Search Using Scikit Learn. Retrieved November 30, 2020, from https://blog.usejournal.com/a-comparison-of-grid-search-and-randomized-search-using-scikit-learn-29823179bc85

K-nearest Neighbours Regression, bookdown.org, accessed on Nov.24 2020, from
2 K-nearest Neighbours Regression | Machine Learning for Biostatistics