Stock Price Prediction Using Machine Learning and Deep Learning Models

Youngseong Kim

Jieung Park

Jiwon Jun

Simon Fraser University

CMPT 353: Computational Data Science

December 10, 2021

Table of Contents

1. Abstract

The primary goal of the project is to observe the possibility of reliable stock price prediction using the time series stock price data. We also aim to observe whether it can apply the predicted stock movement to gain a significant profit in buying and selling the stock. The technical analysis processes are conducted through developing several Machine Learning or Deep Learning-based models, as they are considered the most efficient ways to process such predictions. This project focuses on implementing Linear Regression, K-Nearest Neighbours (KNN) Regression, Random Forest Regression, and Long Short-Term Memory (LSTM) models. The accuracy scores of the individual models are then measured and compared to determine which model outperforms in predicting the precise price movement.

2. The Problem

Stock trend analysis and prediction continuously have been a popular topic in global financial markets and technical research. There are numerous factors and conditions that determine the daily stock price, which makes the stock market a complex system of fluctuations where accurate price forecasting is challenging.

There have been various attempts and studies to scientifically predict the stock price movement, but it still remains an unresolved challenge to predict the future stock values due to its natural volatility. Meanwhile, some researchers state that the stock price movements are chaotic rather than being totally random as the market is mostly dependent on the investors' rational behaviours [1]. This implies that future stock price predictions may be possible through the deliberate analysis of historical data with the support of various technical mechanisms. An efficient method to process such predictive analysis is machine learning or deep learning. Both techniques are in the field of artificial intelligence (AI) which provides the system to automatically learn the information to complete various technical tasks. Thus, we combine the AI and the technical data analysis to examine whether the movement in stock prices can be predicted regardless of other external factors such as the economic or social events, if only the pattern of fluctuations on the chart could be found.

3. Datasets

The experimental data in this project are the actual historical data downloaded from our data crawler program. Our crawler uses the FinanceDataReader module, which is an open-source finance data crawler made by FinanceDataKR [2]. We downloaded some of the stock data and indices put out by 'tickers' such as AAPL, IXIC (Nasdaq Indices), N225 (Nikkei 225). In our project, we decided to use Nasdaq (IXIC) index data and Nikkei (N225) index data for our stock price data as shown in Figures 1 and 2. The data we used as input parameters to train and test our model include: 1. Date, 2. Open price 3. Close price 4. Low price 5. High price, 6. Adj Close, 7. Trading volume. We gathered stock prices data starting from 1985 to increase the modelling speed without affecting the result accuracy of the models.

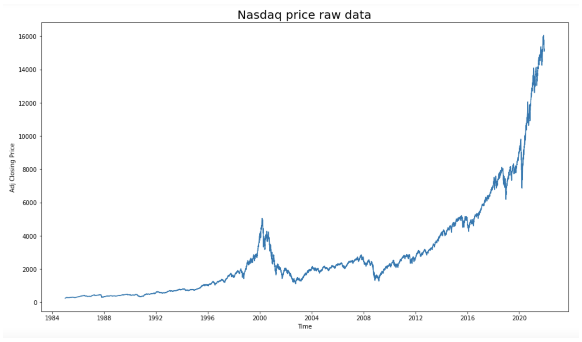


Figure 1. Nasdaq (IXIC) Stock Price Raw Data

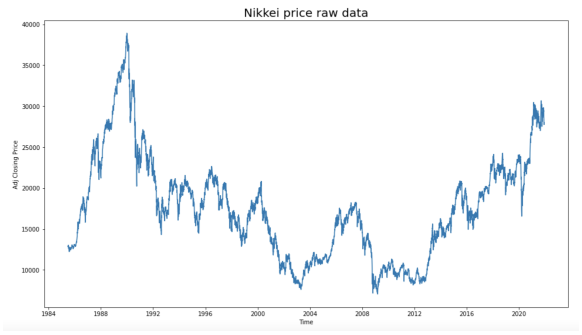
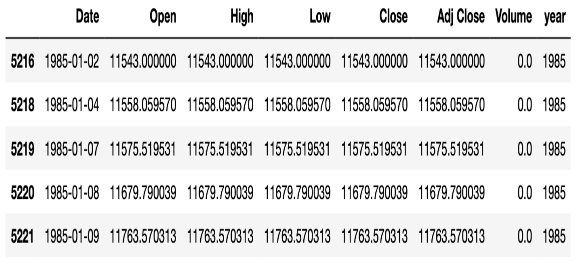


Figure 2. Nikkei (N225) Stock Price Raw Data



	Date	Open	High	Low	Close	Adj Close	Volume	year
5216	1985-01-02	11543.000000	11543.000000	11543.000000	11543.000000	11543.000000	0.0	1985
5218	1985-01-04	11558.059570	11558.059570	11558.059570	11558.059570	11558.059570	0.0	1985
5219	1985-01-07	11575.519531	11575.519531	11575.519531	11575.519531	11575.519531	0.0	1985
5220	1985-01-08	11679.790039	11679.790039	11679.790039	11679.790039	11679.790039	0.0	1985
5221	1985-01-09	11763.570313	11763.570313	11763.570313	11763.570313	11763.570313	0.0	1985

Figure 3. Nikkei Stock price data features

Our stock market dataset is splitted into training and test datasets with a test size of 20% of the total dataset. We will be training our model on historical pricing data using the noise filtered adjusted close prices and list of data points such as open, high, low price, and trading volume.

3.1 Data Cleaning

Financial stock market data, for various reasons, frequently contain missing values. One reason may be due to stock markets closing for holidays, which prohibits the daily stock prices from always being observed. The omission of observations creates information gaps, making it difficult to predict the following day's stock market prices.

One of the frequent methods to deal with the missing data is to exclude all missing values and only use complete records. An alternative method is imputing the missing values with the mean, median, midpoint values, etc. of the known nearby data. To carefully apply both techniques into the cleaning data application, if the percentage of the missing data is less than 5%, remove the missing observations. Otherwise, impute the missing data with the midpoints between the known nearby data by applying the interpolation method.

3.2 Noise Filtering

The noise in the stock market usually represents the short-term price fluctuations caused by volatile intraday trading. The raw financial data is overflowing with noises, which may consequently influence the result of data analysis. To enhance the accuracy of the trend analysis, it is important that we take appropriate signal processing operations before moving on to training the model. The two noise filtering algorithms we used to uncover the true values are LOESS (Locally Weighted Scatterplot Smoothing) and rolling mean.

LOESS is a popular technique used in regression analysis to smooth the data curve, which disregards the impacts of noise. Thus, the trend analysis of LOESS is much less complex compared to other signal processing algorithms. Although LOESS is typically used in noisy plots where data points have weak correlations, our data requires an inclusive consideration of the noise as the outliers are hard to be defined in stock data. Thus, LOESS is not an applicable noise filtering technique in our data, since we require a tool to capture the detailed stock trend change instead of a highly-smoothed curve.

On the other hand, rolling mean, also known as moving average, is a commonly used noise filtering algorithm to identify the stock market trend direction. Rolling mean of the data smoothes the data curve by constantly calculating the average of past price data over the specific timespan, which is intended to mitigate the impacts of random, short-term price fluctuations. The shorter the time period applied to calculate the average, the more sensitive it is to noises of abrupt price changes. For the analysis, we used 20-day averages for short-term movements which are mainly for swing traders, and 100-day averages to predict long-term movements for traders such as position traders, portfolio traders. As illustrated in Figure 4, the data curves of the rolling means tend to follow the stock trend with reasonable reflection of the noises.

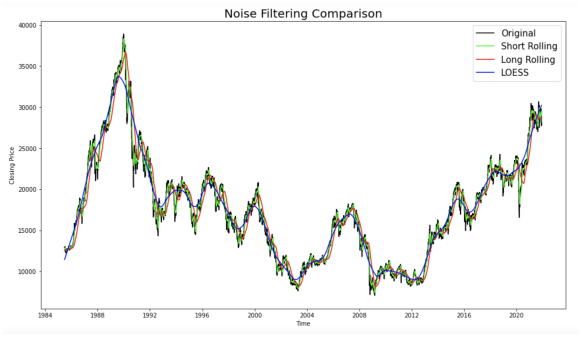


Figure 4. Nikkei 225 Noise Filtering Comparisons

3.3 Normalization

Due to the variety of measurement units of different stock index data, the attribute data are normalized to fall within the range to avoid the impact of different measurement units. Also, a normalization speeds up the gradient descent to find the optimal prediction and improves accuracy. Through the Min-Max-Scaler normalization process, the data is scaled to the range [-1, 1].

4. Modeling

4.1 Machine Learning

Machine learning is the study of computer algorithms that automatically improves accuracy by learning the data. A well-trained machine learning model is capable of predicting the output for any new input values. Theoretically, the stock market behaves similarly, as they are based on several inputs to predict the accurate price outputs. With the premise that the stock price data already discounts all publicly available information, we approach several machine learning models to train the data. As stock price prediction operates on the labeled data of correct price values, machine learning algorithms based on supervised learning are suitable. Any regression-based algorithms may also be acceptable since we aim to predict the price as the continuous quantity.

4.1.1 Linear Regression

Machine learning packages in “scikit-learn” implement a linear regression model, LinearRegression(), that fits a linear model with coefficients to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation [3]. It is utilized in business, science, and other fields where predictions and forecasting are relevant. The following graphs in Figures 5 and 6 represents how well the linear regression model forecasts the stock's adjusted closing values.

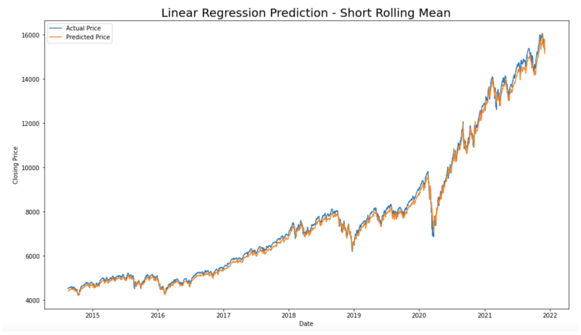


Figure 5. Nasdaq Short Rolling Mean Linear Regression

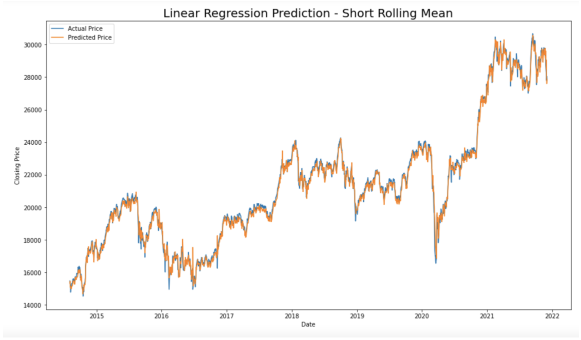


Figure 6. Nikkei Short Rolling Mean Linear Regression

4.1.2 K-nearest Neighbour Regression

K-nearest Neighbour (KNN) regression model is one of the simplest and best-known non-parametric methods of predicting values based on the average values of the nearby k data points. Machine learning module packages in "scikit-learn" provide a KNeighborsRegressor() model that predicts the target by local interpolation of the targets associated with the nearest neighbours in the training set. The following graphs in Figure 6 and 7 show the predicted trend calculated by averaging the k nearest neighbours in our training dataset, where k=5. Smaller k (5~20) is used if the stock prediction is for short-term investment. Larger k (100~200) is chosen for long-term investment.

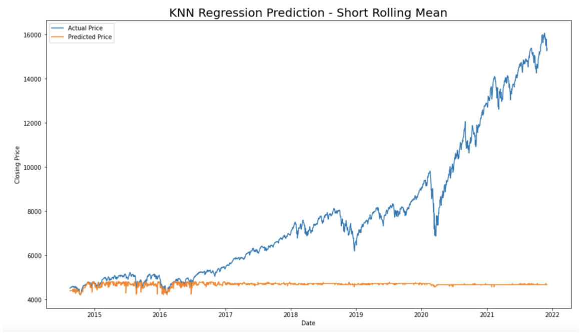


Figure 7. Nasdaq Short Rolling Mean KNN Regression

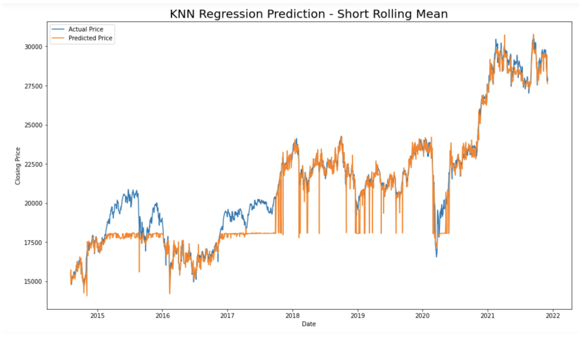


Figure 8. Nikkei Short Rolling Mean KNN Regression

4.1.3 Random Forest Regression

Random Forest regression is one of the most powerful supervised learning algorithms that uses ensemble methods for regression. Ensemble method combines predictions from multiple machine learning algorithms to make more accurate predictions than a single model regression. Random Forest fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [4]. Our model uses the RandomForestRegressor() from the "scikit-learn" package with 500 trees where the maximum depth is set to 30 and minimum samples at leaf node is 20. The parameters are determined by the ParameterGrid() which is used to identify the optimal parameter combination for training the model. The following diagram in Figures 8 and 9 illustrates the prediction from a Random Forest model.

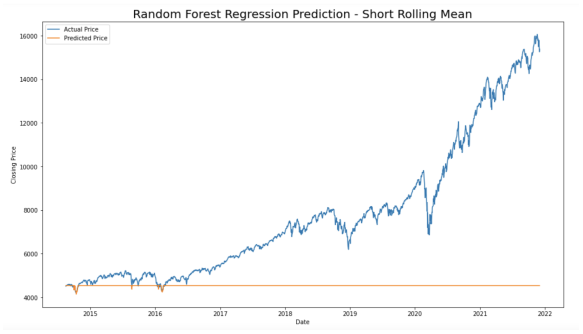


Figure 9. Nasdaq Short Rolling Mean Random Forest Regression

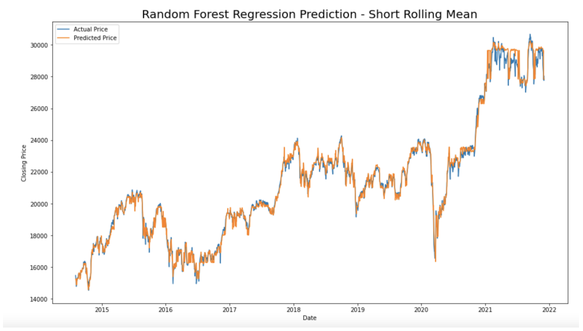


Figure 10. Nikkei Short Rolling Mean Random Forest Regression

4.1.4 Results Analysis

As shown in Figures 7 to 10, KNN and Random Forest models prediction results depend heavily on the data we used. Linear Regression was the only machine learning model with a prediction similar to the actual output, no matter which dataset we used. When KNN and Random Forest regression failed to fit the model, the prediction stopped after a certain period of the year and outputted a meaningless straight line in the plot.

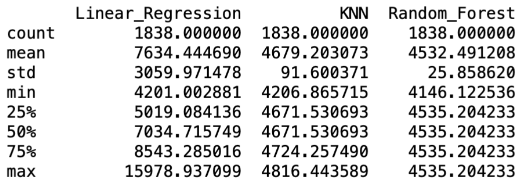
The failure of predicting the Nasdaq (IXIC) dataset from Figures 7 and 9 leads to the fundamental question – why did it fail? The answer to the question is straightforward, as KNN and Random Forest Regression cannot extrapolate outside unknown values from the training dataset. For justification, Figure 11 and 12 explains the distribution of the adjusted close price of all datasets in each model, including the raw data. Considering that our raw data has a close price range [281.58, 15828.80] where the train data ranges only up to 4839.67, it is shown that prediction from KNN model has a maximum price of 4816.44. The Random Forest model resulted the same as it has a maximum price of 4535.20. This indicates that the KNN and Random Forest regression models are only able to predict the values within the range of the training data, and thus cannot extrapolate the values outside that range. If a price value in the test dataset does not fall within the trained price range, the plot is depicted as a straight line. The failure of the stock price prediction in Nasdaq stock resulted because of the rapid increase of the stock price since 2010 which the test dataset contains the closing prices that are higher than the prices from the training dataset.

Meanwhile, the KNN and Random Forest regression with Nikkei (N225) dataset from Figures 8 and 10 shows better prediction. According to Figure 2, the training dataset includes the highest and lowest points of the entire dataset. Consequently, all values of the test data fall within the trained range, which allows the models to continue the prediction without displaying a straight line in the plot.

On the other hand, the Linear Regression model outputted the predicted values outside the range of maximum price distinguished in the training dataset. Since the linear model provides the ability to extrapolate based on trained datasets, it can be considered as one of the suitable models for predicting future stock prices.

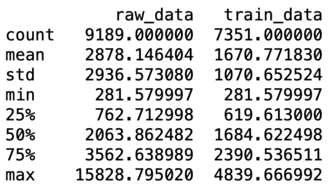
The outputs in Figure 13 and 14 display the comparison of the validation scores for all machine learning models. It is found that the Linear regression model produced the highest validation score over all machine learning models with the validation score of 0.996, while the KNN and Random Forest models resulted in negative scores for the Nasdaq data. The scores with negative values indicate that the performance of the model is arbitrarily worse, since the models were not able to infer the values outside the range of the training dataset.

For the reason that the machine learning models we used cannot always infer outside the range of the trained dataset, they are not sufficient to solve a highly advanced problem such as predicting the future stock price. KNN and Random Forest regression depend highly on the value of historical data with frequent price fluctuations in the stock market. This leads us to the use of deep learning, the evolution of machine learning.



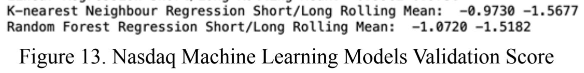
```
       Linear_Regression          KNN  Random_Forest
count        1838.000000  1838.000000    1838.000000
mean         7634.444690  4679.203073    4532.491208
std          3059.971478    91.600371      25.858620
min          4201.002881  4206.865715    4146.122536
25%          5019.084136  4671.530693    4535.204233
50%          7034.715749  4671.530693    4535.204233
75%          8543.285016  4724.257490    4535.204233
max         15978.937099  4816.443589    4535.204233
```

Figure 11. Nasdaq Machine Learning Models Descriptive Statistics



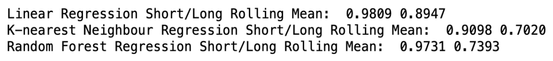
```
          raw_data   train_data
count  9189.000000  7351.000000
mean   2878.146404  1670.771830
std    2936.573080  1070.652524
min     281.579997   281.579997
25%     762.712998   619.613000
50%    2063.862482  1684.622498
75%    3562.638989  2390.536511
max   15828.795020  4839.666992
```

Figure 12. Nasdaq Raw Data Descriptive Statistics



```
Linear Regression Short/Long Rolling Mean:  0.9962 0.9750
K-nearest Neighbour Regression Short/Long Rolling Mean:  -0.9730 -1.5677
Random Forest Regression Short/Long Rolling Mean:  -1.0720 -1.5182
```

Figure 13. Nasdaq Machine Learning Models Validation Score



```
Linear Regression Short/Long Rolling Mean:  0.9809 0.8947
K-nearest Neighbour Regression Short/Long Rolling Mean:  0.9098 0.7020
Random Forest Regression Short/Long Rolling Mean:  0.9731 0.7393
```

Figure 14. Nikkei Machine Learning Models Validation Score

4.2 Deep Learning

Deep learning is a subset of machine learning that uses multiple layers of artificial neural network to imitate the behaviour of the human brain. While machine learning models are heavily dependent on the historical data and require some guidance, predictions in deep learning models are less dependent on human intervention and historical data to gain knowledge. Thus, deep learning models are capable of determining themselves if a prediction is accurate or not through the neural network. Additionally, the neural network allows the models to solve complex predictive analyses that require huge amounts of data with hidden patterns.

4.2.1 Long Short-term Memory

The Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning having feedback connections. LSTM models are widely used for sequence prediction problems and have proven to be extremely effective because they are able to store past information [5]. This is an important factor of predicting stock prices because the previous price of a stock is crucial in forecasting the future price. We used the Tensorflow library for training keras LSTM model in this project.

Our LSTM model was built and trained to predict daily indice price of “IXIC” and N225” as we provided on Figure 1 and 2. The training dataset for the LSTM model always needs to be reshaped to the form of a 3D vector (samples, time-steps, features). The parameters of keras LSTM model include: hidden nodes, hidden layer, dense layer, dropout layer, optimizer, and a loss function. The hidden_nodes represent the number of neurons which is set to 50 for a more powerful network. Four hidden layers are added to increase the complexity of the model, with a single dense layer as the output for predicting the normalized indice price. The dropout layers were added to reduce overfitting by randomly taking a portion of the possible network connections. This model was trained using Adam stochastics gradient descent optimizer and MSE loss function. The loss function explains how bad the model performed and the optimizer finds the minimum of that function. To understand the complexity and performance of the model, each indices were tested with validation_split of 0.1 and 20, 50, 100, and 150 epochs. The validation split was obtained from the last x and y data provided, ensuring that the temporal order of the data is maintained for the training.

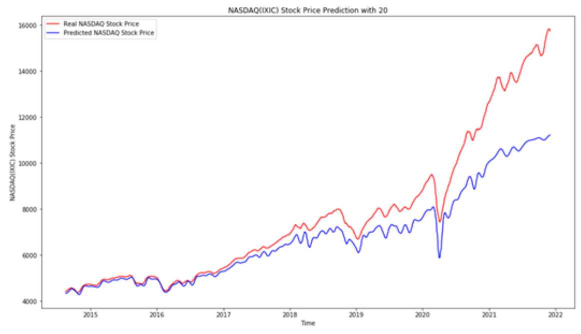


Figure 15. Nasdaq Short Rolling Mean LSTM with epoch=20

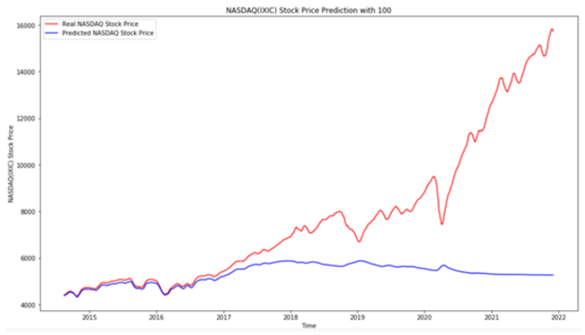


Figure 16. Nasdaq Short Rolling Mean LSTM with epoch=100

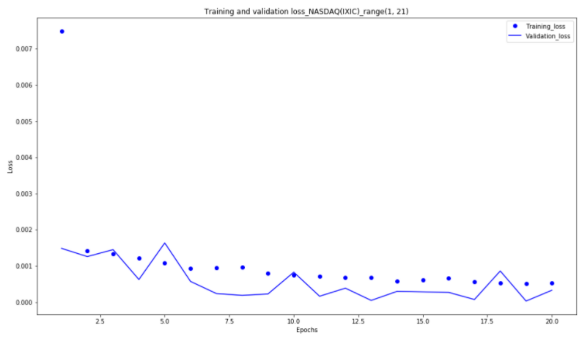


Figure 17. Nasdaq LSTM Loss Graph with epoch=20

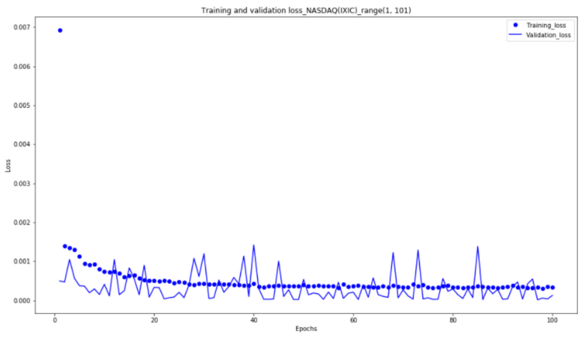


Figure 18. Nasdaq LSTM Loss Graph with epoch=100

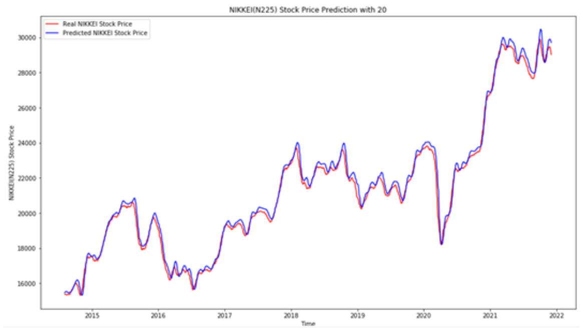


Figure 19. Nikkei Short Rolling Mean LSTM with epoch=20

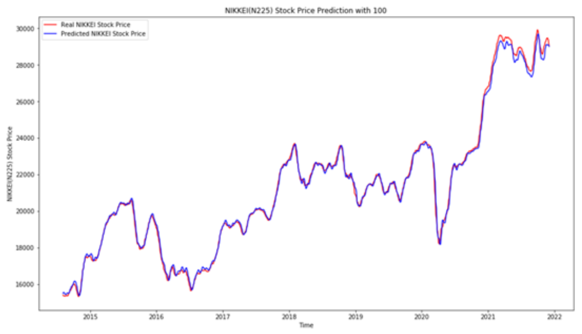


Figure 20. Nikkei Short Rolling Mean LSTM with epoch=100

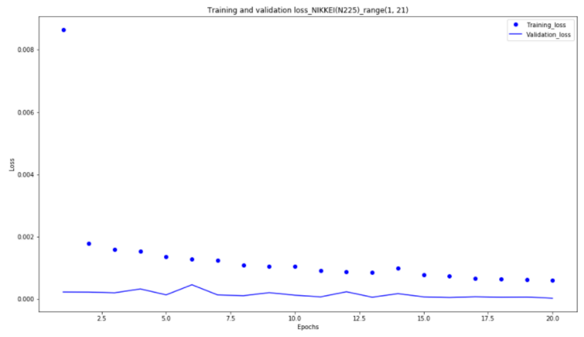


Figure 21. Nikkei LSTM Loss Graph with epoch=20

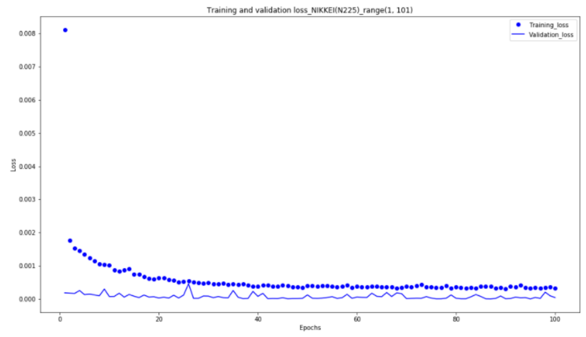


Figure 22. Nikkei LSTM Loss Graph with epoch=100

	LSTM_Epoch 20	LSTM_Epoch 100	Linear
IXIC	1453.55	3799.58	189.68
N225	311.92	185.43	511.57

Figure 23. RMSE Comparison Table

4.2.2 Results Analysis

As shown in Figure 15, use of deep learning LSTM model does not guarantee the successful prediction of the future stock market price. The model is only somewhat capable of forecasting the general trend of the price. Moreover, higher epoch does not always provide better performance. Learning too much of the training data or greater complexity of the model may end up overfitting the data which excludes significant increase or decrease of the price during the prediction as displayed in Figure 16. Using 100 epochs was concluded to be overfitting because the validation loss increased over the training loss multiple times as the training loss continued to decrease as shown in Figure 18. Whereas, the validation loss from Figure 17 with 20 epochs fluctuates near the training loss.

As the epoch increased, the predictive performance of Nasdaq data decreased, while the performance of Nikkei data showed contradictory results. According to Figures 19 and 20, the accuracy of the prediction improved as the epoch increased from 20 to 100. The improvement of the model was observed in Figure 22 as the fluctuation of validation loss approached closer to the decreasing training loss as the number of processed epochs expanded.

To measure the accuracy scores of the models, Root Mean Square Error (RMSE) were calculated for each of the data. The lower scores of RMSE relatively represent a better prediction model. From Figure 23, we can conclude that if the trend of the dataset is simple (IXIC), a linear regression model tends to predict better than the LSTM. The reason being is that complex LSTM with extra hidden layers overfits the data by excessive amounts of training compared to its actual data trend. On the contrary, LSTM shows better prediction with a more complex trend dataset (N225).

5. Limitation

During the stock prediction modelling processes, we have not considered any other features than stock market values: open, close, adj close, low, high, trading volume. If we had more time to work on our project, we would include more factors that affect the stock trends to train our models. These factors include news, inflation, economy, interest rates, political climate, demand and supply, etc.

The deep learning model's hyperparameters such as the learning rate of the optimizer, number of layers and the number of hidden units in each layer, the optimizer are extremely sensitive to the results that we obtain. However, due to lack of datasets and time, our model was set to a simple layer LSTM model. Thus, we were limited to use certain hyperparameters instead of the parameters obtained from the optimization techniques (Grid Search, Random Search).

One of the biggest problems we encountered but impossible to solve is that the future is inherently unpredictable. The stock market data is prone to non-economic factors such as natural disasters and political decisions. No matter how well the models are trained for, the models created to forecast stock markets will always exclude the future events that may affect the stock prices.

6. Conclusion

The primary goal of the project was an attempt to observe the feasibility of stock price prediction to gain profits for the investors. To analyze the movement pattern of the stock price plot, several machine learning and deep learning techniques are used.

We built three different machine learning methods to forecast the future stock prices: linear regression, k-nearest neighbours regression, random forest regression. Only the linear regression model resulted in a high validation score and satisfactorily extrapolated based on the training dataset. KNN and Random Forest regression models heavily relied on the data learned from the training dataset. Thus, if a value in the test dataset exceeds the range from the trained dataset, the prediction will likely fail, expecting the predicted close price to be the same as the maximum price in the trained dataset.

Contrary to our expectations, the deep learning model did not always produce better predictions than the machine learning model. For the dataset with simpler trends, training the model with excessive epochs overfits the data which decreases the accuracy and loses the ability to predict the general trend as well. The predictive performance of the LSTM with the extra hidden layers is proportional to the complexity of the data and the number of epochs.

We concluded that it is never possible to only rely upon historical data to predict accurate upcoming stock prices. Although some of the models, namely the linear regression and LSTM, were able to provide an insight into the general stock trend of the future with high validation scores, it is nearly impractical to produce accurate prices at all times because the future is inherently unpredictable.

7. References

[1] B, B. (2019, January 15). *Using the latest advancements in deep learning to predict stock price movements*. Medium. Retrieved December 10, 2021, from https://towardsdatascience.com/aifortrading-2edd6fac689d#bbae.

[2] FinanceData. (n.d.). *FinanceData/Financedatareader: Financial Data Reader*. GitHub. Retrieved December 10, 2021, from https://github.com/FinanceData/FinanceDataReader.

[3] *Sklearn.neighbors.kneighborsregressor*. scikit. (n.d.). Retrieved December 10, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html.

[4] *Sklearn.ensemble.randomforestregressor*. scikit. (n.d.). Retrieved December 10, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

[5] *Stock price prediction using stacked LSTM*. Analytics Vidhya. (2021, May 19). Retrieved December 10, 2021, from https://www.analyticsvidhya.com/blog/2021/05/stock-price-prediction-and-forecasting-using-stacked-lstm/.

8. Project Experience Summary

Jieung Park
Stock Prediction with Machine & Deep Learning Models
Built with Python & Jupyter

- Cleaned the stock market data using DataFrame functions from the pandas module.
- Built linear and k-nearest neighbour regression models from scikit-learn module.
- Produced validation scores of the test dataset to analyze the models.
- Analyzed the graphs and data produced from the models by researching online.
- Wrote a report to provide understanding of the program built by organizing the programming processes.

Jiwon Jun
Stock Prediction with Machine & Deep Learning Models
Built with Python & Jupyter

- Applied noise filtering algorithms to capture the general trend of the data using LOESS and rolling mean.
- Found the best combination parameters using scikit-learn module to optimize the random forest regression.
- Built random forest regression models using the scikit-learn module.
- Analyzed the statistics of machine learning models to identify the reason for their failure to predict.
- Wrote a report to provide understanding of the program built by organizing the programming processes.

Youngseong Kim
Stock Prediction with Machine & Deep Learning Models
Built with jupyter notebook

- Developed a customized stock data crawler using financeDataReader module to get stock/indice raw data
- Preprocessed the data by short-rolling and long-rolling
- Normalized the data by MinMaxscaler using scikit-learn
- Built and Trained a LSTM model to predict stock price movements using keras and tensorflow
- Analyzed the LSTM model to get statistical analysis (Ex. RMSE) and visualize the results using matplotlib
- Wrote a report to provide understanding of the program built by organizing the programming processes