Stock Price Prediction Using Machine Learning
Cloud Machine Learning Report

Siarhei Staravoitau[1], Apostolos Giannakidis[2], Paul Hyndman[3], Alka Bissora[4]

PGDCLOUD, National College of Ireland
Dublin, Ireland

1 x18162070@student.ncirl.ie
2 x20124066@student.ncirl.ie
3 x20156677@student.ncirl.ie
4 x20167636@student.ncirl.ie

Abstract*— This project performs research in Stock Prediction Cloud Machine Learning. It critically reviews various approaches and parameters whilst performing data preparation; reshaping and scaling of data; creating and training a model; executing stock predictions; and outputting results using cloud services. Predictions of stock market data and prices have now become even more important in today's economy, as investors seek ever-diminishing investment returns. Over 5000 days of data from popular stocks were collected to predict their stock market price trend for the next day using Cloud Machine Learning. The proposed solution includes pre-possessing of raw historical stock pricing data sets; split-train-testing of this data; experimentation upon several averaging techniques and optimisation methods to predict future stock prices. By conducting various approaches for stock price prediction using Cloud Machine Learning, this research is motivated to find a proposed solution that is more reliable, scalable, and computationally efficient.

Keywords: Cloud Machine Learning, Stock Price Prediction, Recurrent Neural Networks, Long Short-Term Memory (LSTM) Networks, Stochastic optimisation (Adamax), Predictive Accuracy, AWS Lambda, AWS ECS, Docker

I. Introduction

Cloud Machine Learning (ML) is a fast-growing branch within Cloud Computer Science delivering new opportunities to improve big data processing algorithms, used for predicting and forecasting various market actions and financial parameters. Stock price prediction has always been an innovative research topic for researchers in financial markets. However, it is also most challenging, as it has been proven that it is almost impossible to make highly accurate predictions [1]. This is because the nature of the stock market is extremely volatile, subject to world politics and forces of nature [2]. To improve and optimize making stock trading decisions, investment firms have recently started to actively make use artificial intelligence and cloud machine learning.

This project work will investigate various approaches for Stock Price prediction using Cloud Machine Learning, by selecting the most reliable methods to: perform end-to-end data processing; create a model; train the model; perform predictions; display results; and analyse them. Prediction accuracy improvement will be obtained by using Adam and Adamax optimisation, as well as predictive value accuracy measurements such as: mean absolute error (MAE); root mean square error (RMSE); mean absolute percentage error (MAPE); median absolute error (MedAE); and median absolute percentage error (MdAPE).

The advantages here of using cloud technology are elastic, on-demand, scalable compute and storage capability based upon algorithmic processing needs, and user ability to access predictions on-demand using serverless Amazon Web Services (AWS) Lambda, APIs, and AWS Elastic Container Services (ECS) with Docker. Experiments will be conducted to compare running ML simulations upon serverless compute services, as opposed to traditional ML execution upon local servers.

A. Research Motivation

There are three main motivational drivers behind this research:

1. Identify an optimal neural network and regression model for predicting stock prices (RNN and LSTM are proposed).
2. Identify changepoints, trends and seasonality components of stock prices, that improve stock price prediction accuracy.
3. Deploy the developed Machine Learning model onto AWS ECS / Docker and AWS Lambda; to investigate how the use of cloud services can improve computational efficiency, scalability, and performance of the algorithm.

B. Paper Structure

This paper consists of the following six sections:

- Related Work - This section conducts a literature review of related academic work in the RNN and LSTM field of stock prediction and summarizes findings. Additionally, it critically evaluates the key positive and negative aspects.
- Methodology - This section describes the approach taken for designing and building the ML model and the types of data processing performed upon the data set. It also describes the technical aspects of model implementation.
- Evaluation - This section presents graphical results of the stock prediction model and discusses them. It presents the performance metrics that were analysed, the data sampling methods used, with an error analysis on model outputs.

- Future Work - In this section, there is a discussion on how the system could be improved (features, improvements, and extensions) in the future.
- Conclusion - A summary of the research findings and a high-level conclusion for the research is presented.

II. Related Work

Determining trends in stock prices is challenging due to their non-linear, dynamic, noisy, and uncertain nature. The most popular methods used involve artificial neural network (ANN) deep learning methods [17] for both data regression and data classification. Budhani [15] had researched upon the superior ability of ANN to perform complex non-linear data mapping with excellent noise tolerance in time series data, as Jabin [5] had demonstrated in research using ANNs to model stock price movements.

Recurrent neural networks (RNN) are an extension of ANN that are optimised to provide a model to process large amounts of sequential time-series stock prices. RNN differentiates from traditional feed-forward ANN, in that neural connections are two-way; thus, neurons can pass data values backwards within the same layer or a subsequent one, this allows "short term" memory when dealing with long data sequences. The "long-term" memory for neural networks is imparted by model "training" on past datasets. This short and long-term memory allows RNNs to deal effectively with dynamic time-series prediction of stock valuation, as evidenced by Roondiwala [13] who researched that the best results derive from effective data collection for subsequent model training and the use of separate data to test the algorithm afterwards in a study of the Indian stock market NIFTY 50.

A key limitation of RNN is its inability to be trained to link information, as old memory will be overwritten by new memory as the training progresses, as uncovered by Bengio in 1994 [11]. This problem gave rise to the usage of LSTM networks that extended RNN architectures further by their inherent ability to remember stock data values for short or long periods of time [6], as evidenced by Jia [8] who tested the efficiency of LSTM architectures for stock price prediction, whilst Chen [7] demonstrated China stock market prediction using LSTM architecture. LSTM is notable [16] given its capability to acquire context-specific temporal dependence knowledge during training (LSTM memory units gather knowledge over long or short time periods, without explicit activation function calls within RNN components).

LSTM networks were proposed by Hochreiter & Schmidhuber [9] in 1997; like RNN, LSTM networks utilise modules with recurrent consistency. By utilising LSTM to compute share prices over long-term time series data, those processing weights that are applied to neural network individual data points can be corrected using stochastic gradient descent [10] with Adam(ax) optimisation as illustrated by Ba and Kingma. As LSTM effectively manages this stochastic descent gradient issue, it is now viewed by Bengio [11] and others as more lenient [12] than other deep learning methods such as RNN or traditional feed-forward loops when predicting future stock price returns, as evidenced by Yaday [14] when using LSTM to predict stock closing prices.

A LSTM model contains memory cells, which comprise of four components [9]: an input gate (allows or blocks new input from changing memory cell state), a self-recurrent connection neuron (loops back on itself – allows the memory cell state to be constant between timesteps), a forget gate (to lose previous state, a novel LSTM extension proposed by Gers [12]), and finally an output gate (used for any onwards propagation within the recurrent architecture). All these components help remove the issues of "vanishing" gradients [12], that recurrent neural networks must manage with long data sequences. The LSTM gates maintain a constant low error flow by facilitating weights adjustment and gradient truncation when not needed.

III. Methodology

A. Data sampling approach employed

To train the stock prediction model, historical and reliable stock market data was retrieved via the free-to-use Yahoo Finance Developer API, which provides many useful cloud-based API methods that facilitate data gathering. The data retrieved by the Yahoo Finance Developer API contains information about High, Low, Open, Close, Adjusted Close and Trading Volume values of each stock. To achieve daily consistency, the close prices were extracted and used as a measure of definitive stock price each day. For future stock price prediction, the following stocks (listed in the NASDAQ (NDAQ) market index) were selected: Google/Alphabet Inc. (GOOG); Microsoft Corporation (MSFT); Apple Inc. (AAPL).

In order to implement the stock prediction model in Python. The retrieved stock price datasets are stored into Pandas dataframes. These dataframes are then processed for training and prediction. Past and future dates were selected after visual analysis of plotted data and selection of relevant time frames during the experiments. It should be noted that stock prices do not change over weekends, hence these days are eliminated from the data set.

B. Data Preparation

Data preparation was performed in the pre-processing stage. First, it was checked if any NaN values exist in data set and if necessary, removed them. The data set was also checked for anomalies, i.e. non-typical data caused by any type of errors. Another measure is to check data for accuracy. This is achieved by using LinearRegressor, RepeatedKFold and RandomForestRegressor from sklearn.

C. Prediction model methodology *employed*

To perform stock price prediction using Machine learning and Python [19, 20], it is required to import data from the chosen data source (YFinance) and pre-process data using Python Pandas and subsequently select feature column(s) for model training. After features are selected, the training data needs to be scaled (min/max) and NumPy array formatted for training model loads.

Long Short-Term Memory (LSTM) networks work by predicting the future elements of a data sequence based on the outputs of the previous data sequence that was analysed, as well as additional new time series inputs provided. This project employed LSTM Networks based upon an autoregressive model with the use of Adamax optimizer and loss parameter methods for predictive value accuracy measurement such as: mean absolute error (MAE); root mean square error (RMSE); mean absolute percentage error (MAPE); median absolute error (MedAE); and median absolute percentage error (MdAPE).

D. Optimization algorithm for parameter updates (Adamax)

When optimising neural networks, a core algorithm required is that of "gradient descent" [10], for which this project employs an enhancement of the well-known Adam method called Adamax that computes adaptive learning rate per neural network weight. This is employed due to its compute efficiency, low memory needs, tolerance of gradient diagonal scaling, and suitability for use in large time series datasets. Adam(ax) works by estimating biased first and second (raw) gradient moment estimates; computing bias-corrected first and second (raw) gradient moment re-estimates; and finally updating all the network parameters.

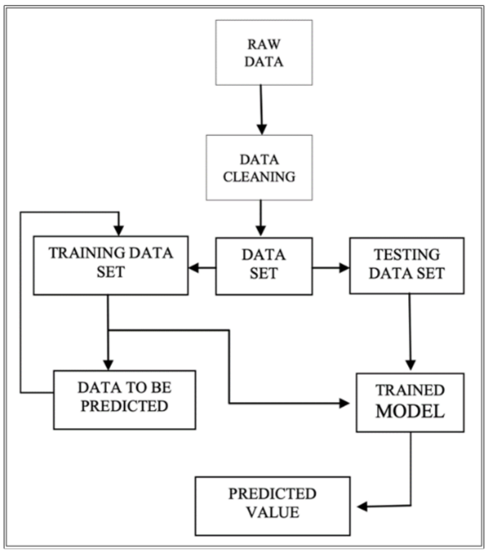


Figure 1 - Stock Trading ML prediction workflow [3]

E. Trends and Changepoint Detection

To better understand the stock market data, this research will identify possible market trends in the stock's prices. By knowing where trends changed in the past, these trend inflection points can be added into the model dataframes and significantly improve the model's performance. In this research, seasonal trends were detected, which are short-term fluctuations that occur due to seasonal factors. The associated changepoints are also detected, which are unexpected fluctuations in the time series data. The research used Facebook's Prophet, which is an open-source forecasting tool that offers built-in algorithms for changepoint detection and seasonality decomposition[1].

IV. Evaluation

A. Data training method employed

Data for model training is downloaded from Yahoo Finance by using YFinance API and saved into Pandas data frame [19, 20]. Price date ranges could be taken from the beginning of initial public offering (IPO), but this could create a lot of anomalies and distort training and prediction values due to significant price increases. To decide upon the relevant date range, researchers would retrieve data for a few years; visualize, for example, Close price movements; and then decide upon which time frame period is more relevant for performing research.

It is noted that an important part of data preparation is to get it checked for inconsistent values and anomalies (missing, out of range values, etc.). This was done using visual graphs that plotted the Close price, Date, Volume and checking data for any anomalies and inconsistencies. However, it is generally expected that YFinance datasets are complete and very accurate in this regard. During the data preparation process, there were additional checks for NaN values to avoid data issues. Also completed following test and train data check: KFOLD, Accuracy and Linear regression (Figure 4).

Before data training, it is necessary to perform selection of features (labels) and data scaling (range normalization). Data needs to be shaped and scaled for future manipulation, as this data scaling exercise improves model accuracy and training time. During the project data pre-processing stage, features selection was completed for: High, Low, Open, Close and Volume. For subsequent scaling within this research project, the *MinMaxScaler* functionality from *sklearn* was utilised.

To train the stock prediction model, historical stock price data would be split into two parts. This training / testing data split could be either percentage ratio based (normally 80% of training data and 20 % of testing data) or divided by actual date range. When preparing data for RNN training purposes, it is important to slice it into multiple data sequences with associated target values. This data slicing is achieved here by using a sliding window algorithm (Figure 2) [4, 18].

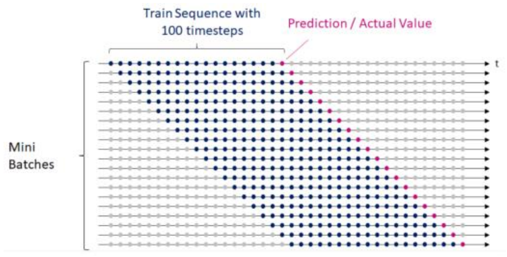


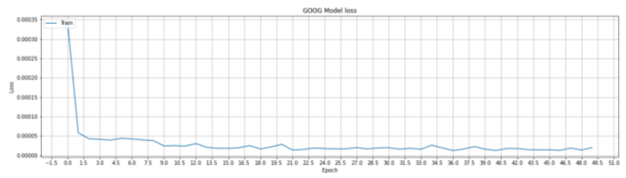
Figure 2 - Sliding window algorithm [4]

The Train sequence length used is a time frame that generates a single prediction. Python code [19, 20] is used to create a series of minibatches equal to the numerical Train sequence length, which are used to create training and test datasets, that are converted into NumPy arrays for subsequent processing.

After data preparation, the next step involves creating a training model, which includes:

- LSTM layer, taking minibatches for neural processing and returning whole sequences. (The number of neurons used here is equal to the size of the minibatch).
- LSTM layer taking sequence from previous step and returning only five values.
- Dense layer having only five neurons.
- Dense layer returning predicted value.

Model training is initiated by calling "*model.fit*", specifying the number of iterations (epochs) and batch size. For testing purposes, it was decided to use epochs equal to 50. Adamax was used as model optimizer. During this process, both the training and testing datasets are utilised. Whilst model raining is progressed, a loss curve is being generated, that reflects the differences between predicted and test values. Ideally a loss value curve should be targeting to zero (Figure 3).



```
Median Absolute Error (MAE): 26.46
Mean Absolute Percentage Error (MAPE): 1.66 %
Median Absolute Percentage Error (MDAPE): 1.52 %
```

Figure 3 - GOOG stock prediction model loss

Model validation is done by calling "*model.predict*", with the testing data used as input argument. After "un-scaling" both

[1] https://github.com/facebook/prophet

the testing and predicted data, it is possible to generate the following prediction evaluation measures: MAE, MAPE, RSME, MedAE and MDAPE (taking unscaled predicted and testing data as arguments).

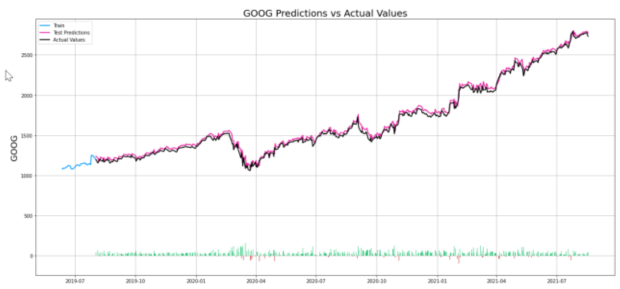
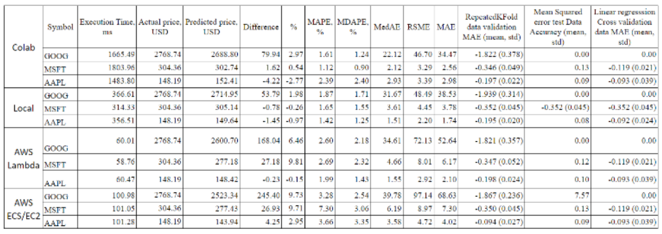


Figure 4 - GOOG stock price: predictions vs actual price

To predict the "next day" stock price, a researcher needs to define the duration timeframe to be used for model prediction (for example, same value as Train sequence length); and retrieve the most recent stock price data from the "current date (T)" down to the date corresponding to the difference of "T minus the Train sequence length". The data subsequently needs to be pre-processed, by dividing the data based upon any selected features, min/max scaling and validation processing in "*model.predict*".

Based upon this design approach [4], the project code was executed for next day stock price predictions, using securities such as GOOG, AAPL and MSFT. This execution took place both on local PC (MacBook pro 16 Intel i7, 16 GB RAM in PyCharm); cloud-based Google Colab; and finally, on "production" AWS Lambda and AWS ECS Docker. The experimentation results are reflected in Figure 5.



	Symbol	Execution Time, ms	Actual price, USD	Predicted price, USD	Difference	%	MAPE, %	MDAPE, %	MedAE	RSME	MAE	RepeatedKFold data validation MAE (mean, std)	Mean Squared error test Data Accuracy (mean, std)	Linear regresssion Cross validation data MAE (mean, std)
Colab	GOOG	1665.49	2768.74	2688.80	79.94	2.97	1.61	1.24	22.12	46.70	34.47	-1.822 (0.378)	0.00	0.00
	MSFT	1803.96	304.36	302.74	1.62	0.54	1.12	0.90	2.12	3.29	2.56	-0.346 (0.049)	0.13	-0.119 (0.021)
	AAPL	1483.80	148.19	152.41	-4.22	-2.77	2.39	2.40	2.93	3.39	2.98	-0.197 (0.022)	0.09	-0.093 (0.039)
Local	GOOG	366.61	2768.74	2714.95	53.79	1.98	1.87	1.71	31.67	48.49	38.53	-1.939 (0.314)	0.00	0.00
	MSFT	314.33	304.36	305.14	-0.78	-0.26	1.65	1.55	3.61	4.45	3.78	-0.352 (0.045)	-0.352 (0.045)	-0.352 (0.045)
	AAPL	356.51	148.19	149.64	-1.45	-0.97	1.42	1.25	1.51	2.20	1.74	-0.195 (0.020)	0.08	-0.092 (0.024)
AWS Lambda	GOOG	60.01	2768.74	2600.70	168.04	6.46	2.60	2.18	34.61	72.13	52.64	-1.821 (0.357)	0.00	0.00
	MSFT	58.76	304.36	277.18	27.18	9.81	2.69	2.32	4.66	8.01	6.17	-0.347 (0.052)	0.12	-0.119 (0.021)
	AAPL	60.47	148.19	148.42	-0.23	-0.15	1.99	1.43	1.55	2.92	2.10	-0.198 (0.024)	0.10	-0.093 (0.039)
AWS ECS/EC2	GOOG	100.98	2768.74	2523.34	245.40	9.73	3.28	2.54	39.78	97.14	68.63	-1.867 (0.236)	7.57	0.00
	MSFT	101.05	304.36	277.43	26.93	9.71	7.30	3.06	6.19	8.97	7.30	-0.350 (0.045)	0.13	-0.119 (0.021)
	AAPL	101.28	148.19	143.94	4.25	2.95	3.66	3.35	3.58	4.72	4.02	-0.094 (0.027)	0.09	-0.093 (0.039)

Figure 5 - Prices prediction metrics on 20.08.2021 [24]

B. Data results output and analysis

To evaluate the stock price prediction results, this project makes use of graphical figures with plotted actual data, training, and prediction curves. The primary prediction criteria employed is to predict stock prices as close as possible to the actual future market prices of the selected stock [2]. The following error metrics were used [3]:

- RSME - Root Mean Square Error
- MAPE - Mean Absolute Percentage Error
- MdAPE - Median Absolute Percentage Error
- MAE - Mean Absolute Error
- MedAE - Median Absolute Error

Model accuracy could be evaluated by the KFOLD – cross-validation technique, which divides the investigated data set into k number of subsets which are called folds [4]. In each iteration the method will generate a list of k accuracy values. In general, we take the average of them and use it as a consolidated cross-validation score. Results for KFOLD, Accuracy and Linear regression data validations are reflected in Figure 5.

As part of output results generation, the RNN output is compared against the target value, the resultant difference error is minimised where possible by the neural network algorithm back-propagation functionality that adjusts network weights and bias. To analyse system efficiency the metrics above are used for predictive value accuracy measurement. It should be noted that Root Mean Square Error (RMSE) is generally used as an all-purpose error metric for time-series prediction; this is mainly driven by its ability to amplify and negate any large error thresholds that are encountered, as compared to MAE.

Figure 5 above illustrates the similar behaviour of calculated metrics across stock securities. It appears that Google metrics are relatively high for RSME and MAE. RSME shows how concentrated the data is around the line of the best fit: i.e., it is 72.13 when running in AWS Lambda, 97.14 when running on AWS ECS, 46.70 when running in Google Colab; and 48.49 when executing the same code on local PC. MAE research figures for Google would indicate that it could be expected to produce an error of 52.64, 34.47 and 38.53 when running forecasting on average in AWS Lambda, Google Colab and then on local PC respectively. At the same time, for Apple and Microsoft shares, RSME and MAE metrics are relatively low.

For Google, this could be explained by using stock prices when they were relatively low and then rapidly spiked up. In the same timeframe, when looking into the results in Fig. 4, the test predictions are relatively close to actual price; this would indicate that next day predictions with the sliding window approach have produced good results within this research project. Predicted price differences for all stocks during this research project were predicted relatively close to the real price as well. Google price prediction showed the biggest difference, that is confirmed by MAE and RSME metrics.

Execution timing is more efficient when running code in AWS Lambda. In average it is 60s in AWS Lambda vs 100s in AWS ECS / Docker on EC2, 340s in local deployment and 1640s in Google Colab. Using AWS Lambda allows researchers to complete more intensive calculations, especially with increasing number of neurons and epochs.

C. Trend Changepoints and Seasonality

Using Facebook's Prophet forecasting tool, we were able to detect changepoints, trends and seasonality for the stocks of our research. We analysed the changepoints of all stocks and all had a significant increase starting in July of 2020 (Figure 6).

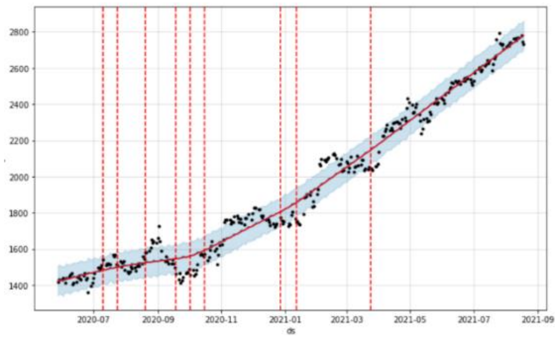


Figure 6 - Google stock price changepoint detection

The same trend is also true for July of 2021. This increasing trend could be explained by the summer season and the opening of the markets in the USA and around the world. Thus, we can predict that while the Covid-19 pandemic impacts the financial markets, the stock market will demonstrate an increase in the stock prices during the summer seasons.

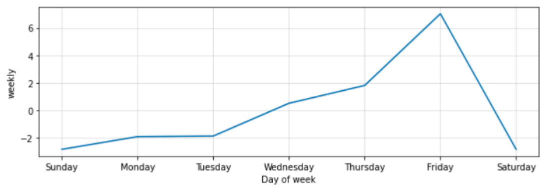


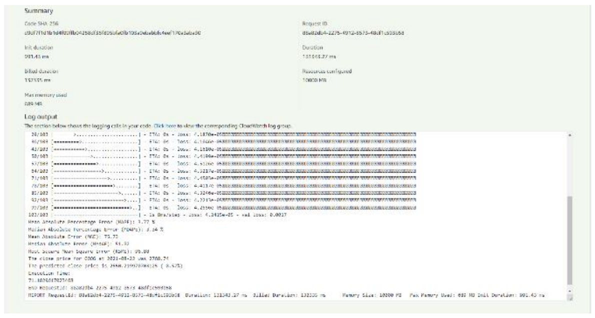
Figure 7 - Google's stock weekly seasonality decomposition

Figure 7 shows the seasonality decomposition of Google's stock for the past year. It is evident from the figure that investors have better performance buying Google's stock on Sunday and selling the stock on Friday, because its price is likely to drop after the weekend.

V. Cloud Deployment

The proposed ML model has been deployed upon three different Cloud environments during this research project:

1. The Python source code has been deployed on Google's Colab, which is a Python development environment that runs in the browser using Google Cloud. The Colab Notebook can be accessed by the hyperlink here.
2. The Python code has been packaged as a Docker image which was pushed onto AWS Elastic Container Registry (Amazon ECR) and then onto DockerHub.
    a. A Docker container has been deployed on AWS Lambda with 10GBs of memory and 6 vCPUs [21]. Figure 8 shows the console logs of a test invocation of the Lambda function for Google's stock prediction.



Summary

Code SHA-256
[illegible]

Request ID
d6a82db4-2275-4912-8573-48df1c593b58

Init duration
991.43 ms

Duration
131543.27 ms

Billed duration
132335 ms

Resources configured
10000 MB

Max memory used
689 MB

Log output

The section below shows the logging calls in your code. Click here to view the corresponding CloudWatch log group.

```
 29/103 [>.......................] - ETA: 0s - loss: 4.1870e-05
 36/103 [=======>.................] - ETA: 0s - loss: 4.1666e-05
 43/103 [=========>...............] - ETA: 0s - loss: 4.1610e-05
 50/103 [===========>.............] - ETA: 0s - loss: 4.4190e-05
 57/103 [=============>...........] - ETA: 0s - loss: 4.5126e-05
 64/103 [===============>.........] - ETA: 0s - loss: 4.3217e-05
 71/103 [=================>.......] - ETA: 0s - loss: 4.4503e-05
 78/103 [===================>.....] - ETA: 0s - loss: 4.4117e-05
 85/103 [=====================>...] - ETA: 0s - loss: 4.3246e-05
 92/103 [=======================>.] - ETA: 0s - loss: 4.2213e-05
 99/103 [========================>.] - ETA: 0s - loss: 4.2556e-05
103/103 [==========================] - 1s 8ms/step - loss: 4.2415e-05 - val_loss: 0.0017
Mean Absolute Percentage Error (MAPE): 3.77 %
Median Absolute Percentage Error (MDAPE): 3.34 %
Mean Absolute Error (MAE): 75.72
Median Absolute Error (MedAE): 51.92
Root Square Mean Square Error (RSME): 96.88
The close price for GOOG at 2021-08-22 was 2768.74
The predicted close price is 2558.2199[illegible] ( 8.67%)
Execution Time:
71.10258[illegible]
END RequestId: d6a82db4-2275-4912-8573-48df1c593b58
REPORT RequestId: d6a82db4-2275-4912-8573-48df1c593b58  Duration: 131543.27 ms  Billed Duration: 132335 ms  Memory Size: 10000 MB  Max Memory Used: 689 MB  Init Duration: 991.43 ms
```

Figure 8 - AWS Lambda stock training and prediction output

    b. Another Docker container has been deployed on an AWS Elastic Container Service (ECS) cluster. An ECS service has been created to spin-off the Docker container in an EC2 *t2.micro* instance with 400 MBs of memory.

To access the produced Python ML model, use the following hyperlinks:

- GitHub: Source code of the ML model
- DockerHub: Docker image of the ML model

To access the AWS ECS endpoint run the following command in a terminal window:

```
curl -X POST
"http://34.248.130.208:9099/trainAndPredi
ct?epochs=50&sequence_length=100&
&stock_symbol=GOOG" -H "Content-Type:
application/json"
```

The AWS ECS endpoint is currently live and can be used to interact with the ML model. HTTP requests can be sent without any HTTP parameters because it uses default ones. The available HTTP parameters are:

- *stock_symbol:* Defines the stock that is used to get predictions for. (Values: GOOG, MSFT, AAPL).
- *epochs:* Defines an integer that represents the number of epochs the model should use.
- *sequence_length:* Defines an integer that represents the sequence days that the model should use.

By adjusting the *epochs* and *sequence_length* parameters, we can experiment with the accuracy and performance of the ML model. The response of the ECS endpoint is a JSON array with the stock prediction and the usage metrics of the model.

VI. Future Work

Predicting stock prices with Machine Learning techniques seems very promising and it would be valuable to continue our research on this topic. The end goal is to further improve the accuracy of the model's predictions by exploring different data pre-processing techniques, regression, and forecasting models.

Additional research could focus on the use of LSTM layers modification, adding additional datasets, layers, dropouts and greatly varying the number of epochs utilised in each experiment run. This is the prime benefit of the cloud docker implementation in that elastic, scalable, on-demand compute and storage resources are available for experimentation on a "pay as you go" basis.

A valuable extension that could be added is that of sentiment analysis sourced from social media such as Reddit, Facebook, and Twitter APIs. This would allow the neural network to be influenced by what the marketplace feels about a stock and its valuation. A prime example here is the recent GameStop controversy where an unrealistic valuation was achieved by social media driven retail purchases interfering with market shorts that existed, thus artificially driving the stock price to market highs. One novel approach that is worth exploring is the use of genetic-based algorithms that have the potential to further improve the performance measures obtained in this research. Another approach we could explore is the usage of the autoregressive integrated moving average (ARIMA) model, which is considered as one of the most popular approaches to forecasting time series data [22].

Finally, the proposed methodology can be further improved by eliminating a data leakage issue in the data preparation process. This can be done by applying data preparation on the training data set before the data is split [23].

VII. Conclusion

The use of machine learning models has been widely proven to be capable of predicting financial time series data and thus, provides researchers with a broader understanding of stock market price prediction; however, this processing is computationally demanding in terms of local compute power needed; and hence it is now valuable to explore cloud deployment alternatives.

In this paper, we used a series of Machine Learning approaches for predicting stock prices. We selected a multivariate linear regression model with a Long Short-Term Memory (LSTM) Network as the most appropriate and reliable approach for our use case. Using historical financial data from the Yahoo Finance Developer API, we trained our algorithm and created an appropriate model to perform the stock price prediction, thus proving the efficiency of RNN / LSTM models for stock prediction.

For the final part of the project, we deployed our Python ML model as a Docker container upon AWS Lambda and AWS ECS. According to our evaluation and analysis, the proposed stock performance prediction model running on Docker and Lambda has significantly better performance. This deployment allows more exhaustive prediction calculations, and hence more precise model training and prediction results.

VIII. Contributions

Report Section	**Contributors**
Research / Design	Siarhei Staravoitau, Paul Hyndman, Apostolos Giannakidis, Alka Bissora
Code Development	
Experiments	
Report Creation	

REFERENCES

[1] J. Sen, ‘Stock Price Prediction Using Machine Learning and Deep Learning Frameworks’, p. 10

[2] S. Shakhla, B. Shah, N. Shah, V. Unadkat, and P. Kanani, ‘Stock Price Trend Prediction Using Multiple Linear Regression’, May 2020

[3] B. B. P. Maurya, A. Ray, A. Upadhyay, B. Gour, and A. U. Khan, ‘Recursive Stock Price Prediction With Machine Learning And Web Scrapping For Specified Time Period’, in 2019 Sixteenth International Conference on Wireless and Optical Communication Networks (WOCN), Bhopal, India, Dec. 2019, pp. 1–3. doi: 10.1109/WOCN45266.2019.8995080

[4] ‘Stock Market Prediction using Multivariate Time Series Models in Python’. https://www.relataly.com/stock-market-prediction-using-multivariate-time-series-in-python/1815/ (accessed Aug. 19, 2021).

[5] S. Jabin, ‘Stock Market Prediction using Feed-forward Artificial Neural Network’, International Journal of Computer Applications, vol. 99, pp. 4–8, Aug. 2014, doi: 10.5120/17399-7959.

[6] T. Gao, Y. Chai, and Y. Liu, ‘Applying long short-term memory neural networks for predicting stock closing price’, in 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Nov. 2017, pp. 575–578. doi: 10.1109/ICSESS.2017.8342981.

[7] K. Chen, Y. Zhou, and F. Dai, ‘A LSTM-based method for stock returns prediction: A case study of China stock market’, in 2015 IEEE International Conference on Big Data (Big Data), Oct. 2015, pp. 2823–2824. doi: 10.1109/BigData.2015.7364089

[8] H. Jia, ‘Investigation Into The Effectiveness Of Long Short Term Memory Networks For Stock Price Prediction’, arXiv:1603.07893 [cs], Aug. 2016, Accessed: Aug. 11, 2021. [Online]. Available: http://arxiv.org/abs/1603.07893

[9] S. Hochreiter and J. Schmidhuber, ‘Long Short-Term Memory’, Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[10] D. P. Kingma and J. Ba, ‘Adam: A Method for Stochastic Optimization’, arXiv:1412.6980 [cs], Jan. 2017, Accessed: Aug. 11, 2021. [Online]. Available: http://arxiv.org/abs/1412.6980

[11] Y. Bengio, P. Simard, and P. Frasconi, ‘Learning long-term dependencies with gradient descent is difficult’, IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157–166, Mar. 1994, doi: 10.1109/72.279181.

[12] F. A. Gers, J. Schmidhuber, and F. Cummins, ‘Learning to Forget: Continual Prediction with LSTM’, Neural Computation, vol. 12, no. 10, pp. 2451–2471, Oct. 2000, doi: 10.1162/089976600300015015.

[13] M. Roondiwala, H. Patel, and S. Varma, ‘Predicting Stock Prices Using LSTM’, International Journal of Science and Research (IJSR), vol. 6, Apr. 2017, doi: 10.21275/ART20172755.

[14] O. Yadav, Y. Shah, S. Talim, and B. Britto, ‘Prediction of Stock Closing Price by applying Long Stock Term Memory Neural Network’, vol. 6, no. 0972, p. 5, 2019.

[15] N. Budhani, C. K. Jha, and S. K. Budhani, ‘Prediction of stock market using artificial neural network’, in 2014 International Conference of Soft Computing Techniques for Engineering and Technology (ICSCTET), Aug. 2014, pp. 1–8. doi: 10.1109/ICSCTET.2015.7371196.

[16] F. Qian and X. Chen, ‘Stock Prediction Based on LSTM under Different Stability’, in 2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Apr. 2019, pp. 483–486. doi: 10.1109/ICCCBDA.2019.8725709.

[17] C. S. Vui, G. K. Soon, C. K. On, R. Alfred, and P. Anthony, ‘A review of stock market prediction with Artificial neural network (ANN)’, in 2013 IEEE International Conference on Control System, Computing and Engineering, Nov. 2013, pp. 477–482. doi: 10.1109/ICCSCE.2013.6720012

[18] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, ‘Stock price prediction using LSTM, RNN and CNN-sliding window model’, in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Sep. 2017, pp. 1643–1647. doi: 10.1109/ICACCI.2017.8126078.

[19] ‘Stock Price Prediction Using Machine Learning | Deep Learning’, Analytics Vidhya, Oct. 25, 2018. https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/ (accessed Aug. 11, 2021).

[20] J. Brownlee, ‘Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras’, Machine Learning Mastery, Jul. 20, 2016. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ (accessed Aug. 11, 2021)

[21] Using container images to run TensorFlow models in AWS Lambda’, *Amazon Web Services*, Feb. 24, 2021. https://aws.amazon.com/blogs/machine-learning/using-container-images-to-run-tensorflow-models-in-aws-lambda/ (accessed Aug. 22, 2021).

[22] 4. “A Hybrid ARIMA and Support Vector Machines Model in Stock Price Forecasting”, Omega, the International Journal of Management Science. vol. 33, no. 3, 2005.

[23] Alice Zheng and Amanda Casari. 2018. Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists (1st. ed.). O'Reilly.

[24] Test runs metrics spreadsheet CLOUDML.xlsx