A FINANCE ROBOT FOR REAL-TIME STOCK PRICE PREDICTION

Nam Nguyen
Department of Electrical Engineering
University of South Florida
Tampa, USA
{namnguyen2}@usf.edu

1 INTRODUCTION

The goal of the project is to offer a finance robot that utilize *real-time* stock price to draw a prediction. As the robotics point of view, the agent (robot software) interacts with the environment (real-time stock market) during the opening hours. Then the information feedback from the environment is processed by an embedded intelligence to drawn the inference. There are three main components of the robot, which are:

1. Embedded AI using machine learning and deep learning to analyse historical data for inference and prediction.
2. Real-time stock price report that streams and analyzed the stock value directly from stock market.
3. Human-machine interface

Moreover, we will investigate several time-series analysis techniques, that are able to offer choices for the embedded AI. Thus, it is more convenience for user to make decisions. Although in fact, there are external attributes for the fluctuation of stock prices, within the scope of the project, we only consider historical data for prediction.

The project is only use for illustration of Robotics-AI final project for the stock prediction. Hence we do not recommend any use of this project for real trading tool, since there are many external factors (domain knowledge, trading technique) that are required to make this tool for real trading use.

2 DESIGN OF AGENT

2.1 AGENT AND ENVIRONMENT INTERACTION

Since the robot streams and analyzes real-time stock price, the online status will follow the trading hours of U.S stock market (New York Stock Exchange (NYSE) and the Nasdaq Stock Market (NASDAQ)), which is from 9:30 AM to 4:00 PM weekdays. The real-time stock price is then streamed directly from API of yahoo finance with a given Ticker (Name of monitored stock). We offer the updating range for the agent, which allows user update/monitor/analyse stock price every 1 min, 2 min, 5 min and 15 min. However, due to the nature of stock trading, the algorithm will only predict the stock price in the next minute. Moreover, the agent also obtains the historical stock price in 3 months and 12 months, that allows traders to observe the time-series characteristic of the stock, such as seasonal trend or unusual trading activities.

2.2 HUMAN-MACHINE INTERFACE

The Human-machine interface is delivered by an web app created in Python 3.7 and Dash. We attempt to design a user friendly UI, which minimises to essential information (for detail see A). Regarding the first tab of the web app, we first offer two drop-down button for choice of Ticker and

updating duration (1). In the main part of the Dashboard (2), the real-time stock price is streamed and analysing by the embedded AI to predict the stock-price for the next time step (set to be 1 minute ahead). In (3) is the overall trend of the monitoring stock reported in the past 3 and 12 months. Finally, the analyses from the embedded intelligence is report in (4), which will be discussed more in Sect. 3. All visualization in the UI is interacting graphs, which allows user focus the reported price by dragging the mouse for selected region. In the second tab, we report the bid-ask price for currency and the most trending article, since the stock price is most of the time sensitive to the news. Unfortunately, we cannot finish the second AI for sentiment analysis articles related to the monitored price. However, the backbone offline implementation is completed and will be also in Sect. 3.

3 Design of Embedded Intelligence

The artifact intelligence embedded in the robot is able to perform inferences based on both historical data and real-time stock price. We designed two separated A.I(s) for each training data. The final prediction is weighted ensemble of two sub-prediction. Moreover, we offer several choices for the behind machine learning, ranging from the most basic linear regression, moving average and more complicated algorithms which are tree-based gradient boosting. Since the sub-intelligence shares the same model, we only discuss the mathematical background in the next section. Furthermore, we apply simple linear regression only to illustrate the effectiveness of more advanced algorithm, so we will take its methodology for granted in this report (see (3) for details).

3.1 Query real-time stock price for training embedded AI

For the historical data, we queried the stock price over 3 weeks before the last week for train set (recommended in practice), leading to $3 \times 5 \times 24 \times 6.5 = 2340$ data points (still consider small sample in overall big data analysis). The test set is acquired from the stock price of the last week. Thus, we preserve the most popular ratio of train/test, which is 80:20. Note that the train the agent using tensor; for example, the stock price of each day is in the shape of $(1, 156)$ then the train set has the shape of $(15, 156)$. It is suggested to queries time-series with tensor as above (2), since when we train/test the model in minibatch, we implicitly taking the time information to account. Moreover, we set up an auto update in Saturday, meaning if we turn on the web app, the offline AI will be automatically updated using the new queries train/test set. The approximate time for training/updating is 4-5 minutes (when training from scratch).

Regarding the online intelligence, for the first 2.5 hours, we do not have enough data for training the online AI, thus we predict the stock price using offline AI. After this duration, the AI compute the time-series statistic for training the online agent. Note that we append the real-time stock price to current train set, so for training it from scratch may take up to 5 minutes, which is not appropriate for short-term prediction. Thus, we only soft-train the model using few number of epoch (10) with offline initial weights, leading to lower than 45 seconds of train-inference-prediction.

3.2 Abnormality detection and Moving Average Convergence/Divergence

The Relative Strength Index (RSI) is indicator for overbought/oversold signal from the monitored stock. The range of RSI is in $[0, 100]$, over 70 indicates overbought while 30 indicate oversold. We left the decision for user to decide buy or sold the stock, since more often than not, we need to adjust the margins of the RSI. Particularly, if the RSI hit the threshold of 70 frequently, users need to visit the second tab for related articles. The mathematical formula for calculation of RSI is given as

$$\text{RSI} = 100 - \frac{100}{1 + \frac{\text{AvgU}}{\text{AvgD}}},$$

where the ratio AvgU : AvgD is the relative strength.

The moving average convergence/divergence (MACD) is the indicator for strength, direction and momentum of the stock. Overall, it measures the trend of the stock. In term of signal processing, these indexes is a filtered values for the first derivative of the input.

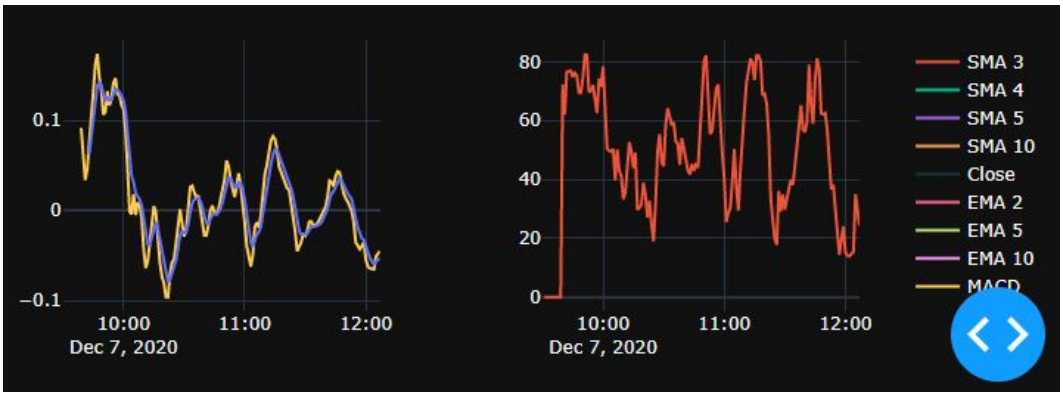

Figure 1: MACD and RSI indicator with real-time updates.

3.3 Predictive Intelligence

Figure 2: Prediction of the agent on real-time stock price. (Top) Prediction based on EMA of 5 minutes, (Mid) based on simple moving average (SMA), (Bottom) Ensemble of offline and online prediction.

For training both intelligence, we extract vital statistics from the given time-series, which is simple moving average (SMA), exponential moving average (EMA), MACD and RSI. Moreover, the original signal (open/high/volume) is also use for the train feature. The target variable is the close price at the given time step. The first model (Figure 2(a)) is trained by stand-alone EMA (computed within every 2 minutes), while the model in Figure 2(b) using SMA only. We can see that the marginal errors of model 2 is larger than model 1 in overall (up to the monitored time). Additionally, we also

observed the lagging effect of prediction (especially from model 2). As mentioned in Sect. 4, the online intelligence only can predict after 30 minutes, since we do not have enough data for the first 30 minutes. So instead we use the offline model for this duration (Figure 2(c)). The mathematical background for gradient boosting machine is given in Sect **??**. The final prediction is weighted ensemble $w_{\text{offline}}, w_{\text{online}} = 0.6, 0.4$.

REFERENCES

[1] Kaggle, *"Huge dataset for stock price prediction."*

[2] Dash, *"https://plotly.com/dash/"*

[3] Trevor Hastie, *"Statistical Learning"*

[4] T. Chen, et al.,*"Xgboost: A scalable tree boosting system"*

[5] Nam Nguyen, *Master Thesis: "Gradient Boosting for Survival Analysis with Applications in Oncology"*

A HUMAN-MACHINE INTERFACE

B BOOSTING ALGORITHM

Gradient boosting machine is a predictive machine learning method that yields prediction by residuals. In Algorithm 1, the target close price is modeled by $\hat{f}(\boldsymbol{z})$, where $\boldsymbol{z}$ is training feature and Φ is simple decision tree.

Algorithm 1 Gradient Tree Boosting Algorithm

1. Initialize
$$f_0(\mathbf{y}) = \arg\min_{\theta} \sum_{i=1}^{N} \Phi(\mathbf{y}, \boldsymbol{\theta}).$$
2. For $m = 1$ to M:
   (a) For $i = 1, ..., N$ compute the pseudo residuals
   $$r_{im} = -\left[\frac{\delta \Phi(y_i, f(z_i))}{\delta f(z_i)}\right]_{f=f_{m-1}}.$$
   (b) Fit a regression tree to predict r_{im}, obtaining terminal regions $R_{jm}, j = 1, ..., J_m$
   (c) For $j = 1, ..., J_m$ compute
   $$\theta_{jm} = \arg\min_{\boldsymbol{\theta}} \sum_{z_i \in R_{jm}} \Phi(\mathbf{y}, f_{m-1}(\mathbf{z}) + \boldsymbol{\theta})$$
   (d) Update
   $$f_m(\mathbf{z}) = f_{m-1}(\mathbf{z}) + \sum_{j=1}^{J_m} \theta_{jm} I(\mathbf{z} \in R_{jm}).$$
3. Output $\hat{f}(\mathbf{z}) = f_M(\mathbf{z})$

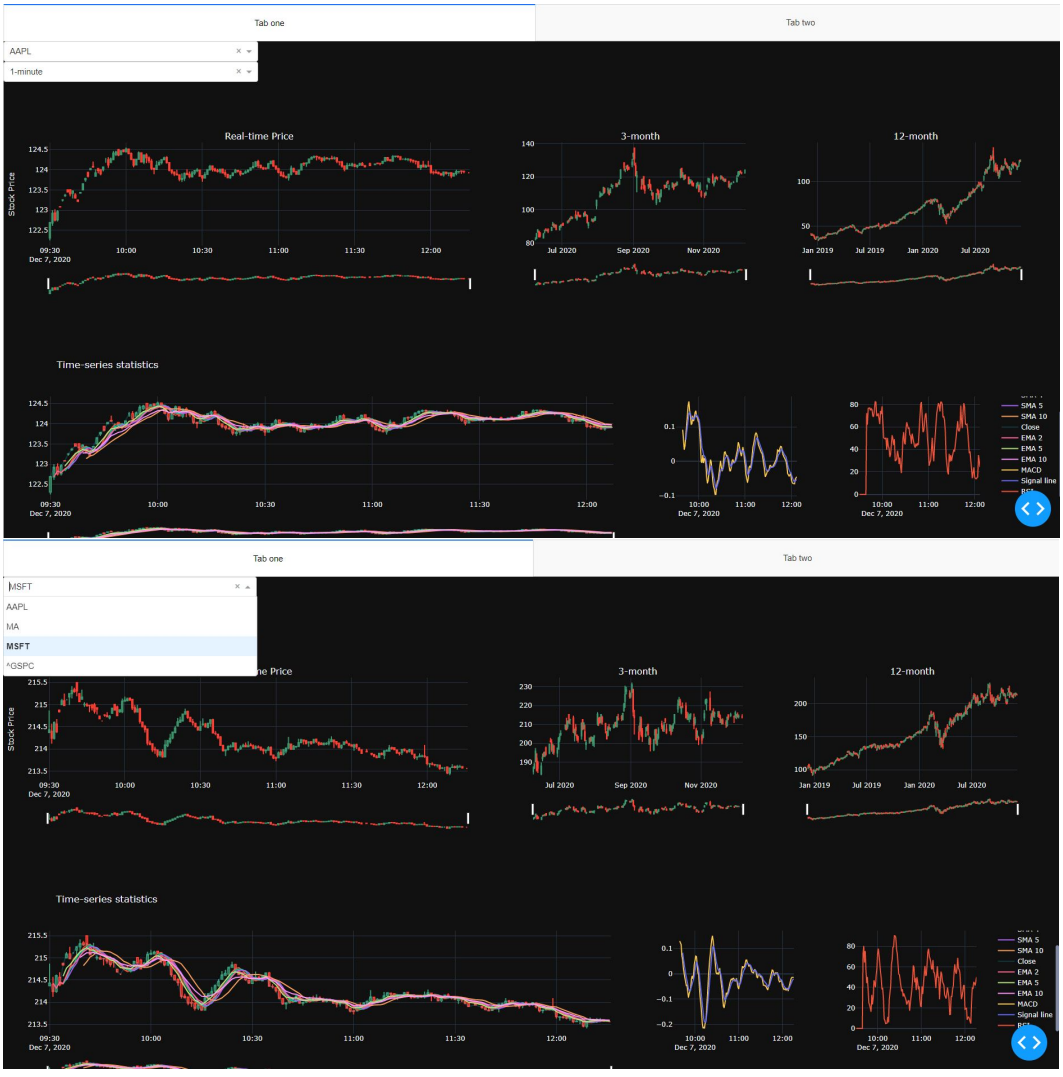

Figure 3: Overall view of HUI, the first drop-box offers choices of monitored stock price (Ticker), while the second allows alternation of update duration.

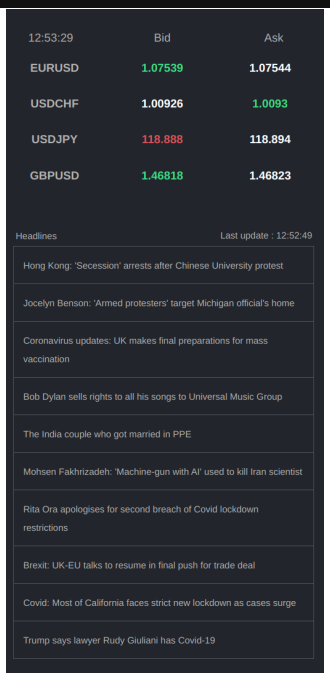

Figure 4: (Top): Real-time stock price streamed directly from yahoo finance with update duration of 1 minutes. (Mid) Historical stock price observed in past 3 months and 12 months. (Bottom) Bid price and article from yahoo news.