Canary: Anomaly Detection and Classification with General Model Supports

Jialun Lyu
Department of Computer Science
University of Toronto
lyujialu@cs.toronto.edu

Rui Yang
Department of Mechanical and Industrial Engineering
University of Toronto
ruimavis.yang@mail.utoronto.ca

Tianqi Yue
Department of Mechanical and Industrial Engineering
University of Toronto
tianqi.yue@mail.utoronto.ca

Abstract

The stochastic and complex nature of stock prices presents a challenge for deep learning techniques. While sophisticated deep learning models can capture underlying patterns, they may fail to generalize by learning rare and irregular patterns. To address this, we propose a novel framework, called Canary, which integrates anomaly detection and time series prediction. This framework enables sustainable online learning by allowing the model to adapt to changes in the input stream while stabilizing the continual learning process. Our simulations demonstrate promising aspects of this framework in highly volatile environments, such as quantitative trading. **Github Link:** https://github.com/carlonlv/ProjectBuffalo.git

1 Introduction

Identifying irregular patterns in time series data is a well-studied problem, and is of great importance in the application of stock trading. Stock prices are influenced by a multitude of factors including macroeconomic conditions, company-related events, and market sentiment, among others. These factors can cause significant changes in the behavior of stock prices over time, making it difficult to predict future prices based solely on past patterns. By identifying irregular jumps and damps in the past, traders can design better trading strategies to hedge against the risk of unexpected turns. In this paper, we introduce a novel framework which jointly trains an arbitrary data model and an anomaly detection model, and outputs the identified change points, outliers and corrected model parameters after removing the effects from influential observations. In this context, we refer outliers [1] as both influential points that affects generalization of parametric models and points of interest results from unexpected events.

1.1 Background

Traditionally, Tsay [1988] categorized outliers into AO (Additive Outliers), IO (Innovative Outliers), LC (Level Change) and VC (Variance Change), and LC can be further classified as LS (Level Shift) and TC (transient level shift). The categorization of outlier types correspond to different scenarios in stock markets; AOs represent random jumps and damps in stock prices that are unforeseeable but only disrupt markets within a short period. IOs represent sudden changes with decaying effect on the market prices that eventually goes to zero. LCs and TCs represent random change of behaviors in level and volatility, respectively. More complicated outlier patterns can result from a linear combinations of those four types.

[1] The terms "outliers" and "anomalies" are used interchangably in the context of this paper.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

The original design of these outlier models are paired with general seasonal and nonseasonal ARMA (Autoregressive Moving Average) models. As stock markets are nonstationary, have complicated behaviors and high volatility, more recent works such as Khare et al. [2017] and Nabipour et al. [2020] proposes deep learning models to predict stock prices. One of our contributions in this paper is to identify and categorize outliers while allowing complicated models to be used to predict stock prices.

1.2 Motivation

Algorithmic trading is challenging problems in stock markets due to many reasons; a) Stock prices have complicated patterns due to long and short term effects, which means simple model cannot describe underlying patterns well. b) Stock prices are highly dynamic requiring algorithms to be flexible and adaptable to shifting conditions, and highly efficient to process large amount of information accurately. c) Stock markets are highly competitive and subject to market regulations, which means algorithms need to handle sudden impacts and hedge against risk in the long run.
In this paper, we present Canary[2]: a framework that composes two modules, a data module whose purpose is to describe and generalize the pattern of time series data, and an outlier module whose purpose is to classify the residuals as different types of outliers. Predicted distribution of future observations is achieved by using convolution operation between the distribution of predicted value and the discrete probabilities of the outlier types. As more data are observed, the predicted distribution of outlier types are subject to change based on the decay of outlier effects. In contrast to the iterative procedure proposed by Chen and Liu [1993], our algorithm for identifying outlier types are scalable to number of observations and provides probabilistic intuition of possible scenarios to be encountered. We believe such breakdown would help traders to develop better trading algorithms that hedge against different scenarios and profit in the long run.
Our contribution includes:

- We propose a flexible framework where users can choose to design and substitute models in data and outlier modules according to their own likings. Since each submodel trains and predicts semi-indepently. Our framework supports ensembles of machine learning models.
- We propose an online update mechanisms that allows the models in both submodules to adjust to new observations. Such mechanism proves to useful to adapt the changes in highly dynamic stock markets.

2 Methodology

2.1 Overview

Pipeline of Canary Figure 1 demonstrates how Canary manages input data stream in the context of online learning; Canary keeps track of an **Offline** data model and an **Online** data model. In the context of monitoring stock market, offline model a pretrained model on multiple stocks over a long historic data, whose purpose is not to predict the exact stock price for any given stock but to memorize as many patterns as possible. The online model is a realized model trained using offline model and most recent data from an arbitrary given stock symbol. The online model is updated frequently to accommodate the constant behavioral changes in the stock. One may also replace this design using autoencoder and decoder, with encoder learning the behaviors of historical data and decoder trying to predict stock prices in higher fidelity.
After online model makes prediction, we obtain the residual time series by simply subtracting the actual stock prices by the predicted values. The residual stream is then passed into outlier detection model, whose job is to learn the unexplained jumps and damps in the residuals. One can design outlier model to classify the outliers in the residuals into one of the categories in the above paragraph. Since TC takes parameter δ, one may set different realizations of δ to make as many labels of outliers as desired.
Once outliers of the residual time series are recognized, we can compute the outlier effect deterministicly using the formulas for each outlier type as described in above paragraph. Since outlier model

[2]The canary bird was used in early mining operations as a warning system for dangerous gases. Miners would bring a caged canary into the mines, since the bird's sensitive respiratory system would be affected by toxic gases before the gases reached harmful levels for humans. If the canary stopped singing or died, it was a sign that the miners needed to evacuate the mine immediately due to dangerous gas levels. We name our framework Canary to indicate its functionality to early detect signs of catastrophic market behaviors.

outputs the probability of each outlier category(and non-outliers), the outlier effect is an expectation over the distribution of outlier types. The adjusted time series can be computed by subtracting the original data stream by the outlier effect. The adjusted time series should be rid of irregular behaviours but still captures the trends and drifts, which would be fed into the online data model for continual learning.

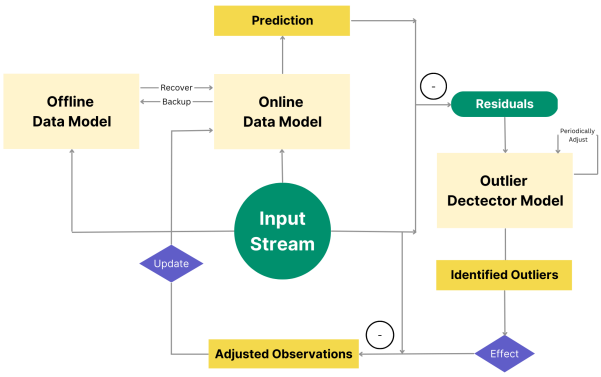

Figure 1: The design of Canary, each directed edge represents interaction between Data Module and Outlier Module.

Choice of data model It should be noted that the categorization of outliers is dependent on the choice of data model used. Simple time series models such as AR(1) may not be able to capture changes in the behavior of stock markets, resulting in correlated residuals even after removing outlier effects. On the other hand, deep neural networks can act as more complex data models and are capable of capturing intricate seasonal behaviors, resulting in fewer and less varied outliers in the residual time series. However, overfitted deep neural networks may memorize rare and irregular behaviors, leading to poor generalization performance. Therefore, the data model should be complex enough to capture all trends and intricate seasonality, but not focus too heavily on rare irregularities. In our design, we chose the same 2-layer RNN architecture with hidden size of 64 but different realization for different stock symbols.

Outlier Identification and Prediction Models As oppose to white noise processes, outliers are typically observed in clusters and are results of economic policies. For example, a typical measure to counter high inflation is to increases interest rate, which is believed to have temporal negative impact on stock markets. This means that one may use autoencoder model to encode the macroeconomic indicators and categorize the residuals at each time stamp (into non-outliers, AOs, TCs, etc). Since outliers such as TC or LC can only be identified after observing the succeeding residuals, one may adjust the categories of the identified outliers once in a while to get more accurately adjusted time series. Intuitively, the adjust frequency of outlier categorization must be higher than the update frequency of the online model to avoid updating the online model with corrupted inputs.

3 Simulation

We simulate our designed system using daily data from 22 stock symbols along with macroecomonic indicators [3] and transformation of stocks [4]. Our simulations include all historic data for each stock symbol from their initial IPO to March 1st, 2023. We report the efficacy of Canary through the assembly of each submodules [5]. As categorizing outliers is supervised learning, we generate the labels for each residual data point through iterative procedure as proposed by Chen and Liu [1993]. The outlier model aims to learn the labels for each residual generated by this algorithm.

[3]List of macroeconomic indicators: Federal funds rate, Payroll, CPI (Consumer Price Index), Unemployment Rate, GDP, GDP per capita.

[4]For OCHLV (Open, High, Low, Close, Volume) of each stock, we compute the following transformations: SMA(Simple Moving Average) with time period of 10, 50, 200, ROC(Rate of Change), HT(Hilbert Transformation), MOM(Momentum)

[5]Due to the limitation of time, we have not tested the overall structure extensively, instead, we tuned individual models through grid search. We hope to include the results of the entire system in the future

3.1 Main Result

Predicting individual stocks through offline learning Using autoregressive models such as RNN and LSTM, we train the data model using 80% of the datapoints and test the model on the succeeding test set. Table 1 reports the MSE loss on training set and testing set. We noticed that simple structured RNN and LSTM are not able to capture complicated bahaviors in both training and testing phases. Deeper layer RNN and LSTM can capture the training set well but fail to generalize. The left subplot of Fig 2 demonstrates the behavior of 4 layered LSTM model. During prediction, LSTM model is able to predict the behaviors of rises and drops in the testing set but fails to predict the exact prices.

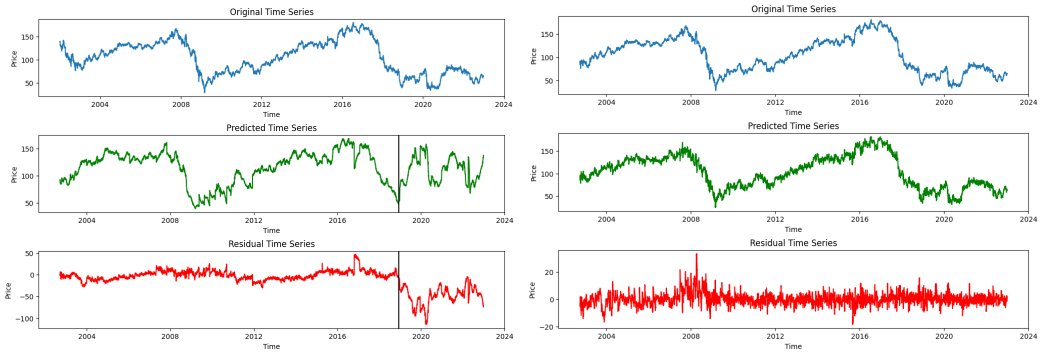

Figure 2: Comparing the performance of online LSTM model with periodic update of once per week and offline LSTM model tested on 20% of the data (train and test set separated by vertical line).

Predicting individual stocks through online learning Using autoregressive models such as RNN and LSTM, we train the data model using all of the existing datapoints up to date. Comparatively, online version of LSTM is much better at predicting both the pattern and the exact value of stock prices, as shown in right subplot of Figure 2. We noticed that occasionally online model predicts unexpected values, such as huge jumps and damps, which means that online model may be unstable and sensitive to the irregular inputs. The memory units in the online model may remember such behavior and out-of-box predictions may manifest in the future, adding risk to deploy trading strategy in an automated fashion.

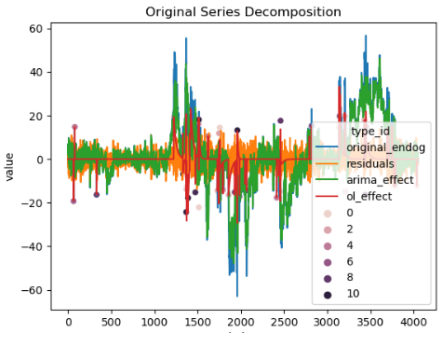

Figure 3: The decomposition of residual time series. Since the residual seem be to correlated, we apply AR(1) model to further decompose the time series and classify the outliers into 11 predefined categories(AO, IO, TC($\delta = 0.1$ to $\delta = 0.9$)).

Efficacy of outlier model We often observe patterns in the residuals time series from online data model. The purpose of outlier model is to distill the pattern change, and discard the outlier effect. Figure 3 demonstrates the further decomposition of residual time series. We used a simple AR1 model to visualize the pattern that our data model fails to identify, and then categorize the data points into 11 outlier categories and non-outliers. The outlier effect is then computed and represented by the red line. After further removing the outlier effect and ARIMA effect, the remaining time series (orange line) resembles a white noise process. The adjusted time series, which is computed by subtracting the original residual (blue line) by outlier effect (red line), is fed into the online model to update the weight of parameters.

3.2 Sensitivity Analysis

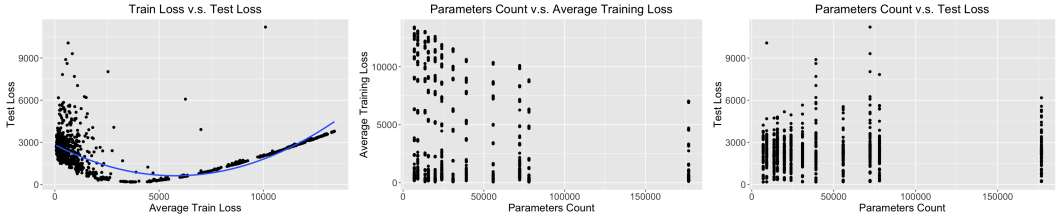

Figure 4: The relationship between the training loss and test loss, parameters count and training loss, parameters count and test loss.

Training loss vs testing loss Figure 4 illustrates the training loss and test loss in RNN models have a non-linear relationship, because there is an overfitting phenomenon where the training loss decreases while the test loss initially decreases and then increases after a point as the model becomes more complex. Regularization techniques such as dropout and weight decay can help to mitigate overfitting.
Model complexity Figure 4 shows the effect of the number of model parameters on the training loss. When the number of parameters is too low, the model may not be able to capture the complexity of the underlying data, resulting in a high training loss. As the number of parameters increases, the model becomes better able to capture the underlying patterns in the training data, leading to a decrease in the training loss. However, we observe that as the number of parameters continues to increase, the model may become too complex and start to overfit the training data, resulting in an increase in the test loss. This indicates that the model is not generalizing well to new, unseen data. When the number of parameters is very large, the model may start to become more general again, as it is able to capture the underlying patterns more accurately.

4 Conclusions and Future Work

4.1 Conclusions

Our experiemnts justify the design choice of Canary in general but we need to fine tune the details of Canary to counter possible problems including inaccurate predictions, unexpected predicted values, catastrophic forgetting and propagated/retained errors.
Why use an outlier model? One of the main benefits for explicitly identifying the positions of outliers is to perform causal inferences. One may associate the identified outliers with published market news to identify the root cause of outlier behaviors. Additional NLP models can be used to compute the similarity of the historic articles when new articles are published, and one may reasonably expect similar stock behavior if a close match is found.
Another reason to use an outlier model is to reduce the complexity of online data model. Deep autoregressive models such as RNN with 8 layers may encounter vanishing gradient problem. Even avoiding vanishing gradient problem, complicated online model may fail to generalize since it memorizes all rare behaviors and generates those patterns when performing predictions. Outlier model effectively reduces the required complexity of online data model by feeding it adjusted outlier-free stream of data so that online data model focuses on common and predictable behaviors.
Why use both offline and online data model? Offline model provides checkpoints for our data model. For outliers that we fail to identify, online model may be updated to accommodate the change of behaviors. When those behaviors are temporary, we can retrain the online model based on the offline model and the time series in between.
Caveats when updating with adjusted series. If outlier model is chosen poorly, the adjusted series may not be representative of the actual input stream, and thus contaminating the online data model. If the data model is chosen poorly and the residual time series still exhibit complex patterns, this will further decrease the accuracy of outlier detection model. We recognize this dilemma which appears in training Generator and Discriminator in GAN models. To address this problem, we propose training data model and outlier detection model jointly as described by Goodfellow et al. [2014] in training GAN models.

A Appendix

A.1 Outlier Categories

The nature and motivation of those categories were discussed extensively by Chen and Tiao [1990] and Fox [1972]. Chen and Liu [1993] proposed an iterative fitting algorithm that jointly updates the time series model and outlier models. Each observation is categorized by comparing the outlier effects under those categories and classified as the category with highest estimated effect, or classified as not an outlier if the none of the estimated effect passes a pre-defined threshold. Tsay et al. [2000] further extends the four categories of outliers in the context of multivariate time series cases.

A.2 Recent Machine Learning Work Of Outlier Identification

Recent work from machine learning communities proposed unsupervised deep learning models to identify outliers in time series data. For example, Kieu et al. [2019] proposed using autoencoder ensembles to compress the original time series and learn its underlying pattern. The difference between original time series and the constructed data from decoder can be categorized as outliers. Hundman et al. [2018] proposed using a mixture of parametic and nonparametic models to identify outliers. LSTM model is used to learn the sequential dependence of data, and a non-parametric dynamic error thresholding is used to identify outliers that cannot be explained by the model. More recently, Tuli et al. [2022] proposed identifying outliers using reconstruction method, but with the use of a state-of-the-art transformer network for encoding purposes.

A.3 Significance of Outlier Detection

Outlier detection in industry is one of the most import tasks, large companies such as Microsoft (Ren et al. [2019]) and Yahoo (Laptev et al. [2015]) developed automated data processing system to monitor their business data and trigger alerts upon detected outliers. More recently, Lai et al. [2021] developed open-sourced data processing pipeline with replaceable submodules to detect outliers from input data stream.

A.4 Decomposition of time series

Consider a univariate time series $\{y_t\}$ that can be decomposed as additively as follows: $y_t = z_t + e_t$, where $\{z_t\}$ is the explained by our data model, and $\{e_t\}$ is the unexplained residual time series. If data model is descriptive on the dataset, $\{e_t\}$ follows a white noise process with zero mean and σ_e^2 variance. If the residual time series contains multiple outliers, e_t can be further decomposed as follows:

$$e_t = \sum_{j=1}^{m} w_j \frac{A(B)}{G(B)H(B)} I_t(t_j) + \epsilon_t,$$

where $t_j, j = 1, ..., m$ correspond to m occurrences of outliers, $I_t(t_1) = 1$ if $t = t_1$ and 0 otherwise, w_j represents the initial impact of outlier, and $A(B), G(B), H(B)$ are polynomials of B that control the dynamic pattern of outlier effect. In our problem setting, the outlier locations, dynamic patterns unknown to us, and we wish to classify outliers into one of the four categories:

$$AO : \frac{A(B)}{G(B)H(B)} = 1 \tag{1}$$

$$IO : \frac{A(B)}{G(B)H(B)} = \frac{\theta(B)}{\alpha(B)\phi(B)} \tag{2}$$

$$TC : \frac{A(B)}{G(B)H(B)} = \frac{1}{1-\delta B}, 0 < \delta \leq 1 \tag{3}$$

$$\tag{4}$$

We note that LC is a special case of TC with $\delta = 1$. The original time series y_t can be written as $y_t = \hat{y}_t + \sum_{j=1}^{m} w_j \frac{A(B)}{G(B)H(B)} I_t(t_j) + \epsilon_t$.

A.5 Model Comparison

Model	Number of Layers	Hidden size	Train Loss (MSE)	Test Loss (MSE)
RNN	2	32	3525	209
RNN	2	64	2394	1693
RNN	4	32	3598	205
RNN	4	64	1276	1688
LSTM	2	32	3349	206
LSTM	2	64	1166 4	1157
LSTM	4	32	3421	200
LSTM	4	64	1163 4	1673

Table 1: Efficacy of offline trained one-directional RNN and LSTM model with drop out rate of 0.4, and 30 epochs during training.

References

C. Chen and L.-M. Liu. Joint estimation of model parameters and outlier effects in time series. *JASA. Journal of the American Statistical Association*, 88, 03 1993. doi: 10.2307/2290724.

C. Chen and G. Tiao. Random level-shift time series models, arima approximations, and level-shift detection. *Journal of Business & Economic Statistics*, 8:83–97, 02 1990. doi: 10.1080/07350015.1990.10509779.

A. J. Fox. Outliers in time series. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(3):350–363, 1972. ISSN 00359246. URL `http://www.jstor.org/stable/2985071`.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 387–395, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3219845. URL `https://doi.org/10.1145/3219819.3219845`.

K. Khare, O. Darekar, P. Gupta, and V. Attar. Short term stock price prediction using deep learning. In *2017 2nd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT)*, pages 482–486. IEEE, 2017.

T. Kieu, B. Yang, C. Guo, and C. S. Jensen. Outlier detection for time series with recurrent autoencoder ensembles. In *IJCAI*, pages 2725–2732, 2019.

K.-H. Lai, D. Zha, G. Wang, J. Xu, Y. Zhao, D. Kumar, Y. Chen, P. Zumkhawaka, M. Wan, D. Martinez, et al. Tods: An automated time series outlier detection system. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pages 16060–16062, 2021.

N. Laptev, S. Amizadeh, and I. Flint. Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1939–1947, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2788611. URL `https://doi.org/10.1145/2783258.2788611`.

M. Nabipour, P. Nayyeri, H. Jabani, S. Shahab, and A. Mosavi. Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis. *IEEE Access*, 8:150199–150212, 2020.

H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 3009–3017, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330680. URL `https://doi.org/10.1145/3292500.3330680`.

R. S. Tsay. Outliers, level shifts, and variance changes in time series. *Journal of forecasting*, 7(1): 1–20, 1988.

R. S. Tsay, D. Peña, and A. E. Pankratz. Outliers in multivariate time series. *Biometrika*, 87:789–804, 2000.

S. Tuli, G. Casale, and N. R. Jennings. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. 15(6):1201–1214, jun 2022. ISSN 2150-8097. doi: 10.14778/3514061.3514067. URL https://doi.org/10.14778/3514061.3514067.