



A Framework to Monitor Machine Learning Systems Using Concept Drift Detection

Xianzhe Zhou, Wally Lo Faro^(✉), Xiaoying Zhang, and Ravi Santosh Arvapally

Mastercard, 2200 Mastercard Blvd, O'Fallon, MO 63368, USA
{steven.zhou,wally.lofaro,evelyn.zhang,
ravisantosh.arvapally}@mastercard.com

Abstract. As more and more machine learning based systems are being deployed in industry, monitoring of these systems is needed to ensure they perform in the expected way. In this article we present a framework for such a monitoring system. The proposed system is designed and deployed at Mastercard. This system monitors other machine learning systems that are deployed for use in production. The monitoring system performs concept drift detection by tracking the machine learning system's inputs and outputs independently. Anomaly detection techniques are employed in the system to provide automatic alerts. We also present results that demonstrate the value of the framework. The monitoring system framework and the results are the main contributions in this article.

Keywords: Monitoring system · Concept drift · Machine learning · Anomaly detection · Framework

1 Introduction

The human body receives more sensory inputs than it can cope with so it has evolved to filter out most of them automatically, only focusing on what is anomalous. Enterprises have grown in complexity to a point where an analogy with the human body is apt. The need for autonomous agents monitoring enterprises for anomalies is felt as well. In this article, we look through the lens of concept drift to illustrate such a monitoring system applied to another machine learning based system. The monitored system can be a classifier, such as those presented in the work of Arvapally et al. [1], other machine learning based applications, or any source of data that displays traceable patterns over time. The monitoring system described herein currently operates at Mastercard.

The remainder of this article is organized as follows. A brief review of concept drift and related work is given in Sect. 2. Section 3 introduces framework of the monitoring system. In Sect. 4, the monitoring system is discussed in detail under the design framework along with the results.

2 Related Work

A machine learning model can malfunction every so often due to production issues or changes in input data. A monitoring system is needed to promptly flag model performance glitches. The essence of the monitoring system is to detect concept drift which is defined as [2]:

$$\exists X : P_{t_0}(X, y) \neq P_{t_1}(X, y) \tag{1}$$

where P_{t_0} denotes the joint distribution between input variable X and output variable y at time t_0 . There has been a substantial research into concept drift detection methodologies [2, 4–7] for various types of drifts [2], demonstrated by Figs. 1 and 2. Because of the slowly changing nature of the underlying data in our application, we are more concerned about sudden drifts than others.

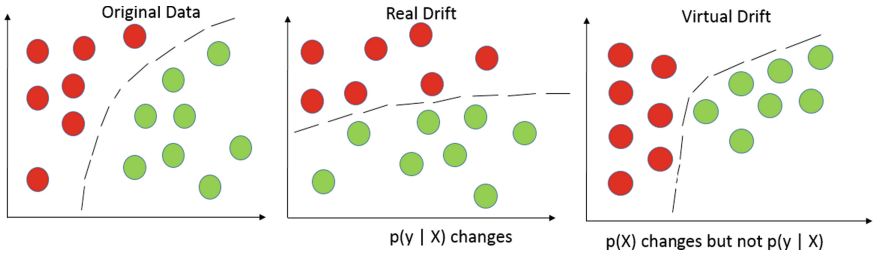


Fig. 1. Types of drift: circles represent instances; different colors represent different classes

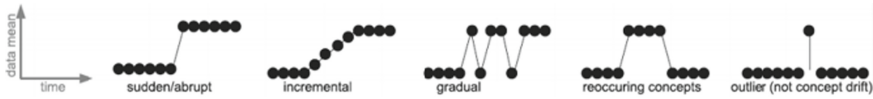


Fig. 2. Patterns of concept change

Moreover, probability chain rule,

$$P(X, y) = P(y|X) * P(X) \tag{2}$$

allows us to monitor a model’s inputs $P(X)$ and outputs $P(y|X)$ separately during operation. In other words, we can track changes in $P(X, y)$ by comparing $P_{t_0}(X)$ to $P_{t_1}(X)$, and $P_{t_0}(y|X)$ to $P_{t_1}(y|X)$. This division of supervision, or modular design, simplifies business operation.

In recent research, we noticed some researchers developing methods that can detect concept drift when concept drift is labeled [16]. This is based on application of both single classifier as well as ensemble classification techniques [16, 17].

These ensemble methods use multiple algorithms for concept drift detection following which a voting procedure will be conducted. These methods also study how to handle when limited labeled data is available. The underlying challenge here is, in most cases concept drift labels are either unavailable or very expensive to come up with. Availability of concept drift labels is very useful in developing accurate models. Our proposed framework is suitable when the concept drift labels are unavailable. In addition, our proposed framework can monitor both supervised and unsupervised learning models.

In addition, our framework does not specifically focus on any particular kind of dataset. Some recent research shows methods to detect concept drift in twitter [18]. Though the underlying nature of the data is streaming, the proposed methods in [18] are more applicable to specific Twitter data rather than general datasets. Some recent studies have shown fuzzy logic based windowing for concept drift detection in gradual drifts [15]. These adaptive time-windowing methods are definitely useful since they will increase the accuracy of the overall methodology. This fuzzy-logic based windowing concept can go with any existing framework. Finally, the modular design of our framework is novel and adds value. For the proposed framework, we applied anomaly detection techniques [8–11] to monitor model inputs and outputs independently, and to ultimately detect concept drift. There is a solid business reason to monitor these distributions. Namely, an event would have to manifest itself in such a way that leaves the distribution unchanged to go unnoticed. This seems unlikely.

3 Framework of the Monitoring System

As we have seen, the probability chain rule allows us to monitor the inputs and outputs of a machine learning system independently. As a result, the proposed framework consists of two components namely the input tracer and the output tracer. Figure 3 presents the proposed framework of the monitoring system. Input and output tracers monitor the anomalies in the input and output data respectively of a machine learning system.

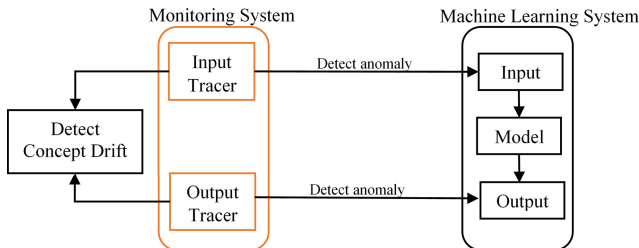


Fig. 3. Monitoring system

Although different techniques are employed in each tracer, both the input tracer and the output tracer follow the same design sequence, inspired by the

work of Trotta et al. [3]. Figure 4 presents three steps of the design sequence which is common to both tracers.

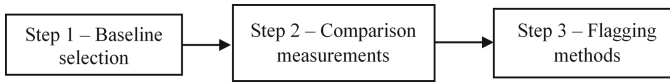


Fig. 4. Design framework

The three steps in the design sequence are explained below.

Step 1 - Detecting concept drift in data is the goal of our proposed framework. One way to detect drift is to track the distributional changes in data, where a baseline must be defined to make comparison. Domain understanding and data observation help in selecting a baseline. One has to consider three different criteria in selecting baselines: which historical values are included in the baseline? Whether to apply a smoothing technique? How frequently should the baseline be updated? Unavailability or inadequate labeled data is commonly observed. This leaves us with the option to leverage historical distributions of the data as our baseline.

Step 2 - After determining the baseline, a comparison measure is needed for benchmarking. Since there are several similarity measures for distributions available, the following criteria should be taken into account:

- Does the quantitative measurement align with the qualitative features of objects being measured?
- Can a threshold be created?
- The computational complexity to assess the scalability

Step 3 - In step 3, a flagging mechanism is designed and developed to automate the alerting effort. The goal of step 3 is to automatically alert on out-of-pattern values generated by step 2. The mechanism consists of two different functions. The first function is developed based on anomaly detection algorithms and the second function is based on heuristic rules provided by the domain experts. Identified anomalies are served as output to the business users to take informed business decisions. This step may involve further processing of the result, for example, providing a ranked list of anomalies.

Both the input and output tracers we are describing follow the three steps in the design sequence described above. Although these three steps are common to the tracers, the data fed into tracers exhibits different characteristics. Input tracer monitors the input data that goes into our machine learning system. While the output tracer monitors the output provided by the machine learning system. Detailed explanations of both input and output tracer are provided in the following sections.

4 The Monitoring System

This section presents the details of the developed monitoring system. The objective of the system is to monitor machine learning based systems. We are referring to a classifier which takes feature variables as inputs and maps the input data to one of several target classes. Each of the following sub-sections explains the business background, the solution, and the results.

4.1 Input Tracer

Business Background. Feature variables (FVs) are category profiles, created for each system account at Mastercard. Table 1 presents a snippet of how FVs are stored in a database. Equivalent to an index, a FV reflects recent activities of an account in a category. FVs are constructed periodically (weekly) for each account, and fed into a classifier. The data quality of FVs is crucial to the success of the classifier. The input tracer is designed to track the data quality of all FVs with the following expectations:

- Effective at alerting data quality incidents
- Scalable to handle large data volume
- Provide guidance for debugging when a data quality incident occurs

Table 1. Feature Variable (FV) examples in a database

Account number	FV	FV values	Creation date
1	FV 22	3.2	01-01-2018
1	FV 22	3.5	01-08-2018
1	FV 5	17.8	01-01-2018
1	FV 5	12.6	01-08-2018
2	FV 22	100.9	01-01-2018
2	FV 5	0	01-01-2018

Solution. The solution is based on the three steps presented in the framework.

Step 1 - Baseline selection: For the input tracer, the baseline is the FV value distribution from the previous week. This baseline is selected because of the following reasons:

- Values of a single FV are observed to have stable distribution over time.
- FV value distribution contains a large number of data points and hence noise can be tolerated.

- FV values can have cyclical fluctuations and hence a long term average is unsuitable.

Step 2 - Comparison Measurements: In order to track distribution changes, weekly FV value distributions are compared to those from the previous week. Multiple metrics are considered by leveraging information theory such as Kullback-Leibler divergence, Jeffrey's divergence and squared Hellinger distance. Despite being successful at signaling distribution changes, KL divergence and the other two measurements can only be computed after distributions are transformed to probability mass functions. The transformation is expensive when a distribution contains a large number of data points. This poses a scalability challenge in production and results in the abandonment of the metrics in this case. Then we turn to moment matching to compare two distributions while addressing the scalability challenge. If moments of two samples are matched, we assume both sample batches come from the same underlying distribution. Here, a moment M at time t is the expectation of a random variable X raised to a power p where p can be any positive integer, given N samples in a sample set:

$$M_{t,p} = \frac{1}{N} * \sum_N X_t^p \quad (3)$$

In this context, the first and second moments (mean and the mean of the squared values) are employed. After first and second moments are calculated for values of a FV over different weeks, week-over-week percentage changes are computed:

$$WOW\%_t = \frac{M_{t,p} - M_{t-1,p}}{M_{t-1,p}} \quad (4)$$

The percentage changes are given different weights based on business needs. In the developed system, the first moment WOW percentage change received 66% weight and the second moment WOW percentage change received 33%. The past 52 weeks of weighted WOW percentage changes are then fed into an anomaly detection model for training. The visualization of moments can help to spot anomalies, and provide clues to unravel a data quality incident. However, an automatic alerting/flagging system is needed, which is discussed below.

Step 3 - Flagging Methods: The objective of this step is to explore ways to provide automated alerts to the business users. For the input tracer different clustering algorithms are considered to enable automated alerts. We explored two different algorithms. DBSCAN, a density based clustering algorithm and K Means clustering algorithm. Application of these two algorithms present a need for post-clustering analysis and computation. This involves splitting and/or combining clusters to identify outlier cluster resulting in additional complexity. After exploration we settled with Isolation Forest algorithm [14] to identify outliers. Isolation Forest is an unsupervised anomaly detection method. Being an ensemble method, Isolation Forest is efficient at performing outlier detection. One reason Isolation Forest appeals is that it is a native binary classifier. Isolation Forest also controls the size of each cluster by user-defined parameters

such as contamination rate. Essentially, the algorithm performs anomaly detection using a ranking system. Each data point is ranked based on average path lengths, which is a representation of the likelihood of being an outlier. For more reading on Isolation Forest please see article [12,14].

One drawback of Isolation Forest in this framework is that normal cyclical variations can be misclassified as anomalies. When a data series has very little fluctuation, even a small change, which can be natural noise, makes a difference in the predictive outcome. This drawback is mitigated by employing a rule-based exception handler on the results produced by Isolation Forest. For example, in order to handle seasonal fluctuations, thresholds for anomaly are adjusted for certain periods of time. These rules are defined by domain experts based on operational needs.

Because of the reasons explained above, the alerting system consists of two layers. The first layer is an application of Isolation Forest and the second layer is a rule based exception handler. The second layer rules are applied on the output obtained from Isolation Forest algorithm (layer 1). The output from second layer is offered as results to the business owner. Details about layer 1 and 2 are presented below.

Layer 1: Isolation Forest

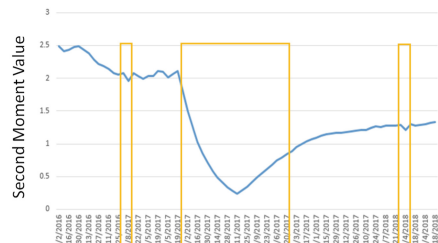
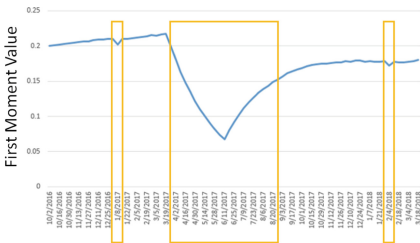
Training data: weighted 1st moment WOW% and 2nd moment WOW% of past 52 weeks

Model parameters:

- Number of estimators
- The number of features to train each base estimator
- Contamination (proportion of outliers in the data set)

Layer 2: Rules Layer

Rule based exception handler, used to adjust the predictive result from layer 1



(a) Time Series of First Moment, FV 77

(b) Time Series of Second Moment, FV 77

Fig. 5. Plots present time series of first and second moment of FV 77 (Color figure online)

Results. The above steps were applied on FV 77. Due to the absence of reliable labels, we rely on visual data analysis (see Fig. 5) and domain understanding to judge the effectiveness of the tracer. Figure 6 presents the results produced by the input tracer. The results in Fig. 6 are after the application of layer 1 and layer 2 rules. The input tracer (see Fig. 6) alerts on the following dates: 2017-01-08, 2017-01-15, 2017-03-26 to 2017-07-16, and 2018-02-04. Based on visual analysis, we notice there are three periods (highlighted by yellow boxes in Fig. 5a and b) during which anomalous behavior occurs. The alerts from the input tracer (Fig. 6) are consistent with our visual data analysis. In addition, the business owners are in agreement with the alerts produced by the system. The input tracer has been tested with a variety of FVs. The system is scalable and effective at capturing outliers.

Input Tracer Conclusion. The proposed input tracer is successful at alerting data quality incidents that occur among feature variables. Ranking data points for outliers with Isolation Forest helps business users prioritize and understand the data behind the drift. This provides information to users to investigate and debug when a data quality incident occurs.

4.2 Output Tracer

Business Background. Often it is possible to see the effect of a production incident on a classifier by examining a machine learning model's output. Given the slowly changing nature of the input data, a dramatic shift in the distribution of model outputs signals a potential issue/incident. An incident is an unplanned interruption to an IT service or reduction in the quality of the service impacting customers. A production issue/incident could be a result of one of several reasons. For example, applying the classifier to incorrectly scored accounts or misprocessed feature variables (input data i.e. FVs) may lead to incidents.

The focus of the output tracer is on the outputs produced by a machine learning system. The output tracer is therefore designed to identify anomalous classifier behaviors. By examining the distribution of all classes, the output tracer can alert out-of-pattern class distributions so that swift corrective actions can be taken.

Solution. The key observation and assumption here are similar to the one made by the input tracer: aggregate account activities for each account family should not change dramatically over time. The output from a machine learning system is categorized into one of the target classes (mapped classes). The mapped classes produced by the classifier are expected to have a stable distribution for each account family. This statement is true unless an incident or a technical issue occurs while the system is in use. The output tracer is turned into a problem of monitoring the distribution changes of classes over time. The solution follows the three steps outlined in the framework (Sect. 3).

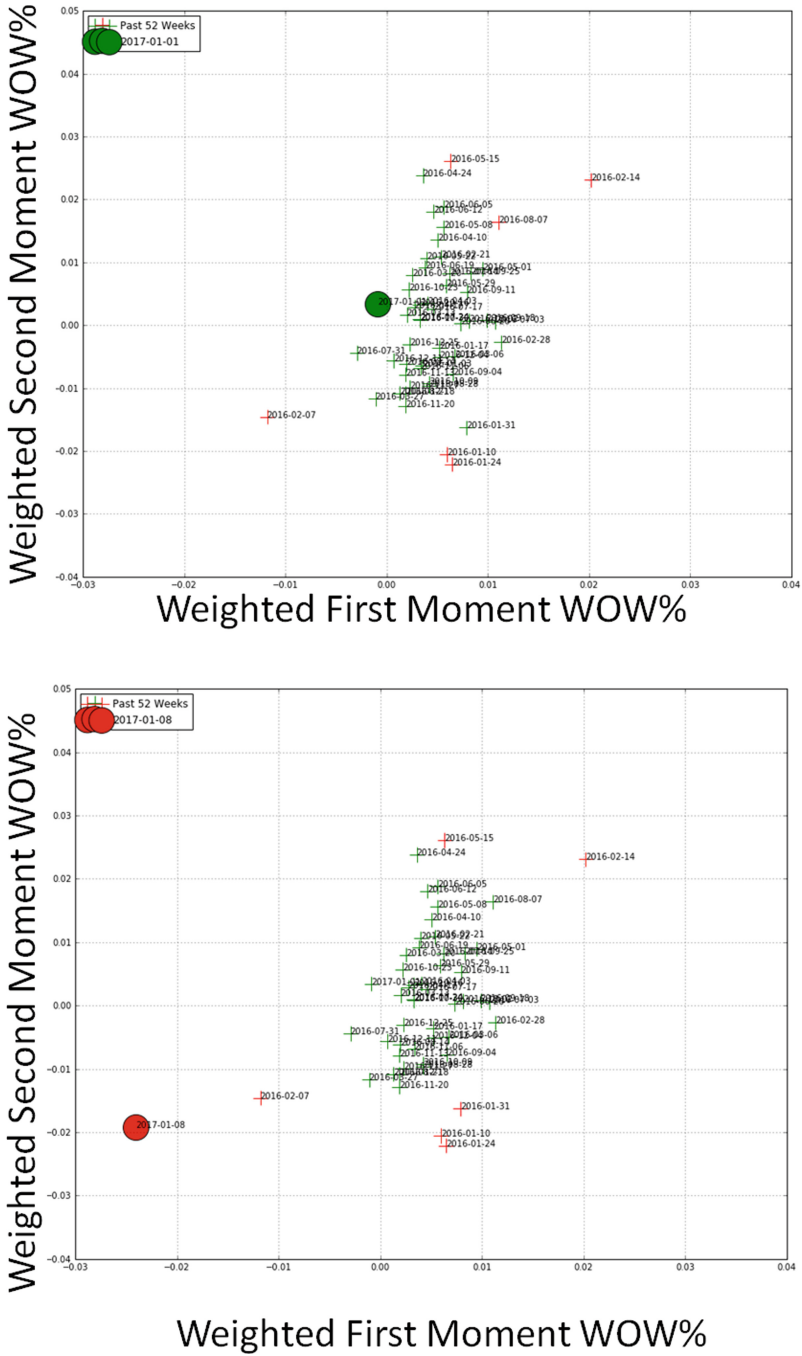


Fig. 6. Weighted First and Second Moment WOW% Scatter Plots, Colored Based on Isolation Forest Outcomes (Best Viewed in Color). *Circle: test data, Crosses: training data (prior 52 weeks), Green: inliers, Red: outliers* (Color figure online)

Step 1 - Baseline selection: The first step to identify anomalous behavior is to define what is normal. Taking both business context and potential seasonality into account, the baseline, or the reference distribution is defined by averaging past ten same day-of-week's class distributions. For example, given Monday 7/16/2018, the reference distribution is calculated by taking the average of class distributions of previous Monday (7/9), the Monday before (7/2) and so on. This granularity was determined by domain experts. In comparison to the input tracer, the baseline of the output tracer involves more computation, by averaging class distributions of the past ten same days-of-week. However, although the computational complexity is high in output tracer, the size of the output data is smaller. Therefore the scalability challenge is easier to handle. A large time window is selected to minimize the impact of noise because the number of data points in a class distribution could be small. On the other hand, the window should not be oversized, so that the latest trend can be reflected. Equal weight is given to each of the ten distributions, and hence no single distribution is dominating the reference distribution. Since data labels are unavailable, abnormally mapped classes are currently included in building the reference distribution. Abnormal class mappings can be removed in future after they are detected and recorded.

Step 2 - Comparison Measurements: Unlike feature variables (inputs), the class data (outputs) has less of a scalability challenge but is more volatile in nature. With those differences in mind, different similarity measurements and anomaly detection methods are evaluated and selected. Kullback-Leibler divergence, also known as relative entropy (RE), is chosen to measure the difference between the reference distribution $q(x)$ and the current distribution $p(x)$. The higher the relative entropy value is, the more skewed current distribution is compared to the reference distribution.

$$D(p||q) = \sum p(x) \ln\left(\frac{p(x)}{q(x)}\right) \quad (5)$$

For Eq. (5), $p(x)$ is the current distribution, and $q(x)$ is the reference distribution. The current distribution $p(x)$ is calculated at the end of each day.

Unlike squared Hellinger distance, KL divergence is not restricted between zero and one. A sizable distribution change will be translated to a change of large magnitude in KL divergence, which facilitates the creation of a threshold. KL divergence lacks symmetry but this can be solved by averaging $D(p||q)$ and $D(q||p)$. A small positive number replaces all zero probability densities.

Step 3 - Flagging Methods: Account families that have fewer observations tend to have more volatile class distributions which lead to higher RE values. So, relative entropy alone cannot be used to detect anomaly. As a result, a second factor is introduced to measure the size of an account family. The second factor is computed by taking the natural logarithm of the average number of activities performed under the account family. Visualization of RE values against the size of account family can be used to identify which account families have skewed class distributions. The objective of this step is to automate the alerting system using the two variables mentioned above.

The alerting system comprises of two layers: the first layer is based on DBSCAN. The second layer consists of human-defined rules. These rules are applied on top of the cluster results produced by DBSCAN. Details about layer 1 and 2 are presented below.

Layer 1: DBSCAN

The first factor – the RE values of all account families for one day

The second factor – the "size" of an account family derived from the number of activities under the account family

Model parameters:

- *The maximum distance (eps) for a neighborhood*
- *The minimum number of samples in a neighborhood for core points*

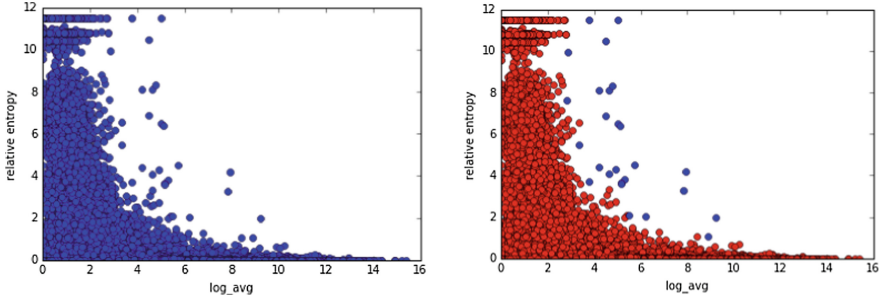
Layer 2: Rules Layer - rule based exception handler, used to adjust the predictive result from layer 1 and to handle cases in which a large number of account families go wrong.

The RE values and account family sizes are fed into a clustering algorithm to detect anomalies. In this case, DBSCAN (Density-based Spatial Clustering for Applications with Noise) [13] algorithm is tested and selected to detect anomalies. According to Scikit Learn, DBSCAN views clusters as high-density areas separated by low density areas [12]. One drawback of the algorithm is that DBSCAN does not necessarily produce two clusters. Based on the density of data points and parameters chosen, the algorithm may produce more than two clusters. In those circumstances, the cluster with the largest number of points is deemed as the "normal" cluster and all remaining clusters are considered anomalies. The parameters (thresholds) of DBSCAN are selected from both mathematical and business perspectives.

Other anomaly detection algorithms were also tested, such as isolation forest. However, the fact that isolation forest always selects a fixed percentage of anomalies does not apply to the condition here in the output tracer. The non-linear nature of the data (as shown in Fig. 7) also negatively impacts the performance of isolation forest.

Layer 2 is applied on the results produced by the layer 1. This will handle cases in which anomalies occur in large number of account families. Rules can also be added to accommodate special situations. For example, a rule restricting the size of an account family can be applied here to remove cases where the sample size is too small to have business values. The alerts from layer 2 are served as output to business users.

Results. Figure 7a presents relative entropy values against the natural logarithm of account family sizes, which are derived from the output of a machine learning system. The plot clearly shows a dense cluster with a few scattered points (see Fig. 7a). This phenomenon aligns with the common assumption of anomaly detection: anomalies are the minority group that behave substantially



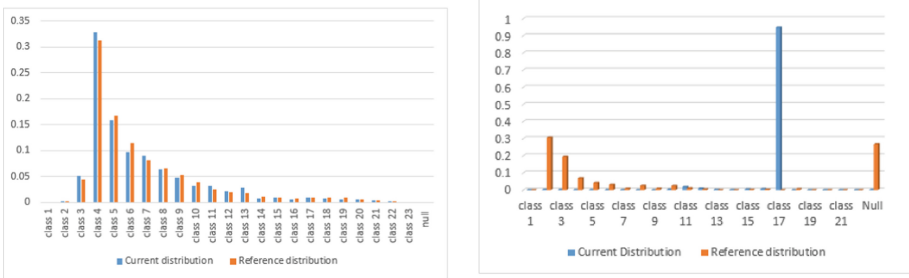
(a) Relative Entropy Values versus Natural Logarithm of Account Family Sizes (b) Anomalies versus Normal Points (Best View in Color)

Fig. 7. Plots present relative entropy and account family size before and after DBSCAN algorithm applied (Color figure online)

differently from the majority. Figure 7b presents a plot of data with labels produced by the DBSCAN algorithm. The figure shows two different clusters (red and blue). Data points colored blue are outliers captured by the DBSCAN algorithm and red points are inliers.

The results are then verified by comparing baseline distribution with current distribution. Points from Fig. 7b are randomly selected for verification. The current distribution and reference distribution are plotted side by side to check if the detected anomalies indeed display any anomalous behaviors. Figure 8a is an example of a normal case (red points in Fig. 7b). Figure 8b is an example of the distribution of a detected anomaly (blue points in Fig. 7b). From Fig. 8b, we can clearly notice how current distribution is significantly different from its reference distribution.

Since there is no labelled data, results are evaluated with feedback from the business. False positives are tuned to be as low as possible to avoid too many false alarms, whereas in the meantime incident information is being collected to evaluate false negatives.



(a) Distribution of a normal point (b) Distribution of a detected anomaly

Fig. 8. Current distributions versus reference distributions (Color figure online)

Output Tracer Conclusion. Timely response to operational issues or incidents is critical because relevant teams are often unaware of misbehaviors of a machine learning system until the impact becomes very large. As more machine learning systems are designed and deployed, a monitoring system is needed to ensure the systems function in the expected way. With the output tracer, business operation can not only receive timely alerts on potential misclassifications of activities, but also locate the anomalies immediately.

5 Conclusion and Future Research

In this article, we presented methods to detect and to alert when concept drift occurs. The input and output tracer monitor the data that goes into a classifier and the classes produced by the classifier respectively. And the modular design of the developed system ensures the two models work independently. The two cases supporting the proposed methods demonstrates success in alerting drifts. Receipt of these alerts allows users to study the data and monitor the system in detail, and this clearly has a huge benefit.

In the future, we would like to continue research in two different directions. On one hand, we will continue experimentation to study the relationship between properties of different data sets and drifts, especially when data changes are unstable. On the other hand, we would like to study how these alerts can assist in developing online machine learning algorithms. Exploring drift and anomaly detection with real time data streams would be interesting as well. In conclusion, concept drift has a huge role to play in resolving challenges in the area of machine learning model deployment and model maintenance.

References

1. Arvapally, R.S., Hicsasmaz, H., Lo Faro, W.: Artificial intelligence applied to challenges in the fields of operations and customer support. In: 2017 IEEE International Conference on Big Data (IEEE Big Data), Boston, pp. 3562–3569 (2017)
2. Gama, J., Zliobaite, L., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. *J. ACM Comput. Surv. (CSUR)* **46**, 44:1–44:37 (2014)
3. Pastorello, G., et al.: Hunting data rouges at scale: data quality control for observational data in research infrastructures. In: 2017 IEEE 13th International Conference on e-Science (e-Science), Auckland, pp. 446–447 (2017)
4. Gamage, S., Premaratne, U.: Detecting and adapting to concept drift in continually evolving stochastic processes. In: ACM Proceedings of the International Conference on Big Data and Internet of Thing, London, pp. 109–114 (2017)
5. Webb, G., Lee, L.K., Goethals, B., Petitjean, F.: Analyzing concept drift and shift from sample data. *J. Data Min. Knowl. Discov.* **32**(5), 1–21 (2018)
6. Gholipur, A., Hosseini, M.J., Beigy, H.: An adaptive regression tree for non-stationary data streams. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing (ACM), Coimbra, pp. 815–817 (2013)
7. Jadhav, A., Deshpande, L.: An efficient approach to detect concept drifts in data streams. In: IEEE 7th International Advance Computing Conference (IEEE IACC), Hyderabad, pp. 28–32 (2017)

8. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. *ACM Comput. Surv. (CSUR)* **41**(3), 15 (2009)
9. Ding, M., Tian, H.: PCA-based network traffic anomaly detection. *J. Tsinghua Sci. Technol.* **21**(5), 500–509 (2016)
10. Zhang, L., Veitch, D., Kotagiri, R.: The role of KL divergence in anomaly detection. In: *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, San Jose, pp. 123–124 (2011)
11. Laptev, N., Amizadeh, S., Flint, I.: 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 (ACM)*, Sydney, pp. 1939–1947 (2015)
12. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
13. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, pp. 226–231 (1996)
14. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation based anomaly detection. *ACM Trans. Knowl. Discov. Data (TKDD)* **6**(1), 3 (2012)
15. Liu, A., Zhang, G., Lu, J.: Fuzzy time windowing for gradual concept drift adaptation. In: *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Naples, pp. 1–6 (2017)
16. Geng, Y., Zhang, J.: An ensemble classifier algorithm for mining data streams based on concept drift. In: *10th International Symposium on Computational Intelligence and Design (ISCID)*, Hangzhou, pp. 227–230 (2017)
17. Hu, H., Kantardzic, M.M., Lyu, L.: Detecting different types of concept drifts with ensemble framework. In: *17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Orlando, pp. 344–350 (2018)
18. Senaratne, H., Broring, A., Schreck, T., Lehle, D.: Moving on Twitter: using episodic hotspot and drift analysis to detect and characterise spatial trajectories. In: *7th ACM SIGSPATIAL International Workshop on Location - Based Social Networks (LBSN)*, Dallas (2014)