Labelless Concept Drift Detection and Explanation

Conference Paper · December 2019

CITATIONS
0

READS
57

6 authors, including:

Shihao Zheng
Eindhoven University of Technology
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Mykola Pechenizkiy
Eindhoven University of Technology
226 PUBLICATIONS **4,825** CITATIONS
SEE PROFILE

Labelless Concept Drift Detection and Explanation

Shihao Zheng
Simon B. van der Zon
Mykola Pechenizkiy
Cassio P. de Campos
TU Eindhoven, 5600 MB, The Netherlands
s.zheng@student.tue.nl

Werner van Ipenburg
Hennie de Harder
Rabobank Nederland
Zeist, 3705 AR, The Netherlands
werner.van.ipenburg@rabobank.nl
hennie.de.harder@rabobank.nl

Abstract

The performance of predictive models used e.g. in fraud detection can deteriorate because of concept drift, i.e. changes in the data distribution. Existing concept drift detection methods assume that true labels become available immediately after predictions are made thus making it possible to monitor predictor's accuracy. However, in many banking applications true labels become known with a considerable delay. We propose a new **L**abelless **Co**ncept Drift **D**etection and **E**xplanation Framework (L-CODE) to bridge this gap. L-CODE allows to localize and visualize detected drifts. We illustrate its promising performance with synthetic and real-world datasets, including the fraud detection case study.

1 Introduction

Common classification models are assumed to be trained on data that are sufficient and representative of the underlying unknown distribution. However, in real-world scenarios, the joint distribution of features and labels $p(X, y)$ is not stationary but drifting from time to time [1]. This phenomenon, referred to as concept drift, can severely deteriorate the predictive performance of an existing model. Fraud detection is a typical example where concept drift may happen because of the adversarial behavior, or new kinds of fraud appearing or policy changes by the regulator or the bank. From the Bayesian perspective, concept drift can manifest itself in two types of changes [1, 2]: 1) real concept drift–change in the posterior distribution $p(y_i|X)$; 2) virtual concept drift–change in the distribution of one or several classes $p(X|y_i)$ (or in $p(X)$).

Most methods for real concept drift detection assume that a true label becomes known after the classifier casts a prediction, and signaling of possible drifts is possible based on classifier accuracy monitoring, e.g. with the ADWIN [3], DDM [4], or PH [5] detector. But in many real applications, true labels might be unavailable or become known with a considerable delay [6]. Even if true label were available, after a drift is detected it can be desirable to inspect it for possible root causes. Methods for virtual drift detection (e.g. CNF [7] and HDDDM [8]), which monitor feature distribution, do not use true labels, but the alerted drifts may have no effect on the classification task. Recent labelless methods attempt to monitor the classifier output prediction $\hat{y}$ (or estimated posterior probability $\hat{p}(y_i|X)$) to make sure detected drifts are related to classification task (e.g. CDBD [9] and MD3 [10]) but they are not equipped to provide interpretation for possible root causes of the drift.

We propose L-CODE: **L**abelless **Co**ncept Drift **D**etection and **E**xplanation Framework. Our drift detection methods tracks changes in the space of Shapley values that capture individual feature contributions to the predictions cast by a classifier [11, 12]. Our method is efficient because we can track changes on each (Shapley) feature separately, and effective, because each Shapley feature captures interactions of the original features wrt classifier's output. Besides these advantages for drift detection, L-CODE facilitates in-depth inspection of the source of the drift alert. We call out approach labelless because true labels are not used for detection or explanation of the drifts.

NeurIPS 2019Workshop on Robust AI in Financial Services, Vancouver, Canada.

2 L-CODE Framework

L-CODE conducts univariate chunk-based drift detection by monitoring the fluctuations of these Shapley values, in combination with changes of the feature distribution in reference and current windows. Here, we assume that drift found in one feature is enough to trigger an alarm and update the model. On top of the detected drift, L-CODE provides three-levels visualization with different granularities.

In the following, We motivate the use of Shapley values and then introduce our approaches for detecting and explaining drifts.

Shapley feature space. Shapley values, originating from the cooperative game theory, can explain the output of any machine learning predictor in terms of each feature contribution to predictor's output. In general case, Shapley values are expensive to compute. There are efficient approximations. However, for decision-tree based methods it is possible to compute Shapley values precisely efficiently that is known as SHAP [12].

Several properties of the Shapley values justify their use for tracking concept drift in a univariate manner: 1) combining local explanations of each prediction allows us to capture global patterns between feature space X and the predictions $\hat{y}$. Hence, we can use the Shapley value to vicariously track the change of $p(X, y)$; 2) The Shapley values of one feature can be decomposed into the main effect and its interaction effect with other features. Hence, even if we monitor each feature separately we can still obtain multivariate changes based on its Shapley values.

Let $(X_1, ..., X_N)$ be a sequence of data with labels $(y_1, ...y_N)$ to be used to train an initial *classifier* $\hat{f}$, where X has d dimensions, y is restricted to binary classes and N is the predefined chunk size. Next, an *explainer* g is obtained based on $\hat{f}$ (we use the SHAP library [12]). Classifier $\hat{f}$ and explainer g stay fixed until any model adaptation takes place. By inputting $X_1, X_2, \ldots$ into g, corresponding Shapley values $S_1, S_2, \ldots$ can be obtained and Shapley feature space formed.

Drift detection. The general drift detection workflow is shown in Figure 1. At time t, we are interested to detect whether there are differences between the reference and current windows. The reference window consists of (X_{Ref}, S_{Ref}), where $X_{Ref} = (X_{lambda}, ..., X_t)$ and $S_{Ref} = (S_{lambda}, ..., S_t)$. Here, $lambda$ is the initial point or the past detected drift position. Similarly, the current window is (X_{Cur}, S_{Cur}), where $X_{Cur} = (X_{t+1}, ..., X_{t+N})$ and $S_{Cur} = (S_{t+1}, ..., S_{t+N})$ with chunk size N. If there is a significant difference between (X_{Ref}, S_{Ref}) and (X_{Cur}, S_{Cur}), then we say that t is a drift position and X_{Cur} is a chunk with drift. Then we can request labels of X_{Cur} to update $\hat{f}$ and g or apply any other drift handling strategy. If there was no drift alert, the reference window will be extended by adding (X_{Cur}, S_{Cur}).

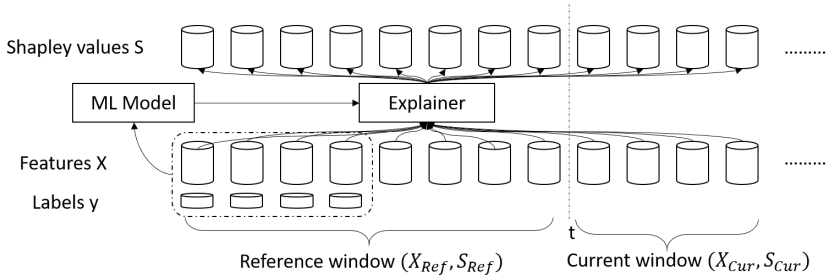


Figure 1: Drift detection workflow

We compare the reference and current windows, per each feature in parallel. For the i^{th} feature x^i, we are comparing (X^i_{Ref}, S^i_{Ref}) and (X^i_{Cur}, S^i_{Cur}), which are formed by selecting the i^{th} feature's feature values and Shapley values from the reference and current window. We first calculate the expected Shapley value distribution for X^i_{Cur} by extrapolating S^i_{Ref} based on X^i_{Ref} to X^i_{Cur}, which is then compared against the observed Shapley value distribution S^i_{Cur}, using two-sample t-test. Below we explain the intuition behind this method and how we compute the expected Shapley distribution.

Let $\{x_k^i \mid k \in [1, m]\}$ denote the k-th unique feature value of x^i (with a total of m unqiue feature values for x^i). Each feature value x_k^i has its corresponding Shapley values $\phi_{i(k)}$. Note that one feature value potentially has many different Shapley values as a result of interaction with other features [13]. We denote the Shapley distribution of one feature value x_k^i as $p(\phi_{i(k)})$ and of the feature x^i as $p(\phi_i)$.

We assume that if the distributions of one feature value x_k^i are the same in reference and current windows, then its corresponding Shapley distributions $p_{Ref}(\phi_{i(k)})$ and $p_{Cur}(\phi_{i(k)})$ should originate from the same distribution. This has two implications: 1) if $p_{Ref}(\phi_{i(k)})$ and $p_{Cur}(\phi_{i(k)})$ originate from different distributions, it indicates multivariate changes since the Shapley distribution also includes interaction information. 2) the change of sample size of x_k^i in two windows does not indicate change if $p_{Ref}(\phi_{i(k)})$ and $p_{Cur}(\phi_{i(k)})$ originate from the same distribution. Hence, if we want to use Shapley distribution to test if feature x_i has changes in two windows, we first need to calculate the expected Shapley distribution $p'_{Cur}(\phi_i)$ by adjusting the sample size of different feature values, then check if $p'_{Cur}(\phi_i)$ and $p_{Cur}(\phi_i)$ are drawn from the same distribution.

Since we use two-sample t-test, we only need to compute the expected mean and standard deviation of the Shapley distribution. For x_k^i, its corresponding Shapley distribution $p(\phi_{i(k)})$ will be (μ_k, σ_k). We expect that the mean and standard deviations of the Shapley values of x_k^i should be the same between the reference and current windows ($\mu_{k,expected}^{Cur} \approx \mu_k^{Ref}$ and $\sigma_{k,expected}^{Cur} \approx \sigma_k^{Ref}$). Then we can obtain the expected Shapley distribution of feature x_i at the current window, $p'_{Cur}(\phi_i) \sim (\mu_{expected}^{Cur}, \sigma_{expected}^{Cur})$ by adjusting the sample size of different feature values (Eq. 1 and 2):

$$\mu_{expected}^{Cur} = \frac{\sum_{k=1}^{m} \mu_k^{Ref} \cdot n_k^{Cur}}{\sum_{k=1}^{m} n_k^{Cur}} \tag{1}$$

$$\sigma_{expected}^{Cur} = \sqrt{\frac{\sum_{k=1}^{m} n_k^{Cur} \cdot {\sigma_k^{Ref}}^2 + \sum_{k=1}^{m} n_k^{Cur} \cdot (\mu_k^{Ref} - \mu_{expected}^{Cur})^2}{\sum_{k=1}^{m} n_k^{Cur}}}, \tag{2}$$

where n_k^{Cur} is the sample size of x_k^i in current window. If $(\mu_{expected}^{Cur}, \sigma_{expected}^{Cur})$ and $(\mu^{Cur}, \sigma^{Cur})$ have a statistically significant difference, then the drift is alarmed.

Note that to make sure that the Shapley distribution $p(\phi_{i(k)})$ related to feature value $x_k^i, k \in [1, m]$ is representative and can give good enough mean and deviation estimates, we need at least 30 samples of x_k^i. And to make sure all features have finite unique values we need to do binning on the numeric features where bin numbers will be set to $\lfloor\sqrt{N}\rfloor$ and N is the chunk size. Accordingly, we have $N \geq \max\{30m, 30\lfloor\sqrt{N}\rfloor\}$. If all the features are continuous, the chunk size cannot be less than 900.

It is also worth noting that when the expected and observed Shapley distributions of a feature have a significant difference, the feature distribution may have changes or not between reference and current windows. When there is also a change in feature distribution, it is trivial to conclude that there is drift. When there is no change in feature distributions, it implies that its interactions with other features have changed. For example, we have a binary classification task where $gender = male \vee education = educated$ is class 1, otherwise 0. We assume that the initial classifier $\hat{f}$ is optimal and correspondingly from the explainer g we can obtain the interaction information between features. If we have 50 educated males and 50 non-educated females in reference window, and 50 non-educated males and 50 educated females in current window, then if we monitor feature distributions of either $gender$ ($male$, $female$) or $education$ ($educated$, $non-educated$), we cannot detect any changes. But if we look into the Shapley distributions of $gender$ in two windows there is a clear difference since the Shapley values of $gender = male$ when it co-occurs with $education = educated$ or $education = non-educated$ will be totally different (the same applies to $gender = female$).

Explanation of detected drift. L-CODE provides three levels of drift information to assist end-users (see Figure 2). First, we know the chunks that have the drift. Second, we can target the features that are responsible for signaling the drift in this chunk. Third, we can go deeper and locate the feature values that have severe changes between the expected and observed Shapley distributions.

Correspondingly, for the three levels of drift information, we provide different forms of visualizations to assist end-user's understanding of the drift. We illustrate their use in Section 4.

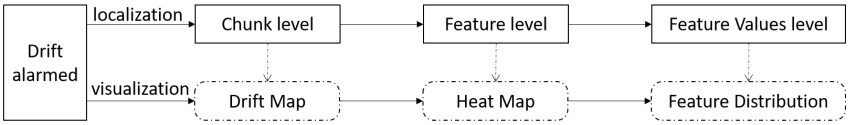


Figure 2: Detected drift locatization and explanation

3 Experiments on drift detection

The goal of our drift detection experiments in to investigate the performance of L-CODE. We conduct experiments on real-world datasets with synthetic drift (i.e. we know the exact positions of drifts) and real-world benchmark datasets that are known to exhibit drifts, but we do not know their positions.

We use a static classifier as a baseline and two existing labelless drift detection methods as competitors. Thus we compare four approaches: 1) No change–after building initial classifier, no drift detection or model update will be conducted; 2) HDDDM–Hellinger Distance Drift Detection Method [8] only monitors the changes in feature space; 3) MD3–Margin Density Drift Detection Method [10] tracks changes in the number of samples that fall into the margin of the classifier (e.g. SVM) to detect drift based on the model output $\hat{p}(y_i|X)$; 4) L-CODE.

For the experiments on the real-world benchmark dataset, we add three more drift detection methods (ADWIN [3], DDM [4] and PH [5]) which are based on monitoring classification accuracy, i.e. they are not competitors, but are expected to show better performance.

We set the initial classifier $\hat{f}$ as a random forest with 20 trees and the explainer g is obtained by using Tree SHAP [12]. For each dataset, we have to do binning to make sure each feature has limited number of unique values. Then the minimum chunk size N is specified for each dataset. The reference window will enlarge if no drift is signaled in the current window. If drift is detected in the current chunk, its labels will be requested, and $\hat{f}$ and g will be updated. The significance level of two-sample t-test is set to 0.001 for all the experiments. The only difference for HDDDM's setting is that it does not have the explainer. MD3 is slightly different since it detects drift instance by instance. If drift is detected, next N labeled data will be requested to update the classifier. To make the results comparable, we report the average accuracy of MD3 in each data chunk.

3.1 Real-world dataset with synthetic drift

Our first goal is to check whether L-CODE can detect the manually induced drift. Since in concept drift research we do not have real-world dataset with known drift positions, by introducing drift in real-world dataset we set a controlled yet realistic environment to test whether drift detection is effective. If L-CODE can detect drift point we manually induced, the performance of the classifier should recover after updating. Thus, we check both, the detection accuracy and its effect on classification performace.

Datasets: Two UCI datasets [14]: Adult and Bank. We follow [10, 15] to manually introduce $p(X|y_i)$ drift in the middle of the dataset by swapping the top 25% features. The chunk size of both datasets are set as 2500.

Results: All manually induced drifts in two datasets are detected by L-CODE. In Adult dataset (the first row of plots in Figure 3), L-CODE has the highest average accuracy (0.912) among the four methods (the second is MD3 with 0.893). Since all the methods have the same setting on the random forest classifier, the differences are due to the different drift position they found. Our method finds drift in chunk 9 and chunk 10 consecutively and updates the classifier twice. One reasonable explanation is that the classifier updated on chunk 9 contains the old and new concepts, so a significant difference between the expected and observed Shapley distribution is still found on chunk 10. The drift signaled on chunk 5 by HDDDM is an obvious false alarm. One possible explanation here is that there are changes in feature space in chunk 5 but these features are irrelevant to the classification task. Consequently, updating classifier on chunk 5 will not help recovery the performance of classification.

L-CODE does not alarm drift in chunk 5. This confirms the expectation that L-CODE is focused on detecting changes relevant to the classifier performance. The drift position of MD3 falls into chunk 10. It has a delay in the detection of drift but after updating the classifier, the accuracy recovers as expected and reaches the same level as with L-CODE.

In dataset Bank (the second row of plots in Figure 3), L-CODE and HDDDM have the highest average accuracy (0.896). Their drift positions are exactly the same. The drift position of MD3 falls into chunk 10, which has a delay in drift detection but after updating the classifier, the performance of classification reaches the same level as our method.

These results confirm our expectations on the performance of L-CODE.

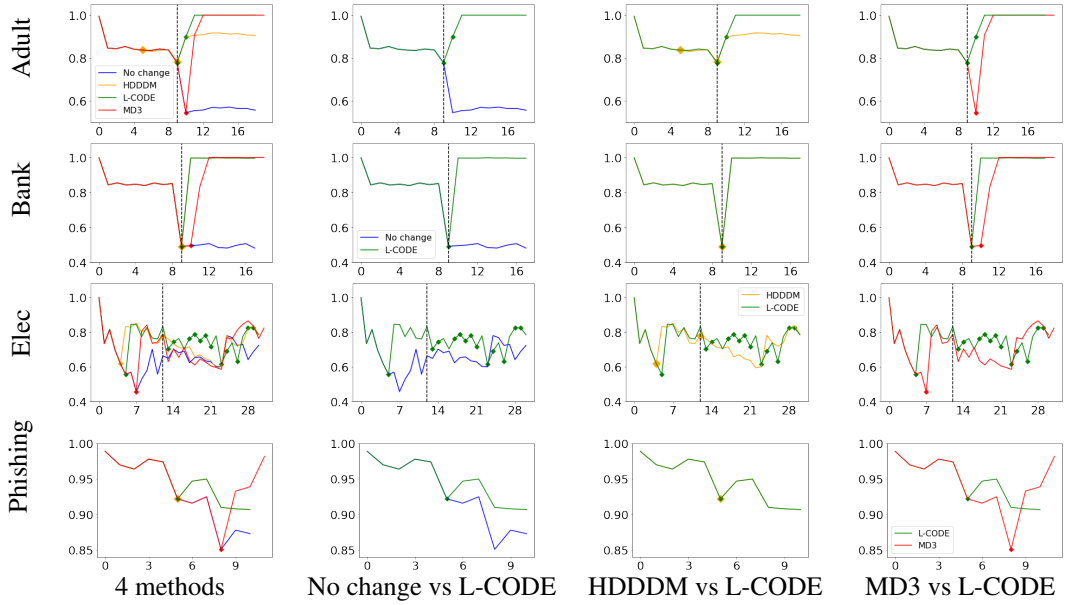


Figure 3: Performance of the baseline and three labelless detectors (chunk number vs. accuracy). Diamonds indicate the detected drift; black dash lines indicate real drift positions.

3.2 Real-world benchmark dataset with real drift

Our next goal is to check the effectiveness of L-CODE method on real-world datasets that exhibit drift. Since drift positions in these datasets are unknown, we use average accuracy as an indirect indicator of drift detection performance. If the drifts are correctly detected, the accuracy of the classifier should recover by retraining.

Datasets: Electricity (Elec) and UCI Phishing. The Elec dataset can be obtained from MOA [16]. From [17], we know that after the 2nd of May 1997 the drift should increase substantially. We keep 7 features and remove the $date$ feature. We set the chunk size as 1440 which is approximately the length of one month. We set the chunk size in the Phishing dataset as 1000.

Results: L-CODE has the highest average accuracy in comparison with the other labelless methods on Elec and Phishing datasets (see Figure 3). Specifically, for the Elec dataset, L-CODE has the highest accuracy 0.748 among labelless methods. L-CODE signals drifts more often comparing to other labelless methods; here our results align with the prior knowledge that after 2nd of May 1997 (black dash line in plots related to Elec in Figure 3), the drift should increase substantially. Our method keeps the average accuracy at a relatively stable level and higher than the other methods all the time except after chunk 24. This may due to the seasonality of the drift so that other methods which still keep the old concept can perform well again.

We also compared the average accuracy of the classifier with L-CODE detector against three state-of-the-art drift detectors that use accuracy (i.e. require true labels) to track concept drift: ADWIN, DDM, and PH. On the Electricity dataset, the highest accuracy is achieved by NB+DDM (0.838) which is 0.09 higher than our method. On the Phishing dataset, the best performance is achieved by

NB+ADWIN (0.920) which is lower than ours (0.943). And with L-CODE the classifier used only 10% labels as compared to the other approaches.

These results show that L-CODE outperforms other labelless methods and shows competitive performance with state-of-the-art methods that require immediate availability of true labels.

4 Experiments on explanation of detected drift

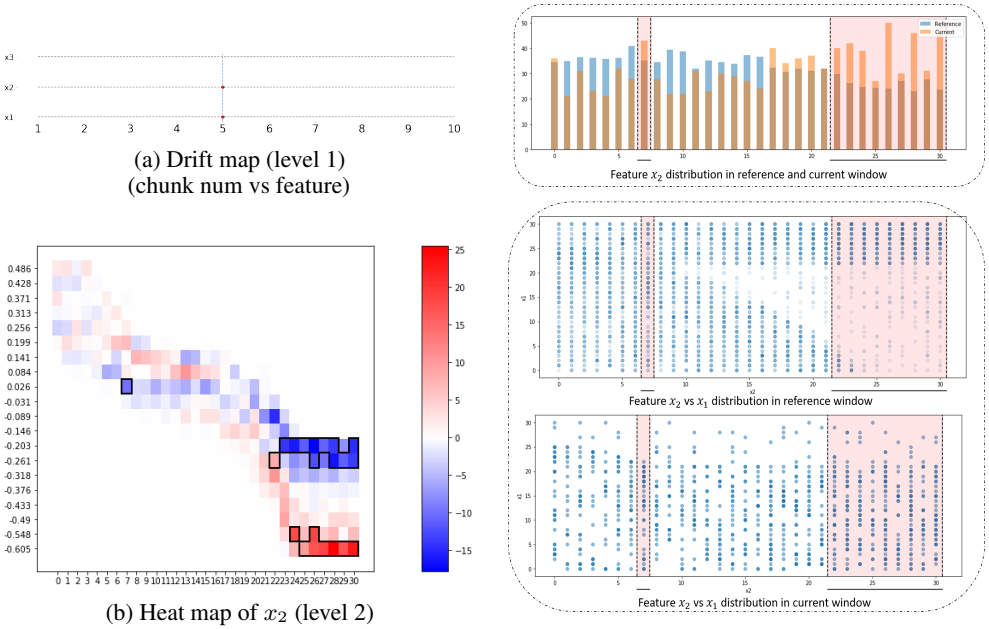
We conduct two sets of experiments to demonstrate how we explain the detected drift. First, with use of one synthetic dataset we illustrate how the three-level visualization can help and then provide evidence with a case study on the fraud detection dataset provided by a Dutch bank.

4.1 One illustrative example on explanation of detected drift

Dataset: We follow [18] to introduce $p(X|y_i)$ drift in the middle of the SEA dataset (with 10000 instances). It has three features (x_1, x_2, x_3). If $x_1 + x_2 < 7$, it will be class 1 otherwise 0. We set $p(x_1 < 5|y = 0) = 0.1$ before drift point and $p(x_1 < 5|y = 0) = 0.9$ after drift point. Chunk size is set to 1000.

Results: We can see three-level visualizations of SEA dataset in Figure 4. From the drift map (level 1) in Figure 4(a), we can see that features x_1 and x_2 have drift in Chunk 5, which is the middle of the dataset. The heat map (level 2) of feature x_2 are shown in Figure 4(b). Each position in the map shows the difference between the observed and the expected frequency of samples with specific feature values and Shapley values. The encircled areas show the positions that have large differences. We can see that large x_2 values can be responsible for the detected drift.

Note that we did not explicitly introduce drift in feature x_2 but we still find drift in it. In this situation, the drift in feature x_2 is caused by its interaction with feature x_1. The distribution of x_2, the scatter plot of x_2 and x_1 in the reference and current windows (level 3) are shown in Figure 4(c). The drift feature values of x_2 are highlighted in the pink area. The possible reason of drift here is that most large x_2 values were co-occurring with large x_1 values in the reference window while in current window large x_2 values are co-occurring more frequently with smaller x_1 values.



(a) Drift map (level 1)
(chunk num vs feature)

(b) Heat map of x_2 (level 2)
(feature values vs Shapley values)

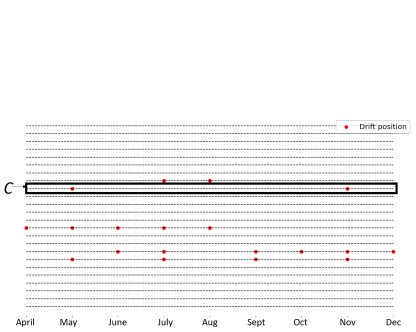
(c) Feature distributions (level 3)

Figure 4: SEA dataset three-level visualization

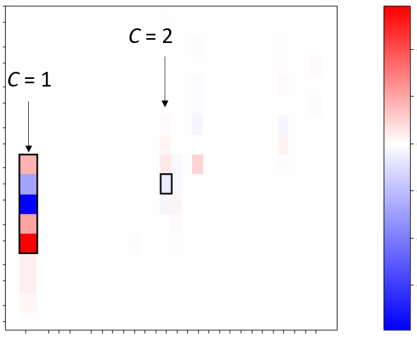
4.2 A case study on the fraud detection dataset from a Dutch bank

Settings: We choose a period in between January and March in 2017 to build an initial fraud detection model and we set this range as the reference window. Then we set samples of each month from April to December as current windows. In this case we do not update the model but only provide explanations on the drifts to assist domain experts' decision on when to update the model. For privacy concerns, we hide the numbers on all figures and add some annotations.

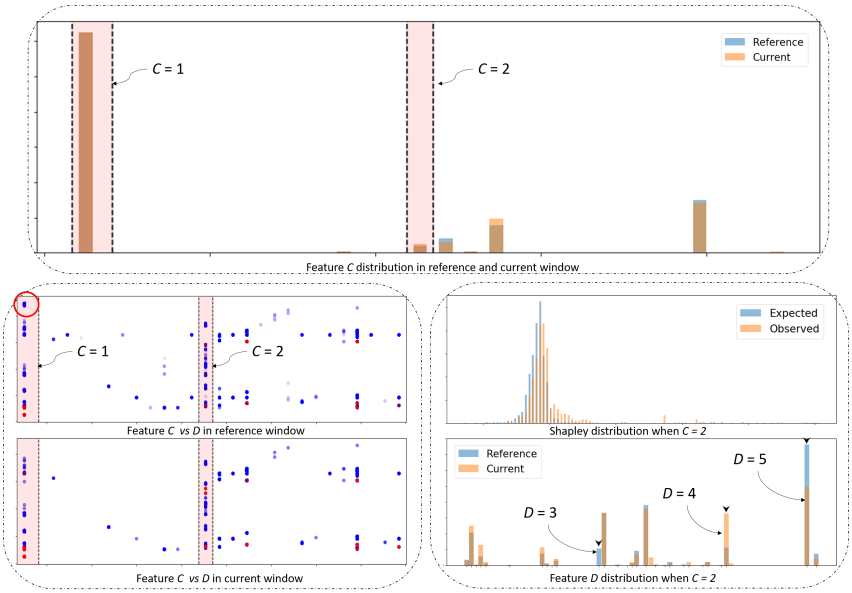
Results: In Figure 5(a), we can see the drift map (level 1). Five features have drift, which indicates that these features have different patterns in some months of the year comparing to the reference window. Here we pick one feature C as an example and go further for the next levels of visualization.



(a) Drift map (level 1)
(month vs feature)



(b) Heat map of C in May (level 2)
(feature values vs Shapley values)



(c) Feature distributions (level 3)

Figure 5: Dutch bank dataset three-level visualizations

Then we can see the heat map (level 2) of C in May in Figure 5(b). The encircled areas in the heat map show the positions that have high differences between the observed and expected frequency. Then we can target the drift feature values: 1 and 2 (from left to right). Apparently, $C = 1$ has more negative contributions than we expected and $C = 2$ has more positive contributions than we expected. Then we know that the proportions of different type of interactions for these two features values have changed.

We can see the distribution of C (level 3) from the left plot in Figure 5(c). Drift feature values 1 and 2 are highlighted in the pink range but from this distribution, we cannot find clear changes. Then we find that one of the top interactive features of C is D. We can see how the distributions between C and D change from the reference window to the current window at the bottom left plot in Figure 5(c). Since we actually know the labels of the test dataset, we also visualize all the fraud transactions with color red on top of the scatter plots to assist our understanding of the drift. When $C = 1$, one type of interaction between C and D is missing in the current window (see the red circle). When $C = 2$, there are some slight changes in the position of frauds between reference and current windows.

Now, we look further in the drift feature value $C = 2$. At the bottom right in Figure 5(c), we can see the expected and observed Shapley distributions of $C = 2$, and the reference and current feature distributions of D when $C = 2$. We can see that $C = 2$ has more positive contributions than we expected in May. And the possible explanation is that $C = 2$ is co-occurring more frequently with $D = 4$ and less frequently with $D = 3$ and $D = 5$.

These results show that L-CODE can provide in-depth explanation on the root causes of the detected drift via three-level visualization in the context of fraud detection.

5 Conclusion

When it comes to drift detection, it is usually not realistic to rely on the immediate availability of true labels for the new-coming data points (transactions). Thus, in real applications it is often impossible to use drift detection methods that are based on monitoring model accuracy. Other drift detection methods that monitor changes in the feature space, may generate alerts on various changes in the data distribution, including those that do not affect the performance of the prediction model. We show that our proposed L-CODE framework can effectively utilize information captured in Shapley values to focus only on those changes in the data which are relevant to the behaviour of the predictive model.

We demonstrated empirically that L-CODE consistently results in better performance of predictive models that need to operated under concept drift. L-CODE outperformed state-of-the-art labelless methods (Figure 3), and shows competitive performance in comparison with the detection methods which rely on availability of true labels. Furthermore, we demonstrated the use of the three-leveled visualization functionality enabled by L-CODE to make the drift detection process more transparent and interpretable, allowing the user to navigate from the Shapley distribution space to the corresponding source data (Figure 4 and 5).

We plan to study the applicability of L-CODE is different application settings, e.g. when we expect drifts or change points to reoccur [19, 20].

References

[1] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):44, 2014.

[2] Bartosz Krawczyk, Leandro L Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

[3] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[4] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[5] Ewan S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[6] Indre Žliobaite, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. In *Big Data Analysis: New Algorithms for a New Society*, pages 91–114. Springer, 2016.

[7] Anton Dries and Ulrich Rückert. Adaptive concept drift detection. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2(5-6):311–327, 2009.

[8] Gregory Ditzler and Robi Polikar. Hellinger distance based drift detection for nonstationary environments. In *2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, pages 41–48. IEEE, 2011.

[9] Patrick Lindstrom, Brian Mac Namee, and Sarah Jane Delany. Drift detection using uncertainty distribution divergence. *Evolving Systems*, 4(1):13–25, 2013.

[10] Tegjyot Singh Sethi and Mehmed Kantardzic. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82:77–99, 2017.

[11] Igor Kononenko et al. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11(Jan):1–18, 2010.

[12] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[13] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

[14] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[15] Indre Žliobaite. Change with delayed labeling: When is it detectable? In *2010 IEEE International Conference on Data Mining Workshops*, pages 843–850. IEEE, 2010.

[16] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

[17] Michael John Roarty. *Electricity Industry Restructuring: The State of Play*. Department of the Parliamentary Library, 1998.

[18] Shuo Wang, Leandro L Minku, and Xin Yao. A systematic study of online class imbalance learning with concept drift. *IEEE transactions on neural networks and learning systems*, 29(10):4802–4821, 2018.

[19] Alexandr V. Maslov, Mykola Pechenizkiy, Yulong Pei, Indre Zliobaite, Alexander Shklyaev, Tommi Kärkkäinen, and Jaakko Hollmén. BLPA: bayesian learn-predict-adjust method for online detection of recurrent changepoints. In *International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 1916–1923, 2017.

[20] Alexandr V. Maslov, Mykola Pechenizkiy, Indre Zliobaite, and Tommi Kärkkäinen. Modelling recurrent events for improving online change detection. In *Proceedings of the SIAM International Conference on Data Mining, SIAM 2016 Miami, Florida, USA, May 5-7, 2016*, pages 549–557, 2016.