Uncovering Drift in Textual Data: An Unsupervised Method for Detecting and Mitigating Drift in Machine Learning Models

A PREPRINT

Saeed Khaki Amazon sakhaki@amazon.com Akhouri Abhinav Aditya Amazon akhoura@amazon.com Zohar Karnin Amazon zkarnin@amazon.com Lan Ma Amazon mamlm@amazon.com

Olivia Pan Amazon olipan@amazon.com Samarth Marudheri Chandrashekar Amazon smarudh@amazon.com

September 8, 2023

ABSTRACT

Drift in machine learning refers to the phenomenon where the statistical properties of data or context, in which the model operates, change over time leading to a decrease in its performance. Therefore, maintaining a constant monitoring process for machine learning model performance is crucial in order to proactively prevent any potential performance regression. However, supervised drift detection methods require human annotation and consequently lead to a longer time to detect and mitigate the drift. In our proposed unsupervised drift detection method, we follow a two step process. Our first step involves encoding a sample of production data as the target distribution, and the model training data as the reference distribution. In the second step, we employ a kernel-based statistical test that utilizes the maximum mean discrepancy (MMD) distance metric to compare the reference and target distributions and estimate any potential drift. Our method also identifies the subset of production data that is the root cause of the drift. The models retrained using these identified high drift samples show improved performance on online customer experience quality metrics.

 $\textit{Keywords}\ \mathrm{Drift}\ \mathrm{detection}\ \cdot \ \mathrm{unsupervised}\ \mathrm{method}\ \cdot \ \mathrm{machine}\ \mathrm{learning}\ \cdot \ \mathrm{MMD}\ \mathrm{distance}\ \mathrm{metric}\ \cdot \ \mathrm{performance}\ \mathrm{improvement}.$

1 Introduction

In the fast-paced world of big data, where the amount of information being generated is constantly growing, efficient data analytics and machine learning techniques are essential to help us make informed decisions. However, with the rapid emergence of new products, markets, and customer behaviors, a new challenge arises - the problem of drift in data. This occurs when the statistical properties of the data being used change over time in unforeseen ways. If left unchecked, data drift can render past data irrelevant, leading to poor decision outcomes. As a result, drift has become a significant obstacle for many data-driven and machine learning systems. In this dynamic and ever-evolving environment, finding ways to provide reliable and accurate data-driven predictions and decision-making capabilities is crucial [Lu et al., 2018, Gemaque et al., 2020].

Drift detection is the process of monitoring the performance of a machine learning model over time and detecting when the model's behavior deviates from its original distribution. There are two general methods for drift detection: supervised and unsupervised [Gemaque et al., 2020]. Supervised drift detection involves using labeled data to detect changes in the model's performance. This method assumes that there is a labeled dataset available that represents the original distribution of the data. The labeled data is used to train the model, and the performance of the model is

measured on new labeled data as it becomes available. If the performance of the model deviates significantly from its original performance, the model is flagged for drift [Lu et al., 2018, Maciel et al., 2015, Gonçalves Jr et al., 2014, Iwashita and Papa, 2018]. On the other hand, unsupervised drift detection involves monitoring the model's behavior without the use of labeled data. This method assumes that there is no labeled dataset available that represents the original distribution of the data. Instead, the model's behavior is monitored using statistical techniques such as change point detection or anomaly detection also distance between two distributions. These techniques compare the model's behavior over time and look for significant deviations that could indicate drift [Friedrich et al., 2023, de Mello et al., 2019, Gemaque et al., 2020].

Despite supervised drift detection being accurate, it is not often used in practice compared to unsupervised drift detection methods because if labeled data is not available, the method cannot be used. Additionally, labeled data can be expensive and time-consuming to obtain, which can make supervised drift detection impractical in some situations [Gemaque et al., 2020]. In the literature, many studies have used these approaches to detect drift. Harel et al. [2014] proposed a supervised method for identifying drift in data streams through the examination of empirical loss of learning algorithms. This approach involves extracting statistics from the distribution of loss by using resampling to repeatedly reuse the same data. Maciel et al. [2015] proposed a new supervised ensemble classifier called drift detection ensemble (DDE), which is designed to improve the effectiveness of three different concept drift detectors. DDE combines the warnings and drift detections from these three detectors using various strategies and configurations to achieve better performance than the individual methods alone. Li et al. [2019] proposed an unsupervised approach for detecting anomalies in multidimensional sequences in data streams that are susceptible to concept drift, by utilizing a feature selection algorithm based on mutual information and symmetric uncertainty information. The proposed approach, called FAAD, consists of three different algorithms and aims to analyze the ordering relationship in sequences to detect anomalies. Costa et al. [2018] used an unsupervised drift detection approach, DDAL, that utilizes active learning to select the most significant instances for monitoring concept drift based on density variation. The proposed method involves two phases, where the first phase generates a classifier using reference data and the second phase involves drift detection, reaction, and classification for new batches of data based on the occurrence of concept drift. Lughofer et al. [2016] introduces two techniques to handle concept drift in scenarios with few labeled instances or no labeled instances, one based on active learning filters and the other on analyzing classifier output certainty distribution. Haque et al. [2016] proposed a semi-supervised framework for detecting drift and novel classes in data streams using an ensemble of kNN classifiers, with outlier detection and concept change detection modules, and incorporating data self-annotation.

In this paper, we present a novel unsupervised approach for detecting drift in unstructured text data used in machine learning models, without requiring human annotation. Our proposed method offers several key novelties which are as follows:

- 1. Our proposed algorithm is a highly versatile, unsupervised drift detection method that can be applied to any machine learning model for both performance regression detection and mitigation in unstructured text data without the need for human input.
- 2. Our method includes a mitigation strategy for addressing model performance regression. It offers a fast and reliable solution for improving model performance when high levels of drift are present in production data.
- 3. We demonstrate the effectiveness of our approach in detecting and mitigating performance regression in a real-world application. By leveraging our novel method, we were able to achieve improved model performance, highlighting the practical value of our approach.

2 Methodology

In this paper, we introduce an unsupervised drift detection method for unstructured data, specifically text data utilized as input for machine learning models. Our proposed method involves the encoding of unstructured text data into vector representation, followed by a comparison of the encoded data to identify potential drift. To achieve this, we use a maximum mean discrepancy (MMD) distance [Gretton et al., 2012] in a kernel-based statistical test, leveraging bootstrap to provide statistics that characterize the drift, such as the median and mean. Additionally, our approach identifies the subset of data likely to be the root cause of the drift. Subsequently, this subset of data can be used in the retraining of models to minimize performance regression.

Specifically, we show how our proposed drift detection method can be used for detecting drift in the inference data for a machine learning model that is in production. Leveraging our proposed method, we can systematically compare the model development data, including training and validation data, with the production data to identify potential drift. Although we demonstrate the proposed method for text data in this paper, it can be used for other data modalities, such as image and voice data.

Our method compares the input data of a machine learning model trained on (X_{tr}, Y_{tr}) with that of X_{prod} , a subset of production data. However, since both the training and production data can be large, we divide them into mini-batches and compare those mini-batches to identify potential drift. To mitigate sampling bias, we first shuffle both the training and production data before dividing them into mini-batches. Given that our proposed method is designed to detect drift in text data, we leverage an encoder such as BERT [Devlin et al., 2018] to obtain fixed-sized embeddings of the text data. Specifically, we compute the average embedding for a mini-batch as its overall vector representation. Subsequently, we use these embeddings as input in our proposed drift detection algorithm, which is presented in Algorithm 2.

We used the maximum mean discrepancy (MMD) test [Gretton et al., 2012] in our proposed method to compare the training (reference) and production (target) distributions. MMD is a distribution-free approach that is particularly well-suited for high-dimensional spaces. It measures the difference between two probability distributions by evaluating the distance between their means in a reproducing kernel Hilbert space (RKHS). MMD's strength lies in its ability to capture complex, nonlinear relationships between variables and its ability to handle high-dimensional data. Therefore, we selected MMD as it can provide an accurate representation of the dissimilarity between the reference and target distributions.

The proposed method utilizes bootstrap to estimate the drift distribution by comparing the training and production distributions. Initially, both distributions are combined under the null hypothesis of no drift. Subsequently, a bootstrapped distribution is obtained by randomly sampling with replacement. Next, the method calculates the drift between the first and second halves of the bootstrapped samples. If the null hypothesis holds, and there is no difference between the distributions, then the amount of drift (measured by MMD) in the bootstrapped distributions should be minimal. Otherwise, a significant drift is observed. Furthermore, the likelihood of the drift can be estimated using bootstrap.

To mitigate the performance regression caused by drift, we propose a process in which we examine the estimated drift between mini-batches from our proposed algorithm. This enables us to identify the samples that exhibit the greatest deviation from the production data, thereby isolating the most significant sources of drift. These samples can then be reintroduced into the training data, allowing the model to be refreshed and better equipped to handle future changes in the data distribution. This approach not only improves the model's overall performance, but also ensures its continued relevance and accuracy in dynamic, real-world settings.

3 Experiment and Results

This section showcases the effectiveness of our proposed drift detection methodology in three areas: (1) detecting model performance regression, (2) implementing mitigation strategies for model performance regression, and (3) text encoder effect ablation study. The overall approach that we present here can be applied to any domain or dataset.

3.1 Model Performance Regression Detection

To demonstrate a significant negative correlation between model performance metrics and estimated drift from our proposed drift detection method, we conducted an experiment based on a binary classification model, where the model is designed to detect if a text sentence is related to shopping in the categories such as searching, buying, and checking price for items. For this binary classification model, we use BERT [Devlin et al., 2018] encoder to obtain the embedding of a sentence and then the embedding goes through three feed forward dense layers for binary classification task. The model was trained on a dataset comprising approximately 800K annotated samples and evaluated on a separate test set of around 150K data points.

After training and tuning the model on validation data, we evaluated its performance on test data. Then, we conducted an experiment to examine the relationship between its performance metrics and drift estimated by our proposed drift detection method. For this experiment, we collected data for about 3 years and divided it into monthly buckets. We then calculated the amount of drift between the model's development data and the data from each monthly bucket, and also computed the model's area under the ROC curve (AUC) and binary cross entropy (BCE) for each bucket as performance metrics. In our drift detection method, we used the BERT encoder for getting the embedding of text data [Devlin et al., 2018]. The results of this experiment are presented in Figure 1, which clearly shows that as the amount of drift increases, the model's performance significantly decreases.

Metric	correlation (%)
MMD vs BCE	76.9
MMD vs AUC	-65.2

Table 1: Correlation coefficient between estimated drift (MMD) and model performance metrics.

Algorithm 1 Proposed Drift Detection Method

```
1: Result: Estimated Drift, Samples Causing Drift
 2: Inputs:
 3: D_1: Reference distribution (e.g. set of embeddings)
 4: D_2: Target distribution (e.g. set of embeddings)
 5: Variable:
 6: \beta: Number of samples to be compared
 7: M: Total number of samples
 8: K : Number of bootstraps
 9: D_{1,j}:jth sample of reference distribution
10: D_{2,j}: jth sample of target distribution
11: R: List of MMD statistics
12: R_b: List of bootstraped MMD statistics
13: X : List of Median MMD statistics
14: t : Index
15: for t = \beta : M do
         Q_1 = \{D_{1,1}, D_{1,2}, ..., D_{1,t}\} Q_2 = \{D_{2,1}, D_{2,2}, ..., D_{2,t}\}
                                                                                                        ⊳ batch of reference distribution
16:

    batch of target distribution

17:
         r = MMD(Q_1, Q_2)
                                                                                              18:
         T = \{Q_1, Q_2\}
19:
                                                                                        \triangleright combine Q_1 and Q_2 (under H_0 of no drift)
         R = \{R; r\}
20:
                                                                                                                  ⊳ append MMD statistic
21:
22:
         for i = 1 : K \operatorname{do}
              T' = Bootstrap(T)
23:
                                                                                                       \triangleright get a bootstrap samples from T
              Q_{1}' = T_{1:\beta/2}'
                                                                                                                        \triangleright get 1st half of T'
24:
              Q_{2}^{'} = T_{\beta/2+1:\beta}^{'}
r^{'} = MMD(Q_{1}^{'}, Q_{2}^{'})
                                                                                                                        \triangleright get 2nd half of T'
25:
26:
              R_{b} = \{R_{b}; r'\}
27:

    □ append MMD statistic

28:
          end for
         d_{med} = Median(R_b)
29:
                                                                                                > compute median of estimated MMDs
          X = \{X; d_{med}\}
30:
                                                                                                                       \triangleright append d_{med} to X
         t+=1

    increase index

31:
         Q_1 = \{Q_1; D_{1,t}\}\

Q_2 = \{Q_2; D_{2,t}\}\

Q_1.popleft()
                                                                                                                  \triangleright add next sample to Q_1
32:
33:
                                                                                                                  \triangleright add next sample to Q_2
34:
                                                                                                          \triangleright remove first sample from Q_1
35:
         Q_2.popleft()
                                                                                                          \triangleright remove first sample from Q_2
36: end for
37: z=Argmax(R)

    b highest drift Index

38: U = \{D_{l,z-\beta}, D_{l,z-\beta+1}, ..., D_{l,z}\} l \in \{1,2\} 39: Return d = mean(X)
                                                                                                                            ⊳ median MMD
40: Return U
                                                                                                          > samples causing highest drift
```

The results in Table 1 demonstrate a significant correlation between estimated drift (MMD) and model performance metrics. Consequently, our unsupervised drift detection approach can effectively monitor model performance and accurately predict model regression. This capability is particularly valuable in production environments where annotated data may not be available, making our drift detection method the only practical option for model performance monitoring. Figure 2 depicts the estimated drift and model performance over time, revealing a notable increase in drift as time progresses. This increase is attributed to the emergence of new data patterns in the production data, causing an increase in the estimated drift and a corresponding drop in the model's performance. These findings underscore the need for ongoing monitoring and recalibration of the model to ensure its continued effectiveness.

3.2 Model Performance Regression Mitigation

To assess the effectiveness of our proposed drift detection method in reducing the decline in model performance from drift when used in production, we outline a comprehensive mitigation process. Firstly, we run our drift detection method by setting the reference and target distributions to be the model's training and inference data during production,

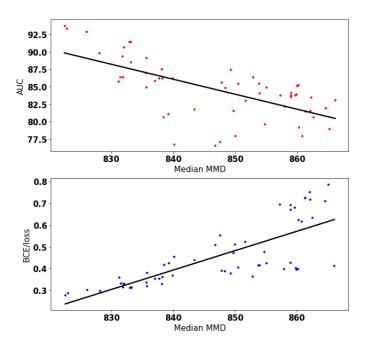


Figure 1: Plot of estimated drift (MMD) vs the model performance metrics for monthly buckets, which indicates that as the amount of drift increases, model performance drops.

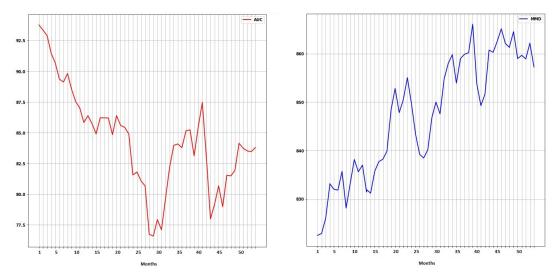


Figure 2: Left and right plots shows the model performance metric (AUC) and the estimated amount of drift (MMD), respectively. The increase in MMD over time is attributed to the emergence of new data patterns in the production data, causing an increase in the estimated drift and a corresponding drop in the model's performance

respectively. Then, our proposed method identifies the samples that exhibit the highest degree of drift or deviation from the production data. Finally, we incorporate the dataset generated from the previous step into the model's training data and retrain the model, enabling it to improve its performance during production.

We implemented the above-mentioned approach on a multi-task model used for domain classification, intent classification, and name entity recognition. The model architecture comprises a blend of Bi-LSTM and transformer models [Devlin et al., 2018].

To improve this model performance, we employed our proposed drift detection method to compare the model's training data with one month of production data. This enabled us to identify samples with the highest degree of drift from production data. However, in order to use these data during model training, they need to be annotated. Therefore, we

utilized a semi-supervised approach, where we used a large transformer model that leverages a variety of offline signals such as previous conversations that are not available during online inference, to generate pseudo-labels for these data. With the help of these pseudo-labels, we added new samples with highest amount of drift to the model's training data and retrained the model.

We conducted a thorough evaluation of the new model trained on samples generated from our drift detection method. Specifically, we tested the model's performance on a held-out false accept (FA) dataset, consisting of only false accept data, to evaluate its ability to reject false accept samples. Additionally, we compared the performance of our proposed approach with several common mitigation methods. These methods include the following:

Baseline: A baseline model trained using the model's original training data without any mitigation.

Bias reduction: This method involves clustering the training and production data to identify patterns that are absent within the training data. We then added new samples from absent patterns to the training data using this approach.

Upsampling: This method first identifies the underrepresented samples within the training data by clustering and then employs upsampling to increase their frequency in the training data. Then, we added underrepresented samples to the training data using this approach.

We re-trained the model using all these methods, and Table 2 provides a detailed comparison of the various methods used in this experiment. The results indicate that the samples of data added by our proposed drift detection method significantly improve model performance and outperform other methods to varying degrees, without an increase in false reject rate.

Method	FAR (%)
Baseline	73.15
Bias reduction	61.85
Upsampling	71.48
Proposed drift detection	59.68

Table 2: Comparison of different mitigation methods on false accept rate (FAR) test set. Our proposed drift detection method shows superior performance compared to the other methods. Lower FAR values indicate better performance.

3.3 Encoder Effect Ablation Study

In this section, we outline an experiment designed to assess the effect of different encoders on the performance of our proposed method. We utilize different encoders to extract text embeddings and focus on simulating data drift within a binary classification scenario. The reference dataset initially maintains a balanced distribution of positive and negative class instances, each accounting for 50% of the dataset. Subsequently, we change the positive class percentage within the target dataset to induce data drift. Following datasets are used in our experiment:

AG news: AG news comprises over one million news articles, compiled from a diverse range of 2000+ news sources over a span of more than a year [Zhang et al., 2015]. We use world and sports classes in our experiment.

Yelp review: yelp review dataset consists of reviews extracted from the Yelp [Zhang et al., 2016]. We use reviews rated 5 as the positive class and reviews rated 1 as the negative class in our experiment.

We sample 5000 observations from each dataset where we increase/decrease the percentage of positive class in the target dataset while comparing it to the reference dataset with equal ratio of positive and negative classes. We set the batch size, β , and bootstrap steps of our method to be 64, 32, and 50, respectively. We use the following encoders in our experiment:

bert-base-uncased: BERT base [Devlin et al., 2018] is a transformer encoder model with 12 layers with 768 hidden size that is pretrained on a large corpus of English data.

bert-large-uncased: this is a BERT model with 24 layers with 1024 hidden size.

all-MiniLM-L12-v2: This model, known as a sentence-transformers model, effectively encodes sentences and paragraphs into a 384-dimensional dense vector space, offering utility in applications such as clustering or semantic search [SBERT, 2023]. This model has 12 layers with 384 hidden size.

all-distilroberta-v1: This model is also a sentence-transformer with 12 layers and 768 hidden size.

We ran our drift detection method using all four different encoders on both datasets and the following Figure 3 shows the results. The results indicate that there is a small amount of drift when the percentage of positive class is close to

50% which is the exact distribution of the reference dataset. However, the amount of drift increases as the percentage of positive class gets more different than the reference distribution. All encoders performed comparably but the sentence BERT and larger transformer models tend to detect drift faster due to producing higher quality text representations.

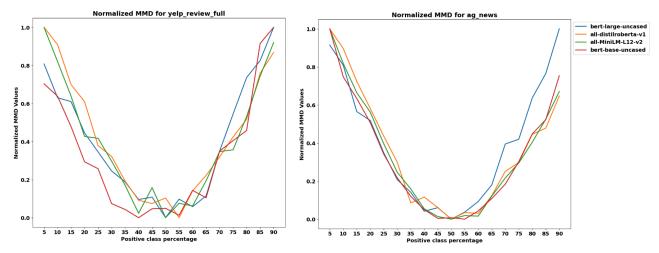


Figure 3: Left and right plots shows MMD values as we change the percentage of positive class in the target data resulting in data drift.

4 Discussion and Conclusion

Our proposed unsupervised drift detection approach for unstructured data, specifically text data, presents a novel and effective method for identifying potential drift in machine learning models. By utilizing maximum mean discrepancy (MMD) kernel-based statistical test and bootstrap, we can accurately identify the subset of data likely to be the root cause of the drift and suggest its adding back to the training data as a mitigation strategy. Our approach is particularly useful in identifying model performance regression and provide a fast mitigation solution for recovery.

In this paper, we demonstrated the efficacy of our proposed method for two use cases, but the approach can be applied to any other domain or dataset. Overall, our proposed method provides a robust framework for detecting and mitigating performance regression in machine learning models caused by drift.

References

Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

Rosana Noronha Gemaque, Albert França Josuá Costa, Rafael Giusti, and Eulanda Miranda Dos Santos. An overview of unsupervised drift detection methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(6):e1381, 2020.

Bruno Iran Ferreira Maciel, Silas Garrido Teixeira Carvalho Santos, and Roberto Souto Maior Barros. A lightweight concept drift detection ensemble. In 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI), pages 1061–1068. IEEE, 2015.

Paulo M Gonçalves Jr, Silas GT de Carvalho Santos, Roberto SM Barros, and Davi CL Vieira. A comparative study on concept drift detectors. *Expert Systems with Applications*, 41(18):8144–8156, 2014.

Adriana Sayuri Iwashita and Joao Paulo Papa. An overview on concept drift learning. IEEE access, 7:1532–1547, 2018.

Björn Friedrich, Taishi Sawabe, and Andreas Hein. Unsupervised statistical concept drift detection for behaviour abnormality detection. *Applied Intelligence*, 53(3):2527–2537, 2023.

Rodrigo F de Mello, Yule Vaz, Carlos H Grossi, and Albert Bifet. On learning guarantees to unsupervised concept drift detection on data streams. *Expert Systems with Applications*, 117:90–102, 2019.

Maayan Harel, Shie Mannor, Ran El-Yaniv, and Koby Crammer. Concept drift detection through resampling. In *International conference on machine learning*, pages 1009–1017. PMLR, 2014.

- Bin Li, Yi-jie Wang, Dong-sheng Yang, Yong-mou Li, and Xing-kong Ma. Faad: an unsupervised fast and accurate anomaly detection method for a multi-dimensional sequence over data stream. *Frontiers of Information Technology & Electronic Engineering*, 20(3):388–404, 2019.
- Albert França Josuá Costa, Régis Ant&nio Saraiva Albuquerque, and Eulanda Miranda Dos Santos. A drift detection method based on active learning. In 2018 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2018.
- Edwin Lughofer, Eva Weigl, Wolfgang Heidl, Christian Eitzinger, and Thomas Radauer. Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances. *Information Sciences*, 355: 127–151, 2016.
- Ahsanul Haque, Latifur Khan, and Michael Baron. Sand: Semi-supervised adaptive novel class detection and classification over data stream. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.
- Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* preprint arXiv:1810.04805, 2018.
- Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2016. SBERT. Sbert pretrained models, 2023.