Data Drift in Machine Learning

Introduction

In this blog post we try to learn about the different types of data-drifts and learn about the few methods of detecting data drifts.

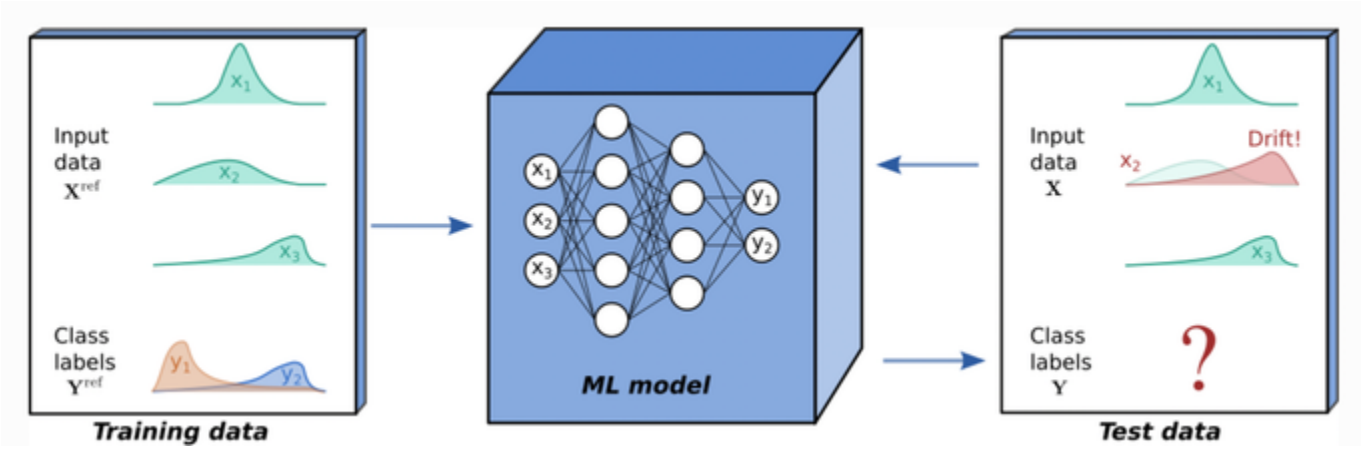
We might hope that when faced with unexpected inputs, well-designed software systems would fire off warnings. Machine learning (ML) systems, however, which depend strongly on properties of their inputs (e.g. the i.i.d. assumption), tend to fail silently.

When a machine learning model is deployed in production, the main concern of data scientists is the model pertinence over time. Is the model still capturing the pattern of new incoming data, and is it still performing as well as during its design phase ? There's the constant fear of model's performance deteriorating over time as the model goes stale.

What is Data-Drift?

Although powerful, modern machine learning models can be sensitive. Seemingly subtle changes in a data distribution can destroy the performance of otherwise state-of-the art models, which can be especially problematic when ML models are deployed in production. Typically, ML models are tested on held out data in order to estimate their future performance. Crucially, this assumes that the *process* underlying the input data X and output data Y remains constant.

Drift is said to occur when the process underlying X and Y at test time differs from the process that generated the training data. In this case, we can no longer expect the model's performance on test data to match that observed on held out training data. At test time we always observe features X, and the *ground truth* then refers to a corresponding label Y. If ground truths are available at test time, *supervised drift detection* can be performed, with the model's predictive performance monitored directly. However, in many scenarios, such as the binary classification example below, ground truths are not available and *unsupervised drift detection* methods are required.



To explore the different types of drift, consider the common scenario where we deploy a model f:xy on input data X and output data Y, jointly distributed according to P(X,Y). The model is trained on training data drawn from a distribution Pref(X,Y). *Drift* is said to have occurred when P(X, Y)Pref(X,Y).

P_ref(X,Y) is the reference dataset that you have and want to compare the new data to it.

Writing the joint distribution as

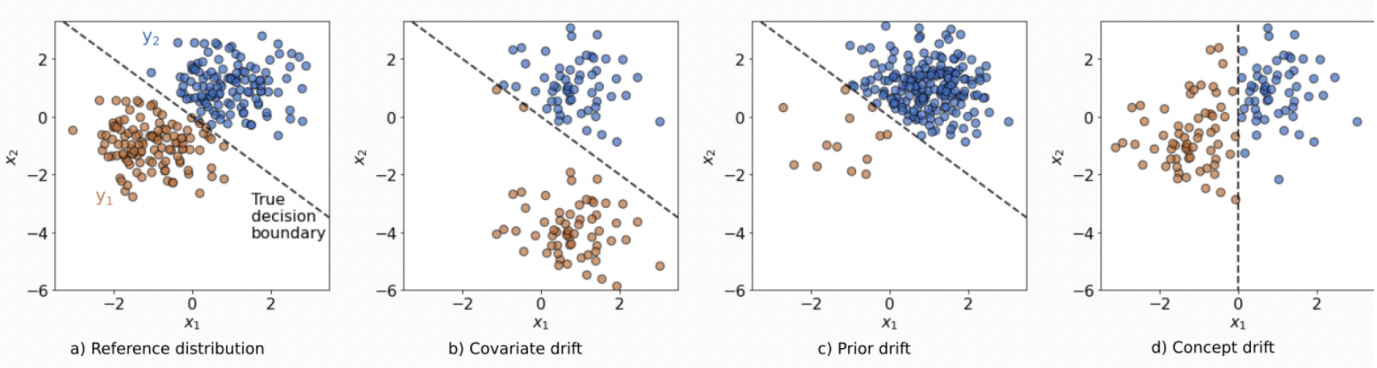
$$P(X,Y)=P(Y|X)P(X)=P(X|Y)P(Y),$$

we can classify drift under a number of types:

- **Covariate drift**:
  - Also referred to as input drift, this occurs when the distribution of the input data has shifted P(X)Pref(X), whilst P(Y|X) = Pref (Y|X).
  - This may result in the model giving unreliable predictions.
  - An example could be regulatory intervention, where new laws would shake up the market landscape, and consumer behavior would follow different behaviors from before.
- **Prior drift**:
  - Also referred to as label drift, this occurs when the distribution of the target variable has shifted P(Y)Pref(Y), whilst P(X|Y)=Pref (X|Y).
  - This can affect the model's decision boundary, as well as the model's performance metrics.
  - An example could be that in sentient vegetation failure detections, the failure to non-failure ratio changes. One of the reasons for drift could be seasonality - more vegetation in fall season.
- **Concept drift**:
  - This occurs when the process generating y from x has changed, such that P(Y|X)Pref(Y|X).
  - It is possible that the model might no longer give a suitable approximation of the true process. The concept term refers to is the relationship between independent and dependent variables.
  - For instance, if we wanted to predict the NOX emissions and if our model is only trained on historic data, it will be useless if very strict regulatory laws are introduced.

Note that a change in one of the conditional probabilities P(X|Y) and P(Y|X) does not necessarily imply a change in the other. For example, consider the pneumonia prediction example from Lipton et al., whereby a classification model f is trained to predict y, the occurrence (or not) of pneumonia, given a list of symptoms x. During a pneumonia outbreak, P(Y|X) (e.g. pneumonia given cough) might rise, but the manifestations of the disease P(X|Y) might not change. In many cases, knowledge of underlying causal structure of the problem can be used to deduce that one of the conditionals will remain unchanged.

Below, the different types of drift are visualized for a simple two-dimensional classification problem. It is possible for a drift to fall under more than one category, for example the *prior drift* below also happens to be a case of *covariate drift*.



It is relatively easy to spot drift by eyeballing these figures here. However, the task becomes considerably harder for high-dimensional real problems, especially since real-time ground truths are not typically available. Some types of drift, such as prior and concept drift, are especially difficult to detect without access to ground truths. As a workaround proxies are required, for example a model's predictions can be monitored to check for prior drift.

To summarize:

Covariate Drift	Change in features P(x)
Label Drift/Prior Probability Drift	Change in target variable P(y)
Concept Drift	Changes in the relationship between independent and dependent variables P(y|X)

Drift Detection Methods

Monitoring model performance drift is a crucial step in production ML; however, in practice, it proves challenging for many reasons, one of which is the delay in retrieving the labels of new data. Without ground truth labels, drift detection techniques based on the model's accuracy are off the table.

However, if getting ground truth labels is not an issue then we can use drift detection methods that monitor the model performance usually through some performance metric. The constraint that the class labels are available right after prediction is unrealistic for our scenario hence, we will be based on the feature distribution.

If you're interested to learn more about drift detection algorithms focused on time series data, a good place to start would be:

> Machine Learning for Time-Series with Python: Forecast, predict, and detect anomalies with state-of-the-art machine learning methods: Ben Auffarth (Chapter 8)

Most of the work here is inspired from the paper:

Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift

Formal Definition of the problem:

Given labeled data {(x1, y1), ...,(xn, yn)} p and unlabeled data {x1' ,x2',..., xm'} q, our task is to determine whether p(x) equals q(x'). Formally, **H0 : p(x) = q(x') vs HA : p(x) != q(x')**

Due to natural randomness in the process being modelled, we don't necessarily expect observations z1,…,zN drawn from P(z) to be identical to z1ref,…,zMref drawn from Pref(z).

To decide whether differences between P(z) and Pref(z) are due to drift or just natural randomness in the data, *statistical two-sample hypothesis* testing is used, with the null hypothesis P(z)=Pref(z).

If the p-value obtained is below a given threshold, the null is rejected and the alternative hypothesis P(z)Pref(z) is accepted, suggesting drift is occurring.

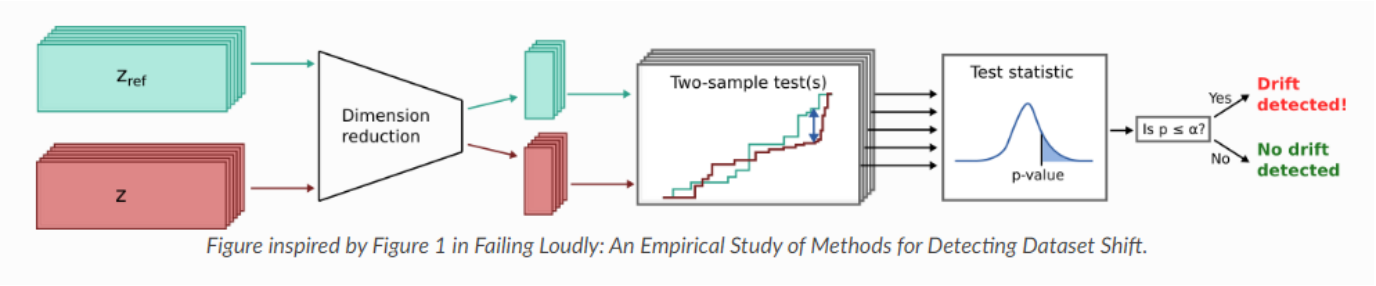


Figure inspired by Figure 1 in Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift.

Chiefly, we explore the following design considerations:

(i) what representation to run the test on;

(ii) which two-sample test to run;

(iii) when the representation is multidimensional; whether to run multivariate or multiple univariate two-sample tests; and

(iv) how to combine their results.

1 - Dimensionality Reduction:

- Motivation : Given an input dataset XR^(N×d), where N is the number of observations and d the number of dimensions, the aim is to reduce the data dimensionality from d to K, where Kd. A drift detector can then be applied to the lower dimensional data X^R^(N×K), where distances more meaningfully capture notions of similarity/dissimilarity between instances.
    - Dimension reduction approaches can be broadly categorized under:
        - Linear projections (PCA)
        - Non-linear projections (Autoencoders, KPCA, tsne)
        - Feature maps (from ML model)

2 - Statistical Hypothesis Testing:

The dimensionality reduction techniques each yield a representation, either uni/multi-dimensional, and either continuous or discrete, depending on the method. The next step is to choose a suitable statistical hypothesis test for each of these representations.

Hypothesis testing involves first choosing a *test statistic* S(z), which is expected to be small if the null hypothesis H0 is true, and large if the alternative Ha is true. For observed data z, S(z) is computed, followed by a p-value p^=P(such an extreme S(z)|H0). In other words, p^ represents the probability of such an extreme value of S(z) occurring given that H0 is true. When p^, results are said to be *statistically significant*, and the null P(z)=Pref(z) is rejected. Conveniently, the threshold represents the desired false positive rate.

The *test statistics* available in Alibi Detect can be broadly split into two categories; univariate and multivariate tests:

- Univariate:
    - Chi-Squared (for categorical data)
    - Kolmogorov-Smirnov
    - Cramér-von Mises
    - Fisher's Exact Test (for binary data)
- Multivariate:
    - Least-Squares Density Difference (LSDD)
    - Maximum Mean Discrepancy (MMD)

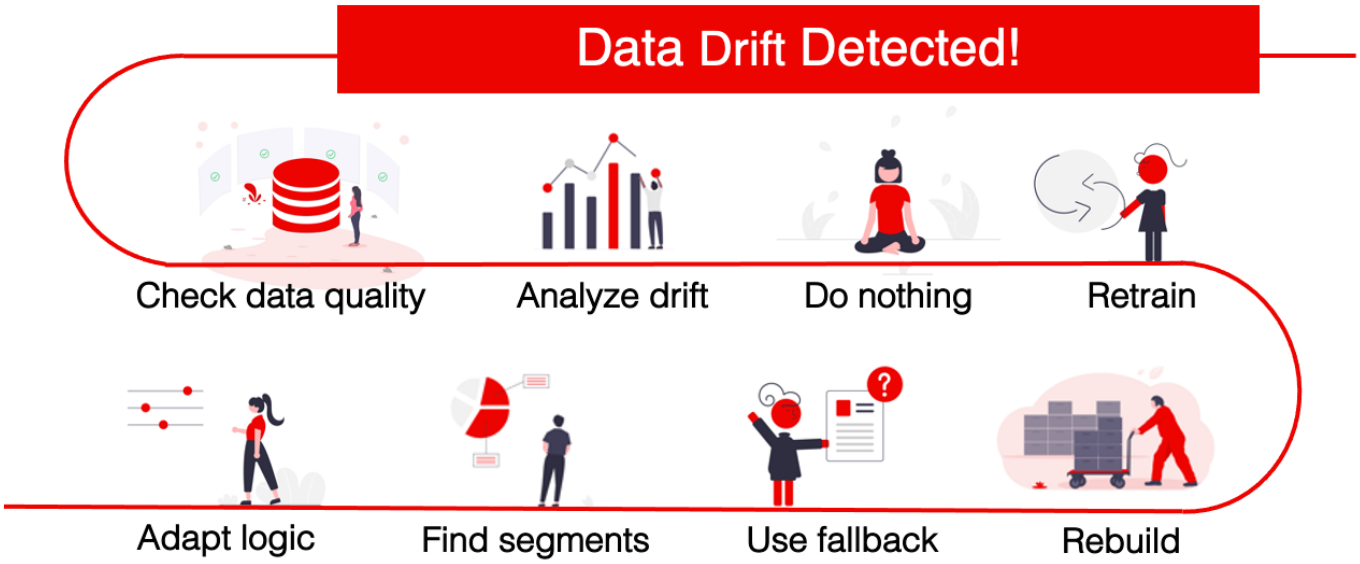
When applied to multidimensional data with dimension d, the univariate tests are applied in a feature-wise manner. The obtained p-values for each feature are aggregated either via the Bonferroni or the False Discovery Rate (FDR) correction. The Bonferroni correction is more

conservative and controls for the probability of at least one false positive. The FDR correction on the other hand allows for an expected fraction of false positives to occur. If the tests (i.e. each feature dimension) are independent, these corrections preserve the desired false positive rate (FPR). However, usually this is not the case, resulting in FPR's up to d-times lower than desired, which becomes especially problematic when d is large. Additionally, since the univariate tests examine the feature-wise marginal distributions, they may miss drift in cases where the joint distribution over all d features has changed, but the marginals have not. The multivariate tests avoid these problems, at the cost of greater complexity.

3 - Example Notebook:

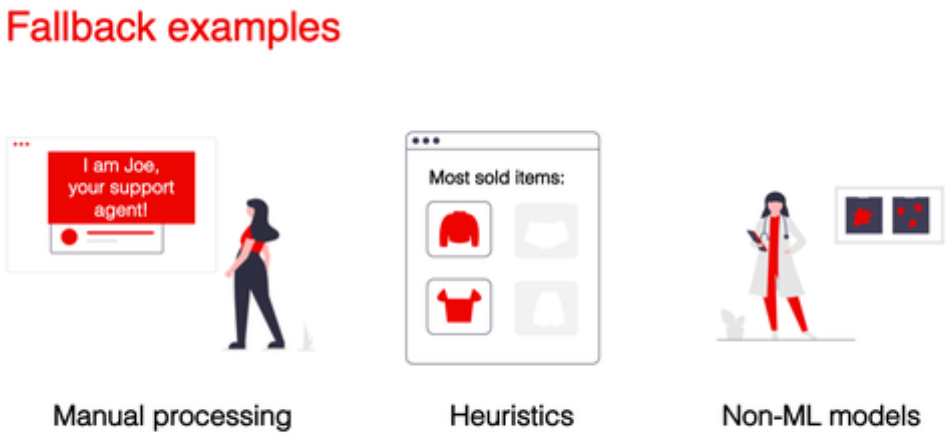
Categorical and mixed type data drift detection on income prediction — alibi-detect 0.10.0 documentation (seldon.io)

Detected Data Drift; What are the next steps?

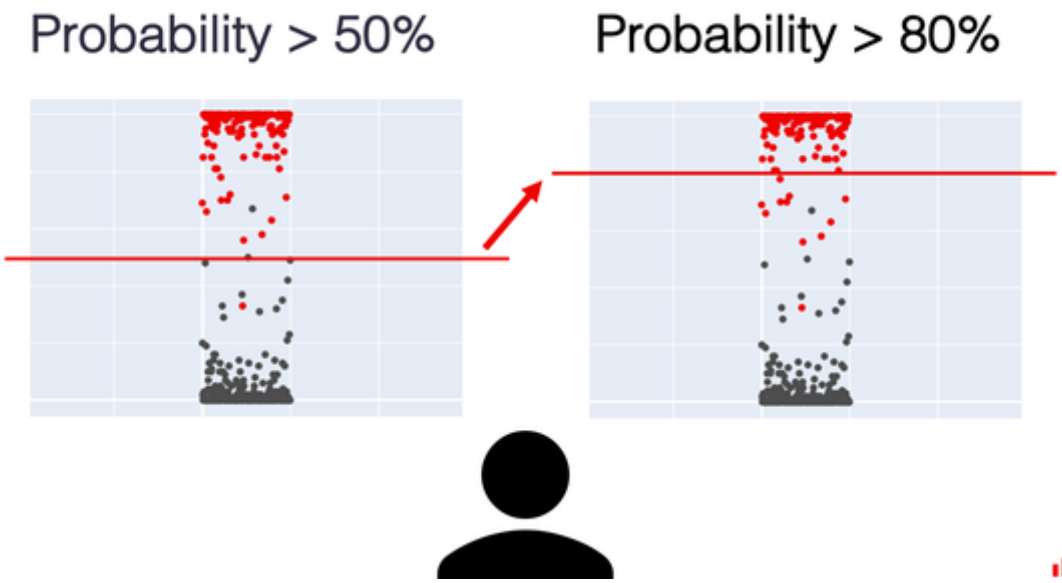


- **Check the data quality.** Make sure the drift is real.
- **Investigate.** Explore the data changes to understand what might be causing them.
- **Do nothing.** You might treat this as a false alarm, be satisfied with how the model handles drift, or simply decide to wait.

- **Retrain it, if you can.** Get new labels and actual values and re-fit the same model on the latest data. Drop the old as needed.
- **Rebuild it, if you need.** If the change is significant, you might need to rebuild the training pipeline and test new model architectures.
- **Use a fallback strategy.** Make a decision without machine learning.



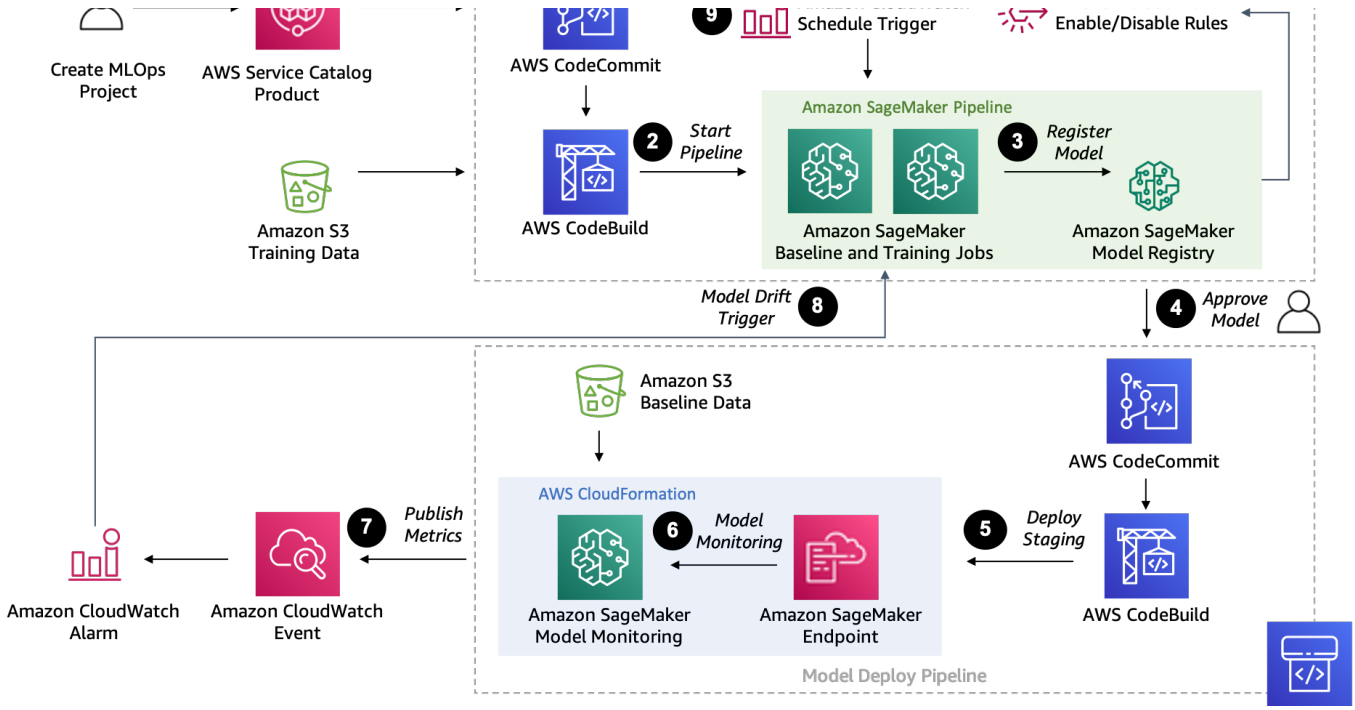
- **Limit the model use.** Find and exclude the segments of low performance.
- **Add custom processing logic.** Add corrective coefficients, change decision thresholds, review outliers. Use with great caution.



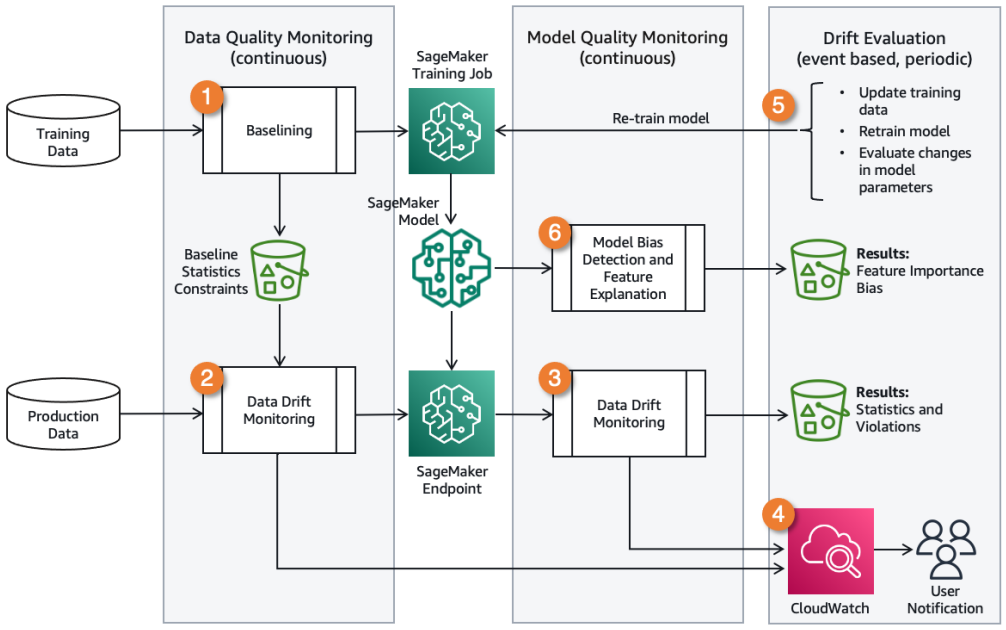
- **Use Adaptive Learning Methods.** Adaptive learning refers to incremental methods with drift adjustment. This concept refers to updating predictive models online to react to concept drifts. The goal is that by taking drift into account, models can ensure consistency with the current data distribution. For instance, Adaptive Random Forest (ARF), Adaptive XGBoost (AXGB) , Hoeffding Adaptive Tree (HAT)

Future Work

Leverage AWS Sagemaker to detect drift and trigger alarms:



Amazon Sagemaker Drift Detection (awesomeopensource.com)



https://aws.amazon.com/blogs/architecture/detecting-data-drift-using-amazon-sagemaker/

Conclusion

"Trust takes a long time to build and can be lost in minutes.", detecting data drift and taking actions on them takes us one step further in maintaining our customer's trust.

In theory, there is not a definitive correlation between the degradation of model performance and any types of data drift. That being said, in practice, observing data drift is a red flag about the pertinence of the deployed model, and a model reconstruction step is highly recommended.

Data Drift detection is not only essential for better model performance but is also crucial to Interpretable AI.

References

- Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift
- Alibi Detect: Algorithms for outlier, adversarial and drift detection
- Machine Learning for Time-Series with Python by Ben Auffarth
- “My data drifted. What’s next?” How to handle ML model drift in production
- Adaptive Learning from Evolving Data Streams | SpringerLink