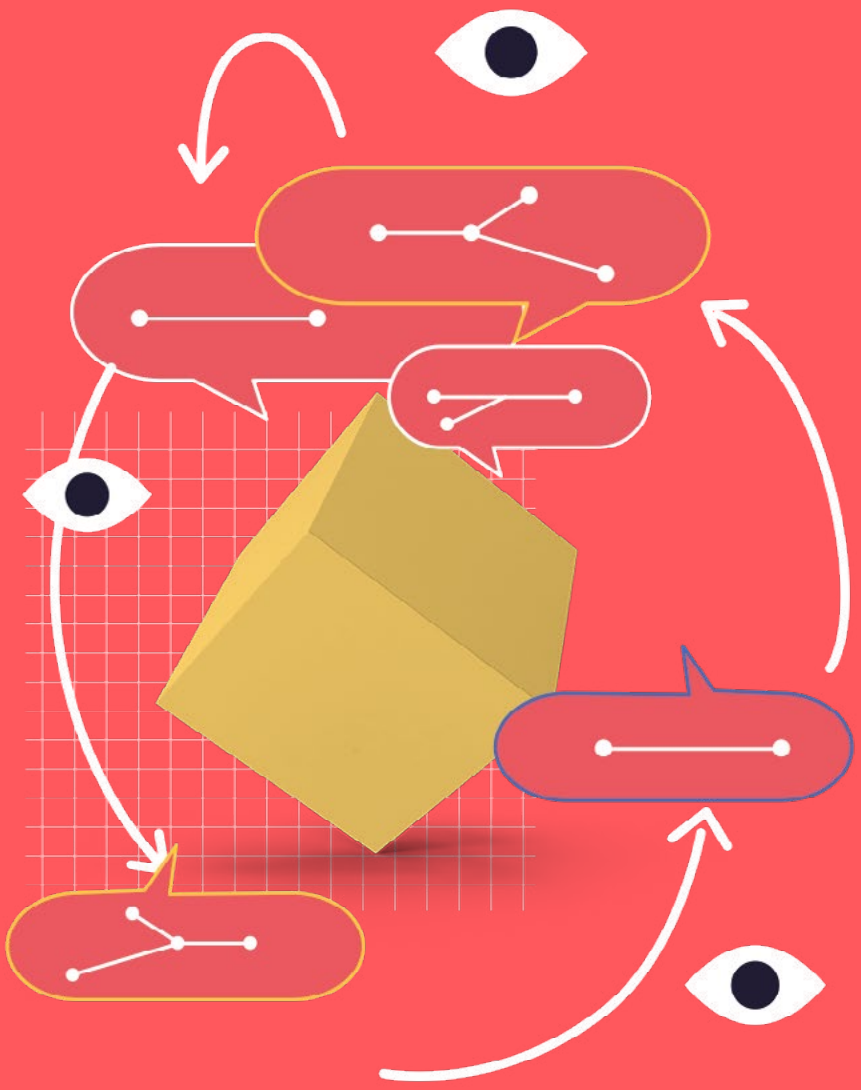
data iku

EBOOK

A Primer on Data Drift & Drift Detection Techniques



About the Authors

After obtaining a Ph. D. in Biomedical Image Processing in 2011, Simona Maggio worked in several companies (CEA, Thales, Rakuten) as a Research Engineer in Computer Vision and Natural Language Processing for applications ranging from video surveillance to document digitization and e-commerce. She's now Senior Research Scientist at Dataiku, exploring MLOps topics, such as model debugging, robustness, and interpretability.

Du Phan is a Machine Learning Engineer at Dataiku, where he works in democratizing data science. In the past few years, he has been dealing with a variety of data problems, from geospatial analysis to deep learning. His work now focuses on different facets and challenges of MLOps.

Introduction

Data is constantly changing, as is the world from which it is collected. When a model is deployed in production, detecting changes and anomalies in new incoming data is key to make sure the predictions obtained are valid and can be safely consumed.

Monitoring model performance drift is a crucial step in production machine learning (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.

This white paper will:

- Take a brief but deep dive into the underlying definitions and logic behind drift.
- Present a simple but real-life use case with different scenarios that can cause models and data to drift.
- Delve into two techniques (the domain classifier and the black-box shift detector) used to assess whether dataset drift is occurring.

Brief Intro to Data Drift
Types and Causes

What is generally called *data drift* is a change in the distribution of data used in a predictive task. The distribution of the data used to train a ML model, called the *source* distribution, is different from the distribution of the new available data, the *target* distribution.

When this change only involves the distribution of the features, it is called *covariate shift*. One simple example of covariate shift is the addition of random noise to some of the features in the dataset.

Another type of data drift is *prior shift*, occurring where only the distribution of the target variable gets modified; for instance changing the proportion of samples from one class. A combination of covariate and prior shifts is also an (unlucky) possibility.

Probably the most challenging type of shift is the *concept shift*, in which case the dependence of the target on the features evolves over time.

An illustration of the possible types of drifts for the electricity dataset[1] is shown in Figure 1. This white paper will touch on all three types of drift, though note that covariate and prior shifts can be addressed without ground truth labels (more on this later).

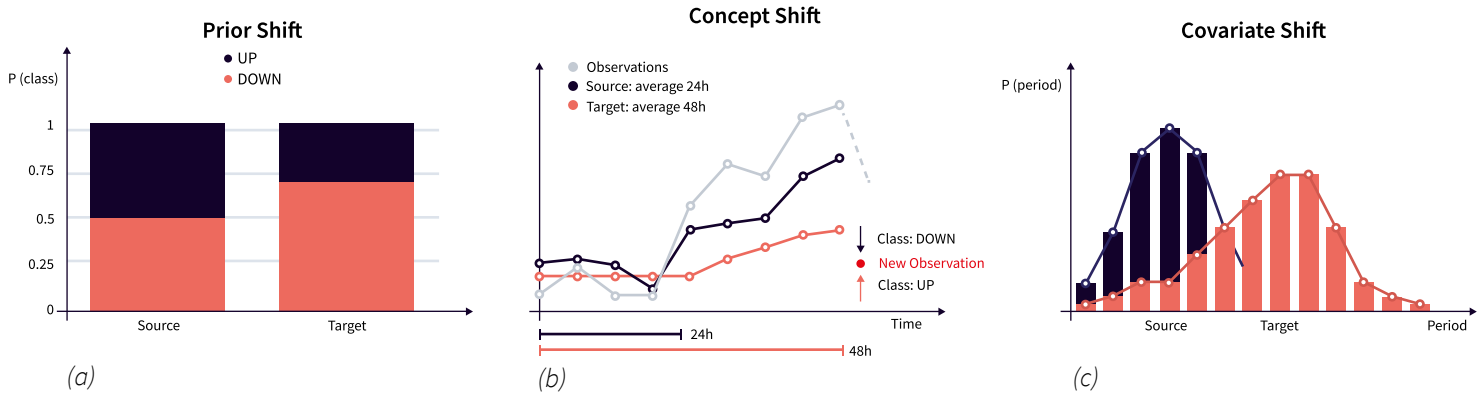


Figure 1: Illustration of prior covariate and concept shift for the electricity dataset. Changing the proportion of samples belonging to the two classes yields to prior shift (a); changing the distribution of the feature "period" yields to covariate shift (c); changing the definition of the class from "relative to average of last 24 hours" to "relative to average of last 48 hours" yields to concept shift (b).

So what causes drift? There are two frequent root causes:

1. **Sample selection bias** (i.e., training sample is not representative of the population). For instance, building a model to assess the effectiveness of a discount program will be biased if the best discounts are proposed to the best clients. Selection bias often stems from the data collection pipeline itself.
2. **Training data collected from source population does not represent the target population.** This often happens for time-dependent tasks — such as forecasting use cases — with strong seasonality effects. Learning a model over a given month won't generalize to another month.

[1] *https://www.openml.org/d/151/*

Going Deeper:
Understanding Drift From a Statistical Learning Perspective

Let's consider a classical supervised learning task with training set $S = \{(x_i, y_i)\}$, with $x_i \in X$ and $y_i \in Y$, and a risk function $r: Y \times Y \rightarrow R$. We wish to learn a predictive function $f : X \rightarrow Y$ that not only minimizes the empirical risk (on the set S) but also the generalization risk (on unseen data).

Under the statistical learning framework, the generalization risk can be faithfully estimated from the empirical risk. This probably approximately correct (PAC) theorem uses Hoeffding's inequality, which relies on the infamous Independent and Identically Distributed (IID) assumption.

Let $\mathcal{H}$ a finite function class. The empirical risk at $f \in \mathcal{H}$ is defined by

$$\widetilde{R}_e(f) = \frac{1}{n}\sum_{i=1}^{n} r(f(x_i), y_i)$$

And the generalization risk is defined by

$$R(f) = \mathbb{E}[r(f(X), Y)]$$

Let $f_a = \text{argmin}_{h \in \mathcal{H}} R(h)$ and $\tilde{f} = \text{argmin}_{h \in \mathcal{H}} \widetilde{R}_e(h)$.

Theorem. *Suppose that* $\forall h \in \mathcal{H},\ R(h) \in [0, 1]$ *and* $|\mathcal{H}| < \infty$. *With probability at least* $1 - \delta$, *we have:*

$$R(\tilde{f}) \leq R(f_a) + 2\sqrt{\frac{\log(|\mathcal{H}|) + \log(2/\delta)}{2n}}$$

Figure 2: Generalization Bounds Theorem.

Under this assumption, each pair of observations (x, y) is drawn independently from the same joint distribution p(X,Y). Intuitively, this means that the training set S is a reasonable protocol through which we can get partial information about the underlying true function f.

If the above assumption does not hold, the empirical risk on the training set is no longer a robust estimation of the expected empirical risk on new unseen data. In other words, the test error is no longer a good estimation of future errors.

In the literature, several terms and formulations are used to describe the problem of data drift, here we employ the framework presented by Quiñonero Candela[2]:

Formally, dataset drift between source distribution S and a target distribution T can be defined as a change in the joint distribution of features and target:

$$P(x_s, y_s) \neq P(x_t, y_t)$$

[2] *Quionero-Candela, Joaquin, et al. Dataset shift in machine learning. The MIT Press, 2009.*

Measuring Data Drift: A Practical Example

Let's say we want to predict the quality of the Bordeaux bottles at the wine shop near our place. This will help us save some money (and avoid some not-worth-it hangovers).

To do this, we will use the UCI Wine Quality dataset[3] as training data, which contains information about red and white variants of the Portuguese "Vinho Verde" along with a quality score varying between 0 and 10.

The following features are provided for each wine: type, fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol rate.

To simplify the modeling problem, let's say that a good wine is one with a quality score equal to or greater than 7. The goal is thus to build a binary model that predicts this label from the wine's attributes.

For purposes of demonstrating data drift, we explicitly split the original dataset into two:

- The first one contains all wines with an alcohol rate above 11%, **wine_alcohol_above_11.**
- The second one contains those with an alcohol rate below 11%, **wine_alcohol_below_11.**

Such clear differences between the two datasets will be formalized as data drift. We split wine_alcohol_above_11 to train and score our model, and the second dataset wine_alcohol_below_11 will be considered as new incoming data that needs to be scored once the model has been deployed.

At this point, you would naturally expect a discrepancy between the scores. Why is that? Because this violates the underlying IID assumption (covered in *Going Deeper: Understanding Drift from a Statistical Learning Perspective*).

To keep the problem simple, we employ a univariate point of view, focusing on the feature **alcohol.** However, the reasoning can be generalized to multiple dimensions, which is usually the case in practice (several features drifting at the same time). The question we want to address here is:

Will the performance of a model trained on my training set (wines with an alcohol level above 11%) change drastically when scoring new bottles (whose alcohol level is below 11%)?

One important detail to note is that the assessment is about the comparative performance of the model between original and new data, and not about the absolute performance of the model.

If we have the ground truth labels of the new data, one straightforward approach is to score the new dataset and then compare the performance metrics between the original training set and the new dataset. However, in real life, acquiring ground truth labels for new datasets is usually delayed. In our case, we would have to buy and drink all the bottles available, which is a tempting choice… but probably not a wise one.

[3] *https://archive.ics.uci.edu/ml/datasets/Wine+Quality*

Therefore, in order to be able to react in a timely manner, we will need to base performance solely on the features of the incoming data. The logic is that if the data distribution diverges between the training phase and testing phase, it is a strong signal that the model's performance won't be the same.

With that being said, the question above can be answered by checking data drift between the original training set and the incoming test set. Let's take a closer look at different situations where data drift occurs. Throughout this section, this is how our model is trained and originally tested:

From the **wine_alcohol_above_11** dataset, we randomly split them into two:

- The first one to train the model, denoted by **alcohol_above_11_train**, will be further split in training and validation sets.
- The other one for testing the model called **alcohol_above_11_test**.

We fit a random forest model on **alcohol_above_11_train**. The model achieves an F1 Score of 0.709 on the hold-out set.

*Figure 3: Results of random forest model on alcohol_above_11_train (done in Dataiku DSS — to follow along with this example, **download Dataiku DSS for free**).*

Situation 1: No Drift

Here the target dataset is the source dataset **wine_alcohol_above_11:**

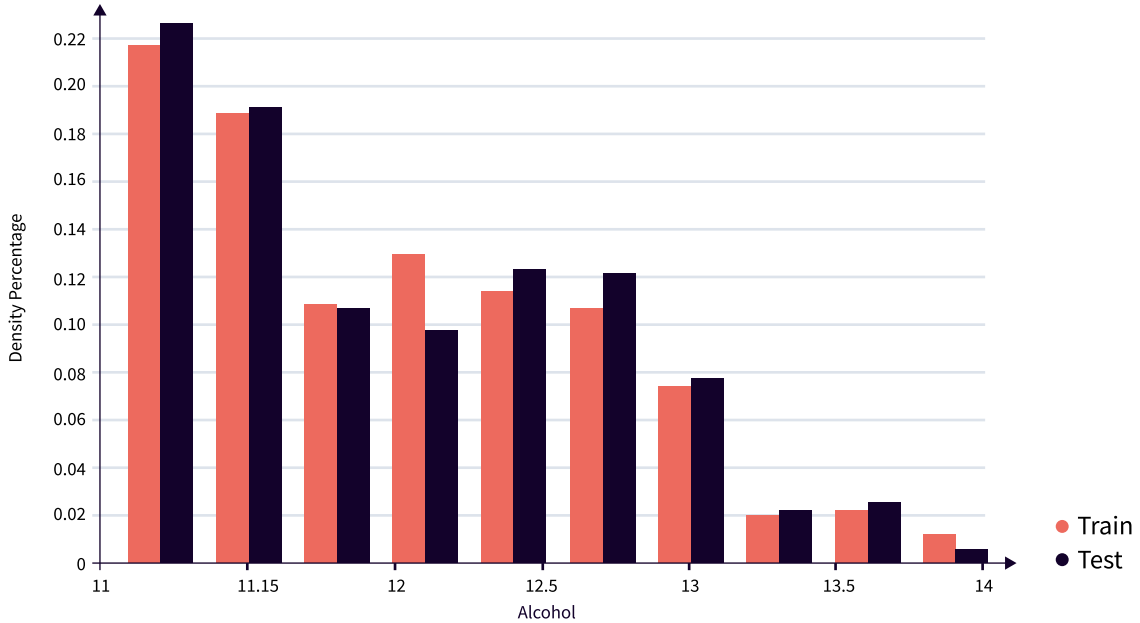


Figure 4: Density distribution of alcohol_above_11_train (red) vs. alcohol_above_11_test (green).

The random sampling gives us two datasets with the similar alcohol rate distribution. Based on the IID assumption, the performance of the model should not change much between hold-out data of train set and test set.

When using this model to score the alcohol_above_11_test dataset, the F1 score is 0.694: when there is no drift in alcohol rate distribution between train and test set, there seems to be no drift with other features too, and the relationship learned between the features and the target holds for the test set.

In formal terms, as neither P(x) nor P(y|x) has changed, the model's performance on the new dataset is similar.

With the trained model above, we take a closer look at the F1 score per alcohol bin on the **alcohol_above_11_test** dataset:

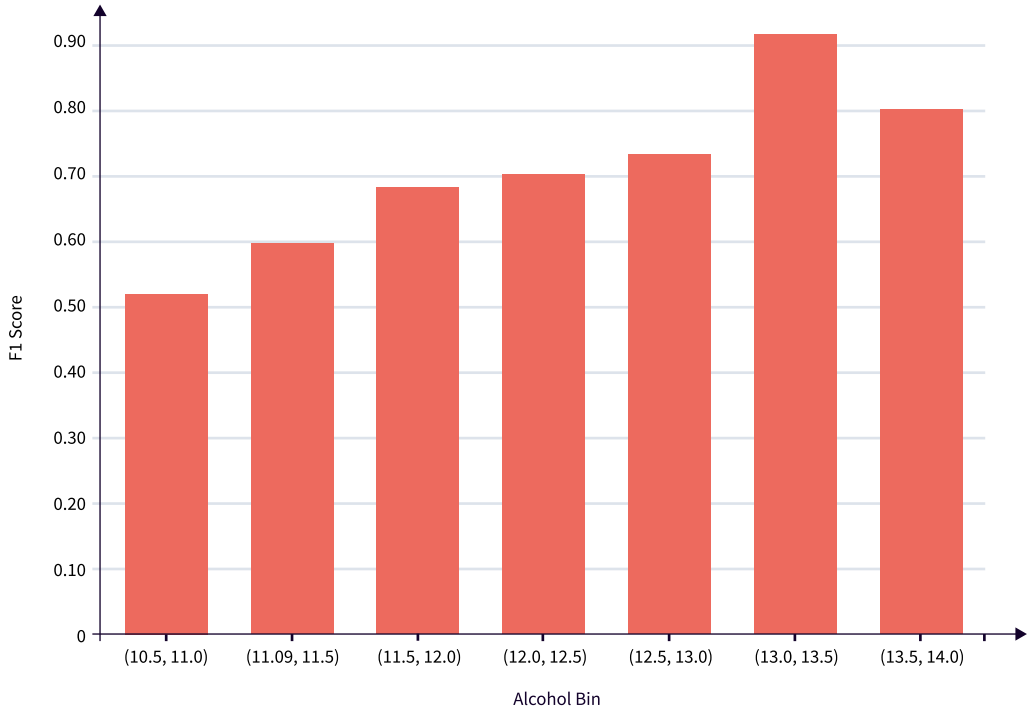


Figure 5: F1 score per alcohol bin on the alcohol_above_11_test dataset.

Two important observations can be drawn from this chart:

- If there are a lot of wines with alcohol levels between 10% and 12% in the new unseen dataset, we should expect the F1 score on these new data to degrade. On the contrary, if more wines between 13% and 14% come in, the performance will be better.
- Until now, the model has only seen alcohol levels between 11% and 14%. If it has to score some wines with alcohol levels out of that range, its performance is unpredictable — it can go either up, or (more probably) down.

The latter observation leads us to the second situation: feature drift.

Situation 2: Covariate Drift (or Feature Drift)

Let's now use this model to predict wine quality of the dataset wine_alcohol_below_11, the one that we assume to be the unseen data in reality. Note that we have not touched it yet during the training phase.

Because of the way the original dataset is split, all the wines in this dataset have an alcohol level out of the training set range. When scoring, we got a general F1-score of 0.295, which is bad compared to the performance in the training phase. The scores per bin are as below:

Figure 6: F1 score per alcohol bin on the alcohol_below_11 dataset.

With the exception of the final bin, which is still in the usual range, the performance drops dramatically for everything else.

This phenomenon is covariate drift (sometimes called feature drift) in action — as discussed in the previous section, it happens when some previously infrequent or even unseen feature vectors become more frequent, and vice versa. However, the relationship between the feature and the target are still the same.

Here's an illustration of this situation in the literature, where the original training set does not faithfully represent the actual population and thus the model learned from it is biased and does not work well on test data:

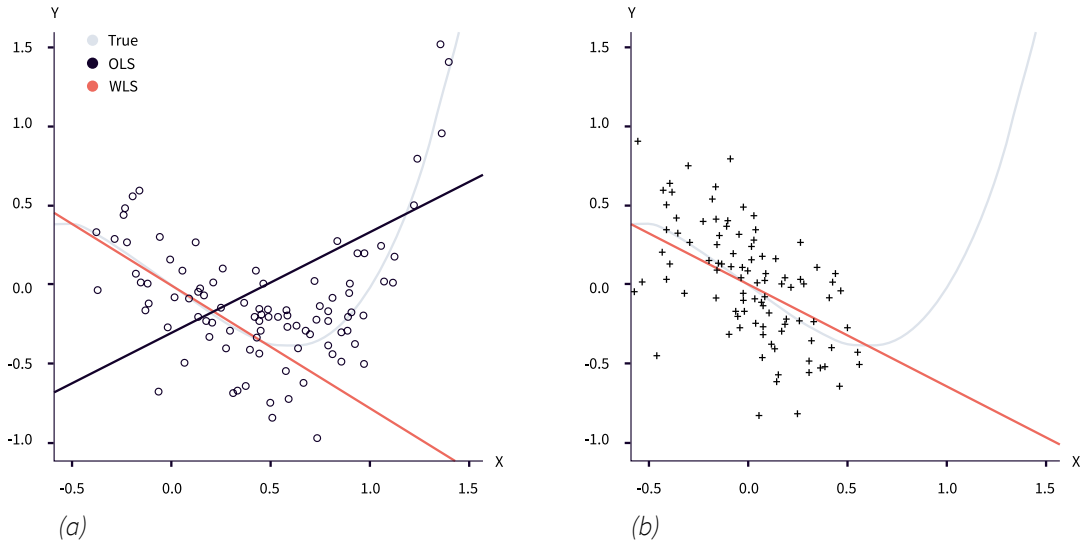


Fig. 1. Fitting of polynomial regression with degree $d = 1$. (a) Samples (x_t, y_t) of size $n = 100$ are generated from $q(y|x)q_0(x)$ and plotted as circles, where the underlying true curve is indicated by the thin dotted line. The solid line is obtained by OLS, and the dotted line is WLS with weight $q_1(x)/q_0(x)$. (b) Samples of $n = 100$ are generated from $q(y|x)q_1(x)$, and the regression line is obtained by OLS.

In our specific use case, model retraining can definitely improve the performance, as the model will have a chance to learn from wines with alcohol range outside of 11% -14%.

Situation 3: Concept Shift

In the beginning, we said that the raw data already comes with a quality score from 0 to 10. Let's take a step back and think about how they get this score.

One logical scenario is that there is a group of wine experts who tested each of the wines and gave out a score. Given that all the wines are from Portugal, we can safely suppose that the experts, too, are Portuguese.

Let's suppose that there are some Portuguese bottles at our local wine shop. The question now is whether the model will work well enough so that we can find a good Portuguese wine for us, people who have more of a taste for French wines.

You can start to see the potential issue: the model was trained to learn about the preference of Portuguese people through the quality scores that they assigned, and it is probable that this score won't be applicable to one who prefers French wine.

We just observed what is called a concept shift or drift: we are still looking at Portuguese wines, so the attributes of the wines does not change, but the notion of "good wine" has changed.

Situation 4: Dual Drift

Now suppose that we take one step further and use the model to score French wines. In this scenario, the attributes of the wines will definitely change, and so does the concept of good wine (as in the previous situation). The results probably won't be very satisfying.

Going Deeper:

Covariate drift is defined as the case where:

$$P(x_s) \neq P(x_t) \; and \; P(y_s|x_s) = P(y_t|x_t)$$

Concept drift is defined as the case where:

$$P(y_s|x_s) \neq P(y_t|x_t) \; and \; P(x_s) = P(x_t)$$

Dual drift is defined as the case where:

$$P(y_s|x_s) \neq P(y_t|x_t) \; and \; P(x_s) \neq P(x_t)$$

We've laid out a framework to talk about data drift in machine learning. This is an issue that can come from either the original training set or new unseen test set. It might be due to a bias in the data collection phase or a change in the context/environment (different place, time, etc.).

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.

Detecting Drift

The Domain Classifier & the Black-Box Shift Detector

In this section, you will learn about two techniques used to assess whether dataset drift is occurring: the domain classifier, looking directly at the input features, and the black-box shift detector, indirectly studying the output predictions.

While both techniques work without ground truth labels, the black-box shift detector is cheaper (as it does not require any training) but in some cases less effective as it needs more data to trigger an alert than the domain classifier.

Although they perform similarly in many cases, they are designed to capture different changes and are complementary for specific types of shift, overall covering a wide spectrum of drift types. Let's see how they work!

Here we focus on classification tasks and take the electricity dataset as an example, on which we train a classifier, say a random forest, to predict whether the price will go UP or DOWN. We call this classifier the primary model. Now let's imagine we get a second new electricity dataset. We want to know if there is a substantial difference between the original dataset and the new dataset before scoring it with our primary model.

Domain Classifier for Covariate Shift

We can train a binary classifier on the same features used in the primary model to predict which domain each observation belongs to: source or target. This is exactly how we build a domain classifier. If the two datasets are different enough, the domain classifier can easily discriminate between the two domains and exhibits high accuracy, which serves as a symptom of drift.

We perform a Binomial test checking how likely is to get such accuracy when there is no drift. If the p-value — i.e., the probability of getting at least such accuracy under the no drift hypothesis — is low enough, this means that a drift might be occurring. In practice, we trigger a drift alert when the p-value of the Binomial test is less than a desired significance level.

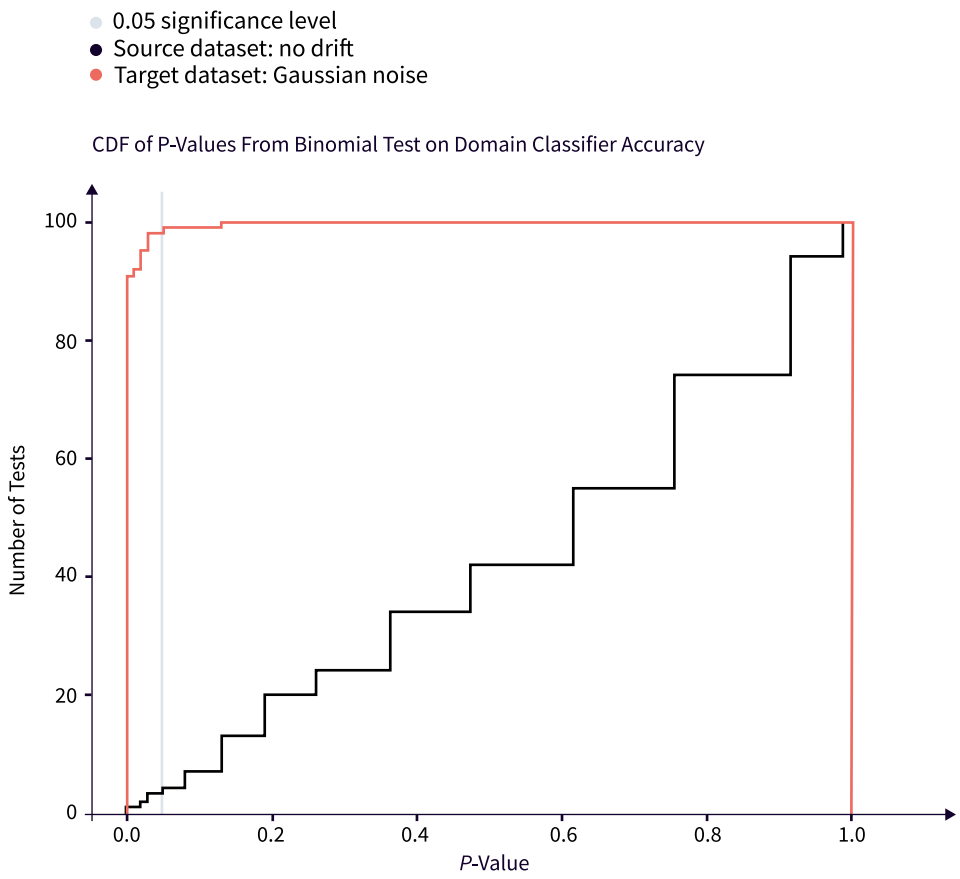


Figure 8: Distributions of the p-values from multiple runs of Binomial test on Domain Classifier accuracy. Comparing random splits of the source dataset the p-value is uniformly distributed (blue), while it is skewed towards zero when comparing the source dataset and a target dataset affected by Gaussian noise (coral).

Using the Binomial test on the domain classifier accuracy is more reliable than directly looking at the accuracy itself. Figure 8 shows how the distributions of the p-values change on the source and target dataset, illustrating how effective this tool is to discriminate the two situations.

By design, the domain classifier is meant to detect alterations of the input features (covariate shift)and is not adequate in case of shifts affecting only the label distribution.

This technique requires performing a new training every time that a batch of new incoming data is available. In this sense, the domain classifier is expensive, but it has many advantages. One of its most appealing aspects is that it can be used *locally*, at the sample level, to predict which observations are the most different from the original dataset. Thus, this simple technique allows a deeper analysis of the drift condition and, in particular, is able to deal with partial drift, a situation when only a part of the observations or a part of the features are drifted.

Black-Box Shift Detector for Prior Shift

What if we can't afford training a new model every time? Or if we have no access to the features used by the deployed model?

In this case, we can build a shift detector on the top of our primary model. Here, we use the primary model as a black-box predictor and look for changes in the distribution of the predictions as a symptom of drift between the source and target datasets[4].

Unlike the domain classifier, the black-box shift detector is designed to capture alterations in the label distribution (prior shift) and is not adequate in case of covariate shifts that have little impact on the prediction, such as small random noise.

After collecting the predictions of the primary model on both source and target datasets, we need to perform a statistical test to check if there is significant difference between the two distributions of predictions. One possibility is to use the Kolmogorov-Smirnov test and compute again the p-value, i.e., the probability of having at least such distance between the two distributions of predictions in case of absent drift.

For this technique we are looking at the predictions (which are vectors of dimension K, the number of classes), and perform K independent univariate Kolmogorov-Smirnov tests. Then we apply the Bonferroni correction, taking the minimum p-value from all the K tests and requiring this p-value to be less than a desired significance level divided by K. Again we can assume there is a drift when the minimum p-value is less than the scaled desired significance level. Figure 9 shows how the distributions of the p-values change on the source and target datasets.

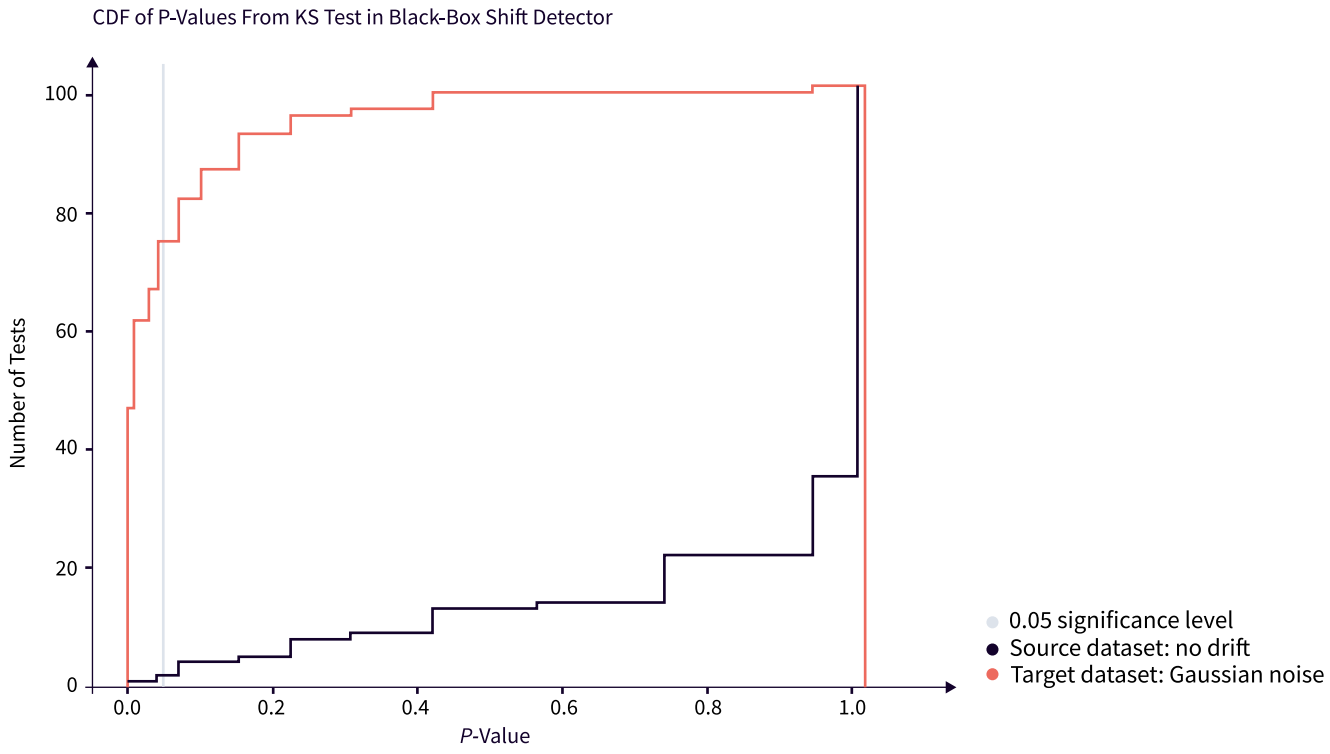


Figure 9: Distributions of the p-values from multiple runs of Kolmogorov-Smirnov test on the predictions of the Black-Box model. Comparing random splits of the source dataset the p-value is skewed towards one (blue), while it is skewed toward zero when comparing the source dataset and a target dataset affected by Gaussian noise (coral).

[4] *Z. C. Lipton et al. "Detecting and Correcting for Label Shift with Black Box Predictors", ICML 2018.*

How can we be sure that only looking at new predictions, one can tell whether the features or the labels distribution has changed? What is the connection between predictions, features and true (unknown) labels?

Under some assumptions, the distribution of the primary model predictions is theoretically shown to be a good surrogate of both features and true unknown labels distributions.

In order to meet the required assumptions to use a black-box shift detector, we first need to have a primary model with decent performance. This one should be easy — hopefully. Second, the source dataset should contain examples from every class, and third we need to be in a situation of prior shift. This one is impossible to know beforehand, as we have no ground-truth labels. Although in real life scenarios it's impossible to know whether those theoretical hypotheses are met, this technique is still useful in practice.

Shaking the Dataset to Measure Robustness

In order to benchmark the drift detectors in a controlled environment, we synthetically apply different types of shift to the electricity dataset:

- **Prior Shift:** Change the fraction of samples belonging to a class (technically this could also change the feature distribution, thus it is not a 'pure' prior shift).
- **Covariate Resampling Shift:** Could be either under-sampling or over-sampling. The former case consists in a different selection of samples according to the features values, for instance keeping 20% of observations measured in the morning. The latter consists in adding artificial samples by interpolation of existing observations.
- **Covariate Gaussian Noise Shift:** Add Gaussian noise to some features of a fraction of samples.
- **Covariate Adversarial Shift:** A more subtle kind of noise, slightly changing the features but inducing the primary model to switch its predicted class. This type of drift is less likely to occur and difficult to detect, but its negative impact is large.

Prior and Covariate Resampling shifts are the most common in a real-world scenario because of the possible selection bias during data collection.

In the experiments, we use 28 different types of shifts from these four broad categories (detailed in this table[5] for the curious readers). For the experiments and to generate a part of the shifts, we use the library provided in the Failing Loudly github repo[6] [7] . For the adversarial shifts, we use the Adversarial Robustness Toolbox[8], while for some of the resampling techniques, we use imbalanced-learn[9].

Benchmarking the Drift Detectors

An ideal drift detector is not only capable of detecting when drift is occurring but is able to do so with only a small amount of new observations. This is essential for the drift alert to be triggered as soon as possible. We want to evaluate both accuracy and efficiency.

We apply 28 different types of shifts to the electricity target dataset from the four categories highlighted in the previous section. For each shift situation, we perform five runs of drift detection by both domain classifier and black-box shift detector. For each run, we perform the detection comparing subsampled versions of the source and drifted datasets at six different sizes: 10, 100, 500, 1000, 5000, and 10000 observations.

[5] *https://airtable.com/tbleAIuUsnMixSmNI/viwqAmJXeO47f3hir?blocks=hide*
[6] *https://github.com/steverab/failing-loudly*
[7] *S. Rabanser et al. "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift", NeurIPS 2019, and its github repo.*
[8] *https://github.com/IBM/adversarial-robustness-toolbox*
[9] *https://github.com/scikit-learn-contrib/imbalanced-learn*

Both the primary model underlying the black-box shift detector and the domain classifier are random forests[10] with scikit-learn default hyper-parameters.

An example of the detection results obtained for a set of covariate resampling shifts is shown in Figure 10. The lines show the average p-values across the runs, while the error areas show the standard deviation of the p-values. The more samples available, the higher the chances to detect the drift.

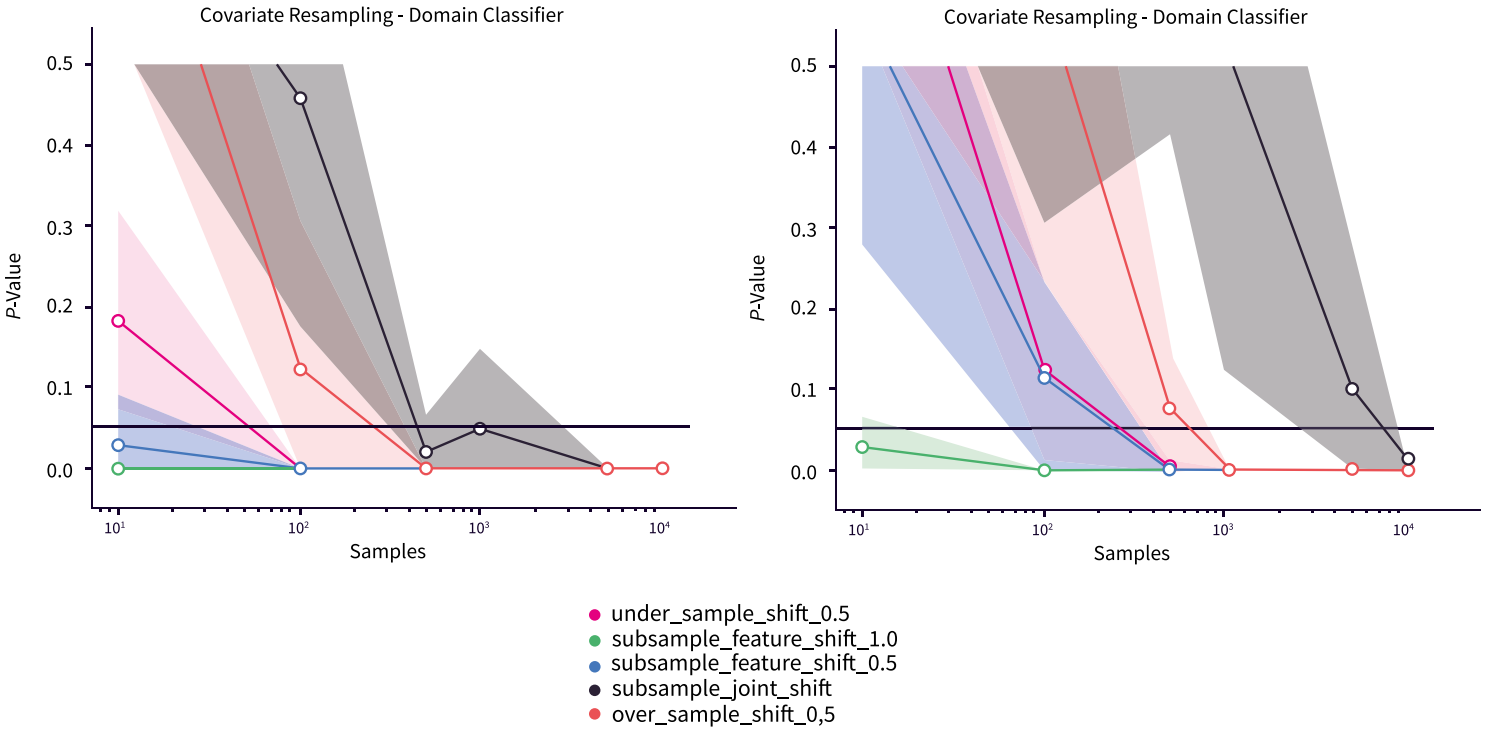


Figure 10: Drift detection results on a set of covariate resampling shifts.

Some of the drifts will cause the primary model to vastly underperform with a large accuracy drop. Although in a real scenario it would be impossible to compute this accuracy drop without ground truth labels, we highlight this information to better illustrate the different types of shift. The drifts yielding to high accuracy drop are the most harmful for our model. Still, whatever alteration may affect the accuracy, it is important to be aware that some change is occurring in the new data and we should probably deal with it.

In order to measure the efficiency of the drift detectors, we score the two techniques with respect to the minimum dataset size required to detect a particular drift. We assign a score from 6 to 1 if the detector first detects a drift at dataset size from 10 to 10000; the detector gets 0 score if it can't detect the drift at all.

The results obtained for the black-box shift detector and the domain classifier are averaged by type of drift and shown in Figure 11 alongside with the average accuracy drop.

[10] *https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html*

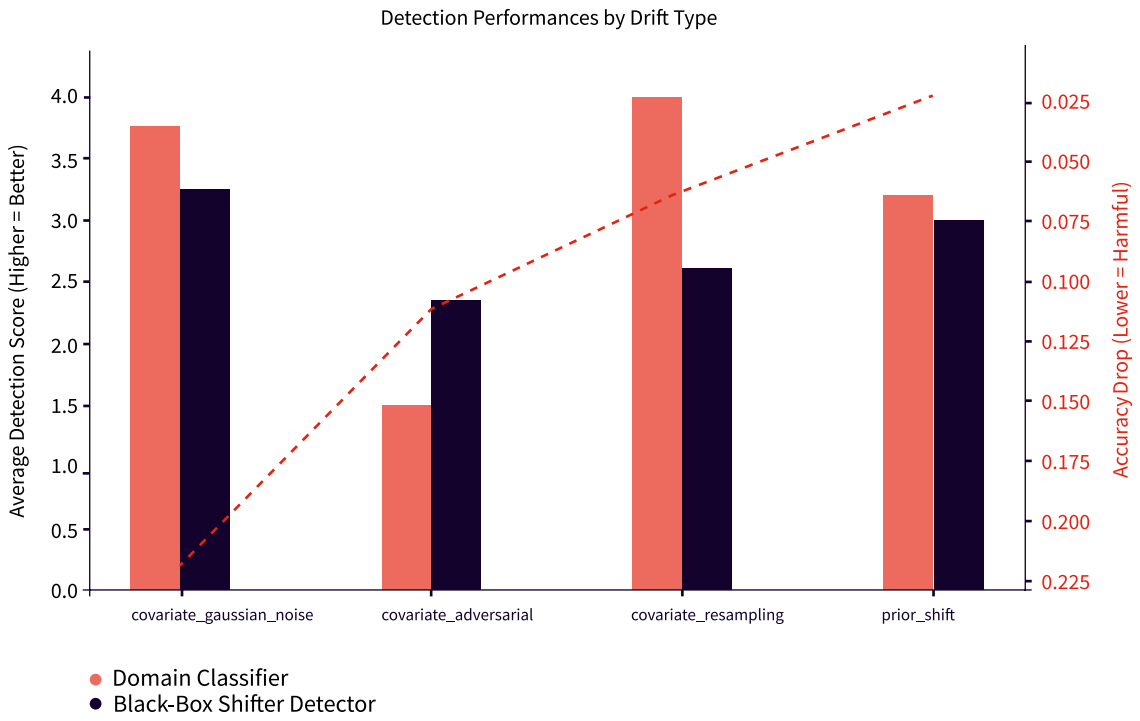


Figure 11: Average detection score of Domain Classifier and Black-Box Shift Detector for different types of drift and primary model accuracy drop due to the drift (red).

Who's the Best Drift Detector?

We can see that the two methods perform similarly in case of Gaussian noise shift. The adversarial noise is so subtle that the Domain Classifier has some trouble catching changes in the features.

On the other hand, as the adversarial noise is designed to make the predictions change, it does not go unnoticed to the black-box shift detector, which is more efficient in this case. The prior and covariate resampling shifts are less harmful to the primary model for this dataset, which also makes the black-box shift detector miss some of them, revealing the domain classifier as more efficient for the most frequent real-world types of shift.

Takeaways

Domain classifier and black-box shift detector are two techniques looking for changes in the data, either directly looking at the features or indirectly at the model predictions.

They perform similarly on many different situations of drift, but there is a divergence in two situations: the domain classifier is more efficient in detecting covariate resampling drift (the most common case of dataset shift) while the black-box shift detector is more efficient in catching adversarial noise.

The black-box shift detector is the way to go when only the prediction is available and the primary model in production is indeed a black-box, but high model performance might be a hard requirement for some specific types of drift.

The domain classifier has also a great advantage of being able to score individual samples, thus providing a tool to perform a deeper local analysis and identify possible anomalies.

References

[1] Moreno-Torres, Jose G., et al. “A unifying view on dataset shift in classification.” Pattern recognition 45.1 (2012): 521–530.

[2] Shimodaira, Hidetoshi. “Improving predictive inference under covariate shift by weighting the log-likelihood function.” Journal of statistical planning and inference 90.2 (2000): 227–244.

[3] Gao, Jing, et al. “A general framework for mining concept-drifting data streams with skewed distributions.” Proceedings of the 2007 siam international conference on data mining. Society for Industrial and Applied Mathematics, 2007

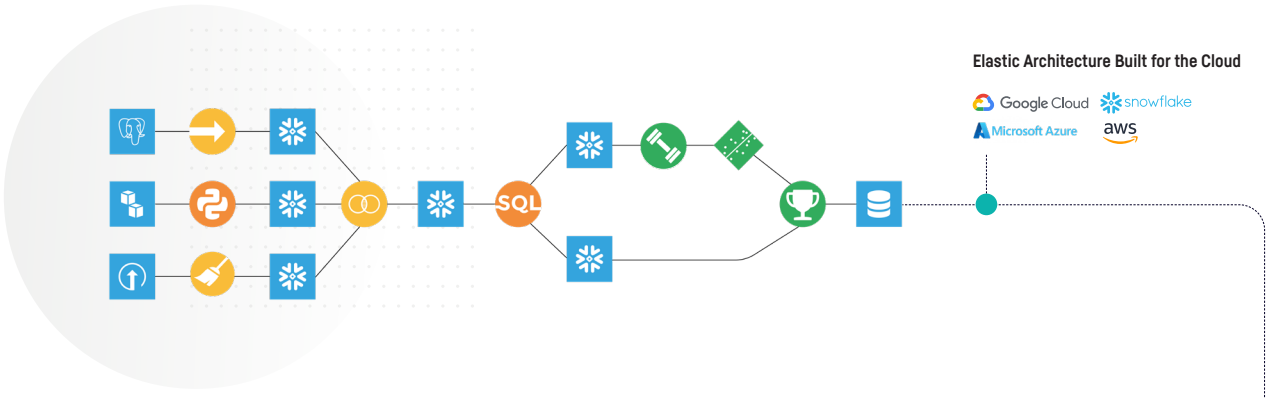
[4] Webb, Geoffrey I., et al. “Characterizing concept drift.” Data Mining and Knowledge Discovery 30.4 (2016): 964–994.

[5] Gama, João, et al. “A survey on concept drift adaptation.” ACM computing surveys (CSUR) 46.4 (2014): 1–37.

[6] PY. Chen et al. “ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models”, ACM Workshop on Artificial Intelligence and Security, 2017.

[7] W. Brendel et al. “Decision-based Adversarial attacks: reliable attacks against black-box machine learning models”, ICLR 2018.

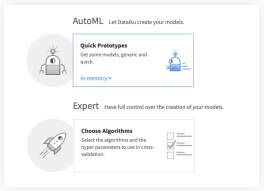
Everyday AI,
Extraordinary People



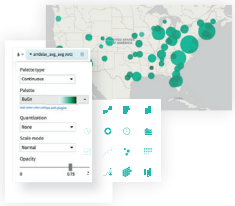
Elastic Architecture Built for the Cloud

Google Cloud snowflake
Microsoft Azure aws

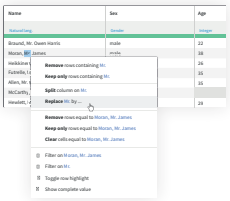
Machine Learning



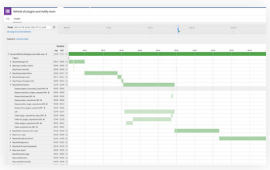
Visualization



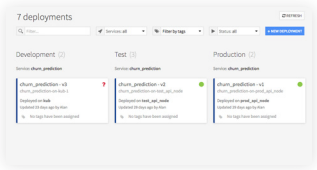
Data Preparation



DataOps



Governance & MLOps



Applications

45,000+
ACTIVE USERS

450+
CUSTOMERS

Dataiku is the platform for Everyday AI, systemizing the use of data for exceptional business results. Organizations that use Dataiku elevate their people (whether technical and working in code or on the business side and low- or no-code) to extraordinary, arming them with the ability to make better day-to-day decisions with data.