Data+Shift: Supporting visual investigation of data distribution shifts by data scientists

João Palmeiro, Beatriz Malveiro, Rita Costa, David Polido, Ricardo Moreira, and Pedro Bizarro

Feedzai

Figure 1: *Views supported by Data+Shift to assist data scientists when investigating data distribution shift. The overview (left) shows how much the distribution of each feature has changed over time. The investigation view (center) enables exploration of related features to better understand sources and the impact of the shift. In addition to the selected feature(s), the related features are shown grouped in ancestors and descendants. The detail view (right) shows how the distribution of a single feature has changed over time.*

Abstract
Machine learning on data streams is increasingly more present in multiple domains. However, there is often data distribution shift that can lead machine learning models to make incorrect decisions. While there are automatic methods to detect when drift is happening, human analysis, often by data scientists, is essential to diagnose the causes of the problem and adjust the system. We propose Data+Shift, a visual analytics tool to support data scientists in the task of investigating the underlying factors of shift in data features in the context of fraud detection. Design requirements were derived from interviews with data scientists. Data+Shift is integrated with JupyterLab and can be used alongside other data science tools. We validated our approach with a think-aloud experiment where a data scientist used the tool for a fraud detection use case.

CCS Concepts
*• **Human-centered computing** → **Visualization systems and tools;** • **Computing methodologies** → Machine learning;*

1. Introduction

Machine learning (ML) on real-time stream processing systems is increasingly present in multiple domains. These complex systems rely on a data stream that is assumed to follow the same distribution as previously seen. However, there is often data shift which impacts the system performance and leads to degradation over time [PSB19]. In the fraud detection domain, data shift can result from multiple factors, such as seasonality, new buying patterns, or some technical issues that can corrupt the observed data.

Depending on the underlying factor behind the drift, its effect can be gradual or sudden, and impact specific features of the data or all of them [PBC*18]. In the data streams literature, these changes are known as covariate shift and concept drift. Several methods [PSB19, BG07, GSaR13, dRFMB16] have been proposed to detect these changes. These automatic systems can signal data features that might be drifting and in which periods. However, while this automatic signal is important to kick-start an investigation, it is only the beginning. Human analysis, often by data scientists, is

essential to diagnose what factors impact the system and then take appropriate measures to adjust it.

In this paper, we introduce a visual analytics tool to support data scientists in the tasks of investigating the underlying factors of drift in data features: *Data+Shift*. Our tool relies on an algorithm that calculates if a feature's distribution has significantly diverged from a reference. This method uses hypothesis testing to check if the difference between distributions is significant. Our visual interface provides a way to analyze the *p*-values of these tests across time for all features. The interactive visualization starts with an overview and supports the user throughout the investigation (Figure 1). *Data+Shift* is directed at data scientists that use Jupyter-Lab [KRKP*16] to perform data analysis and model development. To validate the use of *Data+Shift* for data distribution shift exploration, we gathered feedback from an expert. We asked a data scientist to freely explore the tool in the context of ML for fraud detection, following a think-aloud protocol [vSBS94].

2. Related Work

Visual analytics for data distribution shift (as part of ML systems) is an underexplored field [SKEA20]. For this work, the following topics are considered: (1) interfaces for data distribution shift visualization; (2) heatmaps/matrix-based visualizations for high-dimensional data; (3) time series visualization.

For identification and analysis of data distribution shifts, DataShiftExplorer [SKEA20] presents a feature-oriented interface designed from a previously defined design space. It's the closest work to our proposed solution. On the other hand, Clustergrammer [FGR*17] is an interactive heatmap visualization, embeddable in Jupyter Notebook, prepared particularly for biological data. In terms of implementation, it's the closest solution to ours, albeit for a substantially different use case. In the open-source software landscape, some packages [DE22, HI22] offer visualization capabilities for data distribution shift analysis.

We also looked at the literature on heatmaps for high-dimensional data to complement our design process and account for techniques that could inspire our work focused on data distribution shift. ContiMap [PND20] features a heatmap that, instead of rendering individual cells, sorts and groups them, producing a high-level visualization. Similarly, HiGlass [KAL*18] also provides a scalable heatmap-based interface for Genomics. In Compadre's heatmap [CAFS20], each cell encodes a distance between the original data and/or low-dimensional projections. Furthermore, the top and left borders of the heatmap are used to stack the labels associated with each point. In H-Matrix [SZEAC19], a multi-part linguistic visualization, it's possible to find a heatmap whose cells follow a divergent color scale for an error metric. Responsive matrix cells [HBS*20] offer local zoomable regions, where the user can check other relevant visualizations *in-situ* (with different levels of detail).

Similarly, since the analysis and comparison of features over time is relevant to *Data+Shift*, we also consider techniques for visualizing time series applied in different contexts. Cloud-Lines [KBK11] is a dynamic and space-efficient time series visualization, using an importance function to adapt as new data becomes

available. In ConceptExplorer [WCX*20], an interface for detecting and analyzing time series drifts, two snapshots (before and after a drift, for example) can be compared in a heatmap whose halves are independent. In addition, each feature on the axes is discretized or categorized, allowing for a more granular analysis. Particularly to explore cyclical patterns in time series, Ceneda et al. [CGMT18] enrich a spiral chart with user-adjustable settings and sensible suggestions for them. SAX Navigator [RSM*19] is a hierarchical interface for analyzing and comparing time series. They are simplified and clustered, and the user has access to small heatmaps with summaries for these clusters.

None of these works provide an ML-oriented workflow for analyzing data distribution shifts with various levels of information and interactivity like *Data+Shift*, integrated with JupyterLab and Python.

3. *Data+Shift*

In this section, we describe the algorithm used by *Data+Shift*, the user requirements derived from interviews, and the visual interface.

3.1. Algorithm

The algorithm used by the tool aims at detecting covariate drift by measuring the divergences of each feature's distribution over time. Our algorithm is most similar to [PSB19, dRFMB16], which does not rely on labels. The algorithm works in two steps. It first starts by learning the historical behavior of the features in the dataset and then evaluates how much they diverge over time.

Similar to how machine learning models are trained, the historical behavior is learned from a reference dataset. This can be mapped to the "train" stage of the usual ML model training pipeline. The reference dataset covers a time interval when the data is assumed to be stable and follow expected patterns. In this stage, for every feature, a reference histogram of the distribution in the full reference dataset is stored. To measure the relative stability of the dataset in this period, sampled windowed histograms are also computed and then compared with the reference histogram obtaining a distribution of divergences.

In the second stage, we evaluate the drifts over time on a second dataset. As an example, let us assume we run a daily analysis. Then, for each feature, we compare its histogram in each day of the new dataset with the reference histogram and calculate its divergence measure. However, we need to understand how significant this divergence is compared to the expected distribution of divergences obtained from the reference dataset. We perform a test for what would be the likelihood of observing such a divergence value under the hypothesis that the feature distribution remains similar to the reference. The *p*-value of this test is then the probability that the feature keeps the same distribution as in the reference. To create an alarm for a single feature, we can define a significance level α and use that as a threshold. If the *p*-value is less than the threshold, we consider that the feature is under drift, and an alarm for it is triggered. In order for all the *p*-values to be comparable with the same threshold, these need to be normalized [Hol79]. Hence, the normalized *p*-values are no longer probabilities, as they can take any value greater than zero.
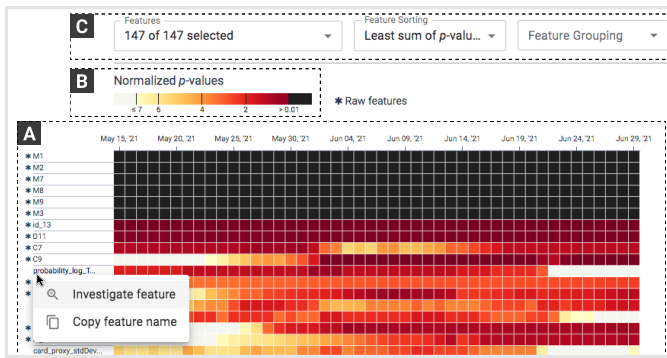
Figure 2: *Overview screen. (A) Heatmap of the p-values for all features. The color encoding is described in the legend (B). A toolbar (C) allows the user to interact with the heatmap.*

3.2. User Requirements

To help us with the design process, we conducted semi-structured interviews at Feedzai with four data scientists. From the interviews, we concluded that, in general, data scientists want to compare the data distributions of features considering the most recent time window with the training, i.e., historical datasets. They leverage charts and summary statistics via single-use Jupyter notebooks or spreadsheets. When starting the analysis, they prioritize the most important features for predicting the target variable, raw fields, and features obtained from data enrichers. In addition, there are two main triggers for a data scientist to investigate features: changes in model/system performance and deployments.

The main goal is to quickly generate hypotheses and decide the next steps to ensure that the performance of ML models is not affected by data distribution shifts and data quality issues. In fraud detection, these hypotheses often consider seasonality, customer behavior changes, or technical issues. These causes are associated with different patterns of data distribution shift. For example, technical issues often cause sudden changes, while changing behavior happens gradually. For *Data+Shift* to support data scientists in the tasks of generating new hypotheses and validating them, we have established the following user requirements:

R1 Identify and compare data distribution shift patterns.
R2 Identify relevant features and periods.
R3 Compare related features.
R4 Identify the main changing values in the distributions of the features.
R5 Integrate with the current workflow of data scientists.

Although the four data scientists work in the fraud detection domain, the elicited requirements appear to be generic. *Data+Shift* requires further testing to validate its application to other use cases.

3.3. Visual Interface

3.3.1. Overview and Investigation

The overview visualization shows how the features diverged from their reference over time (R1, R2). The main element of the appli-

cation is the heatmap (Figure 2.A), where each cell color represents the *p*-value for a given feature at a time instance.

- **Tripartite color scale:** The legend on top of the heatmap describes the color encoding, which uses a tripartite scale (Figure 2.B). Any *p*-value above the analysis threshold will be colored in a light gray, a value lower than the alert threshold will be black, and a value in between these thresholds will be colored according to a gradient from light yellow to dark red. The analysis threshold serves to focus on a shorter, more relevant range of *p*-values.
- **Toolbar:** It is possible to interact with the layout of the heatmap using the toolbar at the top (Figure 2.C). It enables three distinct operations over the heatmap:
  - **Selection** allows the user to search, select, and remove specific features from the view.
  - **Sorting** allows the user to change the vertical ordering of the features to any of the four different sorting options: *Original*, *Alphabetical*, *Most alarms*, and *Least sum of p-values*.
  - **Grouping** allows for a vertical split of the heatmap, grouping features by a specific attribute (e.g., *Raw / Engineered* breakdown).

It is possible to investigate a feature in more detail using the **investigation view** (Figure 1, center). This view can be triggered from any single feature. When right-clicking on a feature name on the heatmap Y-axis, a context menu will appear with the option *Investigate feature* (Figure 2). Once this option is selected, the view of the heatmap will change. The layout will be rearranged into two sections: one for the *Investigation features* and below it, a section for *Related features*.

- **Related features:** This section is divided into *ancestor* and *descendant* features. These are fields that were used to compute the current feature and fields that were computed from the current feature, respectively. This allows the data scientist to see how the changes in that feature correlate with the changes in their related features (R3). The text input box on the *Investigation features* Y-axis can be used to search and select another feature to add to the view. When there are multiple features in the investigation, the user can choose to see all related features or only the ones common to these features.

3.3.2. Understanding Individual Features

The histogram method shows how the distribution of a single feature changed over time (R4). The main element of the application is the histogram/bar chart (Figure 3.A), where it is possible to see the relative frequency of a bin (for numerical features) or value (for categorical features) at a target date and compare it with the relative frequency in the reference period. The frequencies at the target date are represented by the light grey bars and at the reference period by horizontal black lines. There is a separate bar on the right of the main histogram that shows the frequency of *NaN* values, in the case of a numeric feature, and the total of missing and new values not seen in the reference period, in the case of a categorical feature.

- **Target date selector:** On top of the histogram, there is a toolbar (Figure 3.B) where it is possible to select a target date. This can be done using a slider (Figure 3.C) that covers the time interval
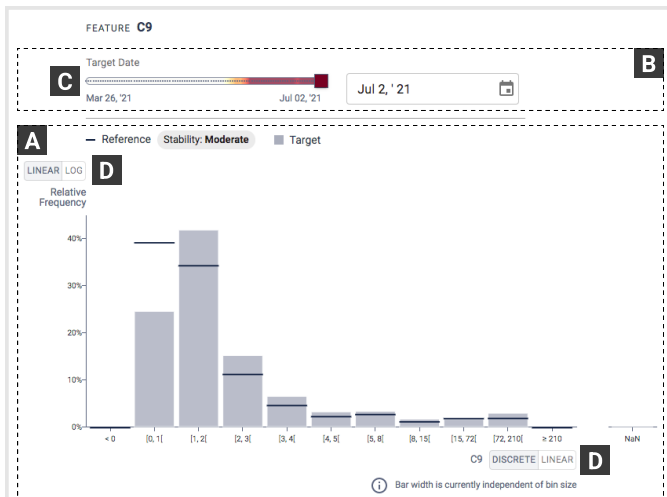
Figure 3: *Detail screen. Histogram (A) to visualize how the distribution of a feature has changed. Toolbar (B) with slider (C) to select a target date. The scale of the axes can be changed (D).*

in analysis or with a date picker. The slider uses the same color encoding of the heatmap to visualize how much the distributions diverged over time. The date picker is most useful when selecting a specific date of interest.

- **Switchable axis scales:** On the histogram view, the user can use toggle buttons (Figure 3.D) to change the Y-axis to use either linear (default) or logarithmic scales. For numerical features, it is also possible to change the X-axis to use either a discrete (default) or linear scale.
- **Brush interaction:** To filter the feature domain on the X-axis, the user can use the brush by clicking and dragging to select a region on the chart area.

3.4. Implementation

Data+Shift is implemented as a Python package. Its API works with data structures used in data science such as pandas DataFrames [WM10, pdt20]. The visualizations are React web applications where charts are built with D3 [BOH11] and visx [SWWA21] in Canvas and SVG. User-interface components, such as buttons and selections, are from MUI [Mat21]. The back-end, responsible for calculating the *p*-values and signaling the alerts is implemented in Python. The visualization methods take as input the result of the evaluation of a dataset and plot the visualizations using IPython's display(). Our implementation approach is similar to *PipelineProfiler* [OCL*20]. The goal of the design is to integrate *Data-Shift* with the existing workflows of data scientist in JupyterLab (R5).

4. Expert Feedback

To validate the suitability of *Data+Shift* for our goals, we ran a second interview with a data scientist at Feedzai who had previously helped us understand design requirements. For this experiment, we considered data on model development for fraud detection at an e-commerce merchant, as the data scientist was already familiar with the use case. We asked the data scientist to use *Data+Shift* to freely explore data distribution shifts during a 45-minute session. We requested him to describe his thought process, following a think-aloud protocol. In this section, we present some of the findings from this session.

When sorting features by *Most alarms* in the **overview screen**, he immediately noticed five features being alerted from the first day of the analysis. "I wasn't expecting them to be so different. I don't see a reason why it would be this different. This could suggest that there is an issue with data quality." He considered the sorting option *Least sum of p-values* to be the most useful to identify drifting features and preemptively detect risks for model performance. The data scientist appreciated the **investigation screen**. "This is interesting. It makes a lot of sense and helps me understand other features that can be affected by this drift." He pointed out that this view can be useful to detect the source of the shift: "An engineered feature can be calculated from many raw fields, and seeing the *p*-values of its raw ancestors helps me understand which might be causing the drift." When using the **histogram**, he enjoyed the slider to see the distribution change over time. "I really like the color encoding in the slider. It really helps to understand what is going on with the feature."

The data scientist mentioned that the workflow that took around 20 minutes with *Data+Shift* can take up to one hour without it. "The tool allows me to quickly direct my attention to the source of the problem instead of manually going feature by feature." However, he pointed out that the complete analysis of these shifting features was still necessary. A relevant task that he mentioned is reviewing the data records of each respective drift pattern. The data scientist also hinted that it could be pertinent to explore multidimensional feature drift.

5. Limitations and Future Work

During the development of *Data+Shift*, we iterated our solution with the help of data scientists. However, the proposed solution lacks formal user testing. As such, we would like to understand the suitability of this tool with more data scientists to identify potential improvements. Additionally, there is functionality that we did not prioritize for this first version but that we believe can add value to the tool. As examples, we want to implement a more flexible way to truncate Y-axis labels, add visual links between each feature and their related ones, and a mechanism to take stateful screenshots. To make *Data+Shift* more accessible, we also intend to improve support for keyboard navigation and screen readers. Finally, we consider that it is valuable to expand the functionality of *Data+Shift* to support data scientists in a more granular subsequent analysis, exploring ways to visualize individual data records.

Acknowledgments

References

[BG07] BIFET A., GAVALDÀ R.: Learning from time-changing data with adaptive windowing. In *In SIAM International Conference on Data Mining* (2007). 1

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011). URL: http://idl.cs.washington.edu/papers/d3. 4

[CAFS20] CUTURA R., AUPETIT M., FEKETE J.-D., SEDLMAIR M.: Comparing and Exploring High-Dimensional Data with Dimensionality Reduction Algorithms and Matrix Visualizations. In *AVI' 20 - International Conference on Advanced Visual Interfaces* (Ischia Island, Italy, Sept. 2020). URL: https://hal.inria.fr/hal-02861899, doi:10.1145/3399715.3399875. 2

[CGMT18] CENEDA D., GSCHWANDTNER T., MIKSCH S., TOMINSKI C.: Guided Visual Exploration of Cyclical Patterns in Time-series, 2018. Posterpräsentation: Visualization in Data Science (VDS at IEEE VIS 2018), Berlin; 2018-10-22. URL: https://publik.tuwien.ac.at/files/publik_272789.pdf. 2

[DE22] DRAL E., EVIDENTLY AI: Evidently, 2 2022. URL: https://github.com/evidentlyai/evidently. 2

[dRFMB16] DOS REIS D. M., FLACH P., MATWIN S., BATISTA G.: Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. KDD '16, Association for Computing Machinery, p. 1545–1554. doi:10.1145/2939672.2939836. 1, 2

[FGR*17] FERNANDEZ N. F., GUNDERSEN G. W., RAHMAN A., GRIMES M. L., RIKOVA K., HORNBECK P., MA'AYAN A.: Clustergrammer, a web-based heatmap visualization and analysis tool for high-dimensional biological data. *Scientific Data 4*, 1 (Oct 2017), 170151. URL: https://doi.org/10.1038/sdata.2017.151, doi:10.1038/sdata.2017.151. 2

[GSaR13] GAMA J. a., SEBASTIÃO R., RODRIGUES P. P.: On evaluating stream learning algorithms. *Mach. Learn. 90*, 3 (mar 2013), 317–346. doi:10.1007/s10994-012-5320-9. 1

[HBS*20] HORAK T., BERGER P., SCHUMANN H., DACHSELT R., TOMINSKI C.: Responsive Matrix Cells: A Focus+Context Approach for Exploring and Editing Multivariate Graphs. *CoRR abs/2009.03385* (2020). URL: https://arxiv.org/abs/2009.03385, arXiv:2009.03385. 2

[HI22] HAVIV Y., IGUAZIO: MLRun, 2 2022. URL: https://github.com/mlrun/mlrun. 2

[Hol79] HOLM S.: A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics 6* (1979), 65–70. 2

[KAL*18] KERPEDJIEV P., ABDENNUR N., LEKSCHAS F., MCCALLUM C., DINKLA K., STROBELT H., LUBER J. M., OUELLETTE S. B., AZHIR A., KUMAR N., HWANG J., LEE S., ALVER B. H., PFISTER H., MIRNY L. A., PARK P. J., GEHLENBORG N.: HiGlass: web-based visual exploration and analysis of genome interaction maps. *Genome Biology 19*, 1 (Aug 2018), 125. URL: https://doi.org/10.1186/s13059-018-1486-1, doi:10.1186/s13059-018-1486-1. 2

[KBK11] KRSTAJIC M., BERTINI E., KEIM D.: CloudLines: Compact Display of Event Episodes in Multiple Time-Series. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2432–2439. doi:10.1109/TVCG.2011.179. 2

[KRKP*16] KLUYVER T., RAGAN-KELLEY B., PÉREZ F., GRANGER B., BUSSONNIER M., FREDERIC J., KELLEY K., HAMRICK J., GROUT J., CORLAY S., IVANOV P., AVILA D., ABDALLA S., WILLING C., [UNKNOWN] J.: *Jupyter Notebooks – a publishing format for reproducible computational workflows*. IOS Press, Netherlands, 2016, pp. 87–90. doi:10.3233/978-1-61499-649-1-87. 2

[Mat21] MATERIAL-UI SAS: MUI, 11 2021. URL: https://github.com/mui/material-ui. 4

[OCL*20] ONO J. P., CASTELO S., LOPEZ R., BERTINI E., FREIRE J., SILVA C.: Pipelineprofiler: A visual analytics tool for the exploration of automl pipelines, 2020. URL: https://arxiv.org/abs/2005.00160, doi:10.48550/ARXIV.2005.00160. 4

[PBC*18] POZZOLO A. D., BORACCHI G., CAELEN O., ALIPPI C., BONTEMPI G.: Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems 29* (2018), 3784–3797. 1

[pdt20] PANDAS DEVELOPMENT TEAM T.: pandas-dev/pandas: Pandas, Feb. 2020. URL: https://doi.org/10.5281/zenodo.3509134, doi:10.5281/zenodo.3509134. 4

[PND20] PHAM V., NGUYEN N., DANG T.: ContiMap: Continuous Heatmap for Large Time Series Data. In *2020 Visualization in Data Science (VDS)* (Oct 2020), pp. 42–51. doi:10.1109/VDS51726.2020.00009. 2

[PSB19] PINTO F., SAMPAIO M. O. P., BIZARRO P.: Automatic model monitoring for data streams. URL: https://arxiv.org/abs/1908.04240, doi:10.48550/ARXIV.1908.04240. 1, 2

[RSM*19] RUTA N., SAWADA N., MCKEOUGH K., BEHRISCH M., BEYER J.: SAX Navigator: Time Series Exploration through Hierarchical Clustering. In *Proceedings of 2019 IEEE Visualization Conference, Short Papers* (2019), pp. 236–240. doi:10.1109/VISUAL.2019.8933618. 2

[SKEA20] SCHNEIDER B., KEIM D. A., EL-ASSADY M.: DataShiftExplorer: Visualizing and Comparing Change in Multidimensional Data for Supervised Learning. *Proc. Int. Conf. Inf. Vis. Theory Appl.* (2020). 2

[SWWA21] SHOFF H., WILLIAMS C., WONGSUPHASAWAT K., AIRBNB: visx, 11 2021. URL: https://github.com/airbnb/visx. 4

[SZEAC19] SHIMABUKURO M., ZIPF J., EL-ASSADY M., COLLINS C.: H-Matrix: Hierarchical Matrix for visual analysis of cross-linguistic features in large learner corpora. In *Proc. IEEE Conf. Inf. Vis. (Short Pap.)* (2019). URL: http://vialab.science.uoit.ca/wp-content/papercite-data/pdf/shi2019a.pdf. 2

[vSBS94] VAN SOMEREN M., BARNARD Y., SANDBERG J.: *The think aloud method: a practical approach to modelling cognitive processes*. Academic Press, 1994. 2

[WCX*20] WANG X., CHEN W., XIA J., CHEN Z., XU D., WU X., XU M., SCHRECK T.: ConceptExplorer: Visual Analysis of Concept Drifts in Multi-source Time-series Data. *CoRR abs/2007.15272* (2020). URL: https://arxiv.org/abs/2007.15272, arXiv:2007.15272. 2

[WM10] WES MCKINNEY: Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference* (2010), Stéfan van der Walt, Jarrod Millman, (Eds.), pp. 56–61. doi:10.25080/Majora-92bf1922-00a. 4