Frouros: A Python library for drift detection in Machine Learning problems

Jaime Céspedes Sisniega  
Alvaro López García
Instituto de Física de Cantabria (IFCA), CSIC-UC, Spain

Abstract

Frouros is a Python library capable of detecting drift in machine learning problems. It provides a combination of classical and more recent algorithms for drift detection: both supervised and unsupervised, as well as some capable of acting in a semi-supervised manner. We have designed it with the objective of being easily integrated with the scikit-learn library, implementing the same application programming interface. The library is developed following a set of best development and continuous integration practices to ensure ease of maintenance and extensibility. The source code is available at https://github.com/IFCA/frouros.

Keywords: Machine learning, Drift detection, Concept drift, Data drift, Python

1. Introduction

When it comes to deploying machine learning models in real-world applications, there is often a mistaken belief that models will be used in a stationary environment. This belief assumes that the same concepts learned during the training phase will remain valid at inference time Gama et al. (2004), or that training samples and production-time samples will come from the same distribution Ackerman et al. (2021). However, in real-world scenarios this is often not true, and both situations may result in some type of drift that can end up affecting model’s performance Žliobaitė et al. (2016). In addition, due the high cost of collecting and labeling samples, this performance loss can often not be confirmed in many real-world problems and other methods that only rely on distribution changes must be used.

In this paper we present Frouros, a Python library for drift detection in machine learning problems. The library tries to fulfill two main objectives: 1. to be able to easily integrate in a machine learning model development pipeline with the scikit-learn Pedregosa et al. (2011) library; 2. to unify in a single library the part of concept drift detection and adaptation (traditionally researched and used for streaming/evolving data streams and incremental learning Khamassi et al. (2018)) with the research of change detection in the covariate distributions (also known as data shift, related to the field of statistical two-sample testing Rabanser et al. (2019) and methods that measure distance between distributions Goldenberg and Webb (2019)).

2. Drift detection

Traditionally there has been little consensus on the terminology and definitions of the different types of drift, as stated in Moreno-Torres et al. (2012). In order to adopt some clear definitions for the remainder of this paper, we apply those used in Gama et al. (2014)
for the concept drift part, in combination with those used in Rabanser et al. (2019)’s work for detecting dataset shift using only the covariates. Therefore, we set up the following definitions assuming two different time points, $t_0$ and $t_1$:

**Concept drift.** There is a change in the joint probability $P(X,y) = P(y|X)P(X)$, with or without a change in $P(X)$. Thus, it can be defined as $P_{t_0}(X,y) \neq P_{t_1}(X,y)$. Known as real concept drift when changes only affect $P(y|X)$ Gama et al. (2014).

**Virtual drift.** There is a change in $P(X)$ but does not affect the conditional probability $P(y|X)$ Gama et al. (2014); Ramírez-Gallego et al. (2017). Thus, $P_{t_0}(X) \neq P_{t_1}(X)$ and $P_{t_0}(y|X) = P_{t_1}(y|X)$.

**Data drift.** As well as virtual drift, there is a change in $P(X)$ but due the fact that there is no labeled data $y$ available, it cannot be verified that $P(y|X)$ is being affected or not. Therefore, this type of drift only focuses in the distribution of the covariates $P(X)$, so $P_{t_0}(X) \neq P_{t_1}(X)$. Hereafter, we rename dataset shift Rabanser et al. (2019) to data drift in order to maintain consistency with the above definitions and with some of the related software mentioned in Section 5 that also refer to it as data drift Van Looveren et al. (2019); Leigh et al. (2022).

3. Overview and design

The design and implementation of the library has been carried out with the aim of making it compatible with the use of scikit-learn, both through the use of its estimators, as well as its pipeline. We also provide transformations that can be included in a pipeline and allow to use multiple of the unsupervised methods, described in Section 3.2, at the same time. Additionally, we provide prequential error metrics Gama et al. (2013) to evaluate the performance of supervised methods described in Section 3.1.

Detection methods are divided in the supervised, unsupervised and semi-supervised categories depending on the type of drift they can detect, according to the definitions given in Section 2, and how they detect it. These categories are explained in what follows.

3.1 Supervised

These methods are aimed at detecting concept drift, so in order to update the detector they require the ground-truth labels of the predictions that have previously been made.

In terms of implementation, the detector wraps the scikit-learn estimator and receives through the update method the value that is used to perform the necessary steps (at least update detector’s inner statistics and check if drift is occurring) on each iteration. In addition to receiving a scikit-learn estimator when the detector is instantiated, it receives a configuration class that contains a set of parameters that determine how it will behave.

Despite the fact that detectors can be updated by interacting with them directly (via update method), we provide the following helper classes, so-called modes, to facilitate the interaction with them.

IncrementalLearningMode works with methods that use warning and drift thresholds, and scikit-learn estimators that support partial_fit method. It acts on an instance-incremental manner by passing a tuple that contains a feature vector $\vec{x}$, the prediction made
by the model and the ground-truth label, only when the user requires it. Additionally, a value functions that uses that tuple needs to be provided to compute a value for the detector, usually an error function. Therefore, this helper class allows to check the presence of drift and add samples to the model’s decision.

NormalMode can be used by all the supervised methods without incrementally training the wrapped model. It only checks if the detector has raised the drift flag and resets its inner statistics. Like IncrementalLearningMode, a value function must be provided, but estimators that only support fit method are allowed, so it is not restricted only to those that support partial_fit.

3.2 Unsupervised

The unsupervised methods are focused on detecting data drift by considering only the covariates and regardless of the existence of labels or its lack thereof. Therefore, these algorithms try to detect changes at a feature level by comparing new data distributions against reference data distributions.

BaseEstimator and TransformerMixin classes from scikit-learn are used to implement these type of methods. The fit method stores the reference distribution and the transform method applies the corresponding algorithm to compare new samples distribution with the reference distribution. In order to check if drift has happened after calling transform method, distance or test attributes can be acceded, depending on the type of algorithm used. All the implemented methods act in batch mode, expecting to receive multiple samples each time a drift check is performed. Moreover, these detectors are implemented considering the type of data that they are expected to work with: categorical or numerical, and the number of features that can be considered: univariate or multivariate.

3.3 Semi-supervised

As with supervised methods, semi-supervised ones aim to detect concept drift by providing ground-truth labels only when they are required. Therefore, when drift is suspected (equivalent to the warning zone present in several of the supervised algorithms) it is necessary to manually provide the detector with new labeled samples coming from an external entity to verify whether drift is occurring or not. If drift is present, the model is replaced by one trained with these new samples, otherwise the algorithm returns to a control state.

As far as implementation is concerned, their way of working is similar to supervised methods with the exception of the use of the helper classes described in Section 3.1.

4. Development

With the intention of following a set of open source software development standards that allow the maintainability and extensibility of the library over time, we emphasize the following areas:

Continuous integration. A Continuous integration workflow based on GitHub Actions ensures that new modifications easily integrate with the existing code base and that they are compatible with multiple Python versions.
Documentation. An API documentation is provided using sphinx and hosted in Read the Docs\textsuperscript{1} website. Some basic examples on the use of these detection methods are included in the documentation.

Quality code. In order to ensure minimum standards in terms of code quality, code coverage is set to be greater than 90\% and some Python quality and style tools that comply with PEP8 standards are used, such as flake8, pylint, black and mypy.

Open source. In addition to the source code being available on GitHub\textsuperscript{2}, Frouros package can be installed through the Python Package Index (PyPI)\textsuperscript{3}. In terms of licensing, it is distributed under the BSD-3-Clause license.

5. Comparison to related software

With regard to the concept drift detection part, MOA Bifet et al. (2010) has most of the supervised methods that we are including, but they are implemented in Java. In Python, River Montiel et al. (2021), that is focused on online machine learning and streaming data, offers some supervised algorithms but only a subset of those presented here. Another Python library that contains this type of detectors is scikit-multiflow Montiel et al. (2018), but it has not been taken into account due to the fact that it was merged with the online machine learning library Creme Halford et al. (2019), resulting in the above-mentioned River library.

For the data drift part, Alibi-detect Van Looveren et al. (2019) has several algorithms related to the field of statistical two-sample hypothesis testing and some of them can act both online (single sample) and offline (batch sample). TorchDrift Viehmann et al. (2021) also implements some statistical two-sample hypothesis testing methods but in this case uses PyTorch for their implementation.

To the best of our knowledge, Menelaus Leigh et al. (2022) is the only open source library that has both supervised and unsupervised methods, as well as a semi-supervised algorithm, although they classify them in the following types: change detection, concept drift and data drift. Supervised algorithms are implemented in such a way that the user must necessarily be in charge of controlling each iteration of the sample without offering some helper functions or classes to interact with the detector, as Frouros does and it is explained in Section 3.1.

Table 1 provides a more detailed view of the methods implemented in each of the libraries mentioned above, as well as those included in Frouros. At the time of writing this paper, Frouros is listed as the library with the highest number of methods available with 24.

Moreover, there are several other libraries and tools that have been excluded from Table 1, due to the fact that they implement a more limited number of methods, or that are more focused on building graphical dashboard and visual representations, such as Deepchecks Bressler et al. (2022), Eurybia Roux et al. (2022), Evidently Dral (2020) or NannyML Nuyttens and Perrakis (2022).

\textsuperscript{1} https://frouros.readthedocs.io
\textsuperscript{2} https://github.com/IFCA/frouros
\textsuperscript{3} https://pypi.org/project/frouros/
| Method              | Allhi-detect | Menelaus | MOA | River | TorchDrift | Frouros | Reference                                      |
|---------------------|--------------|----------|-----|-------|------------|---------|-----------------------------------------------|
| **Supervised**      |              |          |     |       |            |         |                                               |
| ADWIN               | ✓            | ✓        | ✓   |       | ✓          | ✓       | Bifet and Gavalda (2007)                     |
| CUSUM               | ✓            | ✓        | ✓   |       | ✓          | ✓       | Page (1954)                                  |
| DDM                 | ✓            | ✓        | ✓   |       | ✓          | ✓       | Gama et al. (2004)                           |
| ECDD-WT             | ✓            | ✓        | ✓   |       | ✓          | ✓       | Ross et al. (2012)                           |
| EDDM                | ✓            | ✓        | ✓   |       | ✓          | ✓       | Baena-Garcia et al. (2006)                   |
| GMA                 | ✓            |          | ✓   |       | ✓          | ✓       | Roberts (1959)                               |
| HDDM-A              | ✓            | ✓        |     |       | ✓          | ✓       | Frias-Blanco et al. (2014)                   |
| HDDM-W              | ✓            | ✓        | ✓   |       |            |         | Frias-Blanco et al. (2014)                   |
| KSWIN               | ✓            |          | ✓   |       |            | ✓       | Raab et al. (2020)                           |
| LFR                 | ✓            |          | ✓   |       |            |         | Wang and Abraham (2015)                      |
| Page Hinkley        | ✓            | ✓        | ✓   | ✓     |            |         | Page (1954)                                  |
| RDDM                | ✓            |          | ✓   | ✓     |            | ✓       | Barros et al. (2017)                         |
| SEED                | ✓            |          | ✓   | ✓     |            |         | Huang et al. (2014)                          |
| SeqDrift1           | ✓            |          | ✓   |       |            |         | Sakthithasan et al. (2013)                   |
| SeqDrift2           | ✓            |          | ✓   | ✓     |            | ✓       | Pears et al. (2014)                          |
| STEPD               | ✓            | ✓        | ✓   | ✓     |            | ✓       | Nishida and Yamauchi (2007)                  |
| **Semi-supervised** |              |          |     |       |            |         |                                               |
| MD3-RS              | ✓            |          | ✓   |       |            | ✓       | Sethi and Kantardzic (2017)                  |
| MD3-SVM             | ✓            |          | ✓   |       |            | ✓       | Sethi and Kantardzic (2017)                  |
| **Unsupervised**    |              |          |     |       |            |         |                                               |
| C2ST                | ✓            |          | ✓   |       |            | ✓       | Lopez-Paz and Oquab (2016)                   |
| CDBD                | ✓            |          | ✓   |       |            | ✓       | Lindstrom et al. (2013)                      |
| Context-aware MMD   | ✓            |          | ✓   |       |            | ✓       | Cobb and Van Looveren (2022)                 |
| CVM                 | ✓            |          | ✓   | ✓     |            |         | Cramér (1928)                                |
| EMD                 | ✓            |          | ✓   | ✓     |            |         | Rubner et al. (2000)                         |
| FET                 | ✓            |          | ✓   | ✓     |            |         | Upton (1992)                                 |
| HDDDM               | ✓            |          | ✓   | ✓     |            |         | Ditzler and Polikar (2011)                   |
| Histogram Intersection |          |          | ✓   | ✓     |            |         | Swain and Ballard (1991)                    |
| JS Divergence       | ✓            |          | ✓   | ✓     |            |         | Lin (1991)                                   |
| kdq-Tree            | ✓            |          | ✓   | ✓     |            |         | Dasu et al. (2006)                           |
| KL Divergence       | ✓            |          | ✓   | ✓     |            |         | Kullback and Leibler (1951)                  |
| KS                  | ✓            | ✓        | ✓   | ✓     |            | ✓       | Massey Jr (1951)                             |
| Learned Kernel MMD  | ✓            |          | ✓   |       |            | ✓       | Liu et al. (2020)                            |
| LSDD                | ✓            |          | ✓   |       |            | ✓       | Bu et al. (2016)                             |
| MMD                 | ✓            |          | ✓   | ✓     |            | ✓       | Gretton et al. (2012)                        |
| Partial MMD         | ✓            |          | ✓   | ✓     |            | ✓       | Viehmann (2021)                              |
| PCA-CD              | ✓            |          |     | ✓     |            | ✓       | Qahtan et al. (2015)                         |
| PSI                 | ✓            |          | ✓   | ✓     |            | ✓       | Wu and Olson (2010)                          |
| Welch’s t-test      | ✓            |          | ✓   | ✓     |            | ✓       | Welch (1947)                                 |
| χ²                  | ✓            |          | ✓   | ✓     |            | ✓       | Pearson (1900)                               |
| **# Methods**       | 9 12 14 7 4 24 |          |     |       |            |         |                                               |

Table 1: Drift detection methods by library
6. Conclusion and future work

This paper presents Frouros, a Python library for drift detection in machine learning problems that can be used with scikit-learn library, both for algorithms that aim to detect concept drift and for those that try to detect data drift. Moreover, this library tries to meet some of the open source software development standards that allow to extend it, both in terms of adding new methods or modes to interact with the detectors as helper functions or classes. In view of future work, we plan to adapt supervised methods to support batch-incremental learning, as long as the nature of each algorithm allows it. We will also consider to extend the unsupervised part to include some methods that work with individual instances (online) and not only in batch mode (offline). Finally, adding new modes, as described in Section 3.1, to interact with the detectors would make it possible to adapt the library to handle more real-world use cases.

Acknowledgments

The authors acknowledge the funding from the Agencia Estatal de Investigación, Unidad de Excelencia María de Maeztu, ref. MDM-2017-0765.
References

Samuel Ackerman, Orna Raz, Marcel Zalmanovici, and Aviad Zlotnick. Automatically detecting data drift in machine learning classifiers. *arXiv preprint arXiv:2111.05672*, 2021.

Manuel Baena-Garcia, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavalda, and Rafael Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86, 2006.

Roberto SM Barros, Danilo RL Cabral, Paulo M Gonçalves Jr, and Silas GTC Santos. Rddm: Reactive drift detection method. *Expert Systems with Applications*, 90:344–355, 2017.

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

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, and Thomas Seidl. Moa: Massive online analysis, a framework for stream classification and clustering. In *Proceedings of the first workshop on applications of pattern analysis*, pages 44–50. PMLR, 2010.

N. Bressler, I. Gabbay, S. Chorev, P. Tannor, M. Perlmutter, N. Hutnik, J. Liberman, D. Ben Israel, and Y. Romanyshyn. Deepchecks: Comprehensive testing and validation of ml models and data, 2022. URL https://www.deepchecks.com/. Software available from https://github.com/deepchecks/deepchecks.

Li Bu, Cesare Alippi, and Dongbin Zhao. A pdf-free change detection test based on density difference estimation. *IEEE transactions on neural networks and learning systems*, 29(2):324–334, 2016.

Oliver Cobb and Arnaud Van Looveren. Context-aware drift detection. In *International Conference on Machine Learning*, pages 4087–4111. PMLR, 2022.

Harald Cramér. On the composition of elementary errors: First paper: Mathematical deductions. *Scandinavian Actuarial Journal*, 1928(1):13–74, 1928.

Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006.

Gregory Ditzler and Robi Polikar. Hellinger distance based drift detection for nonstationary environments. In *2011 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE)*, pages 41–48. IEEE, 2011.

Emeli Dral. An open-source framework to evaluate, test and monitor ml models in production. https://github.com/evidentlyai/evidently, 2020. Online; accessed 14-8-2022.
Isvani Frias-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jimenez, Rafael Morales-Bueno, Agustín Ortiz-Díaz, and Yaile Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding’s bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2014.

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

João Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine learning*, 90(3):317–346, 2013.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

Igor Goldenberg and Geoffrey I Webb. Survey of distance measures for quantifying concept drift and shift in numeric data. *Knowledge and Information Systems*, 60(2):591–615, 2019.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

Max Halford, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, and Adil Zouitine. creme, a Python library for online machine learning, 2019. URL https://github.com/MaxHalford/creme

David Tse Jung Huang, Yen Sing Koh, Gillian Dobbie, and Russel Pears. Detecting volatility shift in data streams. In *2014 IEEE International Conference on Data Mining*, pages 863–868. IEEE, 2014.

Imen Khamassi, Moamar Sayed-Mouchaweh, Moez Hammami, and Khaled Ghédira. Discussion and review on evolving data streams and concept drift adapting. *Evolving systems*, 9(1):1–23, 2018.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Nicholl Leigh, Thomas Schill, India Lindsay, Anmol Srivastava, Kodie P. McNamara, and Shashank Jarmale. Menelaus. https://github.com/mitre/menelaus, 2022. Online; accessed 14-8-2022.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

Patrick Lindstrom, Brian Mac Namee, and Sarah Jane Delany. Drift detection using uncertainty distribution divergence. *Evolving Systems*, 4(1):13–25, 2013.
Frouros: A Python library for drift detection in Machine Learning problems

Feng Liu, Wenkai Xu, Jie Lu, Guangquan Zhang, Arthur Gretton, and Danica J Sutherland. Learning deep kernels for non-parametric two-sample tests. In International conference on machine learning, pages 6316–6326. PMLR, 2020.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. arXiv preprint arXiv:1610.06545, 2016.

Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. Journal of the American statistical Association, 46(253):68–78, 1951.

Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. Scikit-multiflow: A multi-output streaming framework. Journal of Machine Learning Research, 19(72):1–5, 2018. URL http://jmlr.org/papers/v19/18-251.html.

Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourt, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River: machine learning for streaming data in python. Journal of Machine Learning Research, 22(110):1–8, 2021. URL http://jmlr.org/papers/v22/20-1380.html.

Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. Pattern recognition, 45(1):521–530, 2012.

Kyosuke Nishida and Koichiro Yamauchi. Detecting concept drift using statistical testing. In International conference on discovery science, pages 264–269. Springer, 2007.

Niels Nuyttens and Nikolaos Perrakis. Nannyml. https://github.com/NannyML/nannyml, 2022. Online; accessed 14-8-2022.

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

Russel Pears, Sripirakas Sakthithasan, and Yun Sing Koh. Detecting concept change in dynamic data streams. Machine Learning, 97(3):259–293, 2014.

Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 50(302):157–175, 1900.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

Abdulhakim A Qahtan, Basma Alharbi, Suojin Wang, and Xiangliang Zhang. A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 935–944, 2015.
Christoph Raab, Moritz Heusinger, and Frank-Michael Schleif. Reactive soft prototype computing for concept drift streams. *Neurocomputing*, 416:340–351, 2020.

Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019.

Sergio Ramírez-Gallego, Bartosz Krawczyk, Salvador García, Michał Woźniak, and Francisco Herrera. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39–57, 2017.

S. W. Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 1(3):239–250, 1959. ISSN 00401706. URL [http://www.jstor.org/stable/1266443](http://www.jstor.org/stable/1266443).

Gordon J Ross, Niall M Adams, Dimitris K Tasoulis, and David J Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern recognition letters*, 33(2):191–198, 2012.

Nicolas Roux, Johann Martin, and Thomas Bouché. Eurybia. [https://github.com/MAIF/eurybia](https://github.com/MAIF/eurybia), 2022. Online; accessed 14-8-2022.

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover’s distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

Sripirakas Sakthithasan, Russel Pears, and Yun Sing Koh. One pass concept change detection for data streams. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 461–472. Springer, 2013.

Tegjyot Singh Sethi and Mehmed Kantardzic. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82:77–99, 2017. ISSN 0957-4174. doi: [https://doi.org/10.1016/j.eswa.2017.04.008](https://doi.org/10.1016/j.eswa.2017.04.008). URL [https://www.sciencedirect.com/science/article/pii/S0957417417302439](https://www.sciencedirect.com/science/article/pii/S0957417417302439).

Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991.

Graham JG Upton. Fisher’s exact test. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 155(3):395–402, 1992.

Arnaud Van Looveren, Janis Klaise, Giovanni Vacanti, Oliver Cobb, Ashley Scillitoe, Robert Samoilescu, and Alex Athorne. Alibi detect: Algorithms for outlier, adversarial and drift detection, 2019. URL [https://github.com/SeldonIO/alibi-detect](https://github.com/SeldonIO/alibi-detect).

Thomas Viehmann. Partial wasserstein and maximum mean discrepancy distances for bridging the gap between outlier detection and drift detection. *arXiv preprint arXiv:2106.12893*, 2021.

Thomas Viehmann, Luca Antiga, Daniele Cortinovis, and Lisa Lozza. Torchdrift. [https://github.com/torchdrift/torchdrift](https://github.com/torchdrift/torchdrift), 2021. Online; accessed 14-8-2022.
Heng Wang and Zubin Abraham. Concept drift detection for streaming data. In 2015 international joint conference on neural networks (IJCNN), pages 1–9. IEEE, 2015.

Bernard L Welch. The generalization of ‘student’s’ problem when several different population variances are involved. Biometrika, 34(1-2):28–35, 1947.

Desheng Wu and David L Olson. Enterprise risk management: coping with model risk in a large bank. Journal of the Operational Research Society, 61(2):179–190, 2010.

Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. Big data analysis: new algorithms for a new society, pages 91–114, 2016.