MSc Artificial Intelligence
Master Thesis

Drift detection on embeddings

by
Theofilos Papapanagiotou
11903406

June 30, 2022

42 Credits
November 2021 - June 2022

Supervisor:
dr. ing. Sebastian Schelter

Examiner:
dr. ing. Sebastian Schelter

Second reader:
ing. Barrie Kersbergen

Universiteit van Amsterdam

Contents

Abstract

Language models produce multidimensional vectors, but the context of the embedding is not known. Storing the embeddings in a platform also requires advanced monitoring capabilities to detect data shifts. Drift detection in the domain of contextual embeddings is a challenging task, as it requires a large number of data points to be compared. This thesis is an empirical study of the problem of detecting data shifts in the domain of contextual embeddings. We see that a pre-trained classifier for dimensionality reduction can be used to detect data shifts, and performs better than traditional methods like MMD, KS and LSDD. We also analyse the problem of scaling such a solution and discuss the implications of implementing such a system in a ML platform.

Chapter 1

Introduction

Machine learning systems are used in the industry to learn users' dietary habits, entertainment preferences, financial behaviour, etc. By doing so, they power the most innovative products which take advantage of that rich, ever-changing information. The evolution of data processing in order to better encode the underlying information, is a key driver for the success of these systems. The transformation of the data to features is a key part of the modeling process. But such models, are not always robust to the presence of distribution shift in the data.

Storing the features in a dedicated system such as a feature store, helps decoupling the generation of the features from the consumption [19]. Data analysts with domain expertise generate features from the data. Data scientists consume these features to build models, focusing on the modeling activity.

Today, language models (LMs) and large language models (LLMs) are used to extract embeddings from text data. They are used by downstream tasks to perform prediction actions on new data. The most common activity over these embeddings is the similarity search, to find the nearest neighbors of a given embedding. The utilization of an embeddings store next to a feature store brings a significant advantage to development process [21]. By using a vector database as an embedding store, we can avoid the need to load the full set of embeddings into memory. Essentially we offload the task of the search to a dedicated system [30] [31].

Continuous model training pipelines make use of signals observed during inference time, to trigger the retraining of models [1]. Such signals are produced by monitoring techniques [22] that understand the changes in the data quality [26] and eventually the degradation of the model's performance. Drift detection is a common technique, used to detect the presence of distribution shift in the data.

Common drift detection methods rely on dimensionality reduction techniques on the features of the data, seeing success in the modalities of image and tabular data. In the case of text data though detecting shift in the words or the sentences is not effective, as they do not represent the semantics of the input [17]. The introduction of LMs and LLMs brings the heavy use of the contextual embdeddings. These embeddings are vectors of floats, extracted by the last layers of pre-trained models, sometimes fine-tuned to improve the performance of the model on a particular domain. The detection takes place on the embeddings, and not on the raw input data.

In this thesis, we study empirically the use of drift detectors on such contextual embeddings, extracted by modern LMs. Reviewing the literature, we see that there has been significant studies [24] in other modalities. We also see that detecting distribution shifts in embeddings is not a popular topic in the literature. We explore the mechanics of shift detectors such as Maximum Mean Discrepancy (MMD), Kolmogorov-Smirnov (KS), Least Squares Density Difference (LSDD) and a pre-trained classifier (DistilBERT). In the experiment section we evaluate the performance of these methods on the embeddings of the text data, measured

on the original and drifted test sets. We use the notion of distance as well as the runtime to discuss the results of the experiment and their applicability to production use-cases. We see that dimensionality reduction with subsequent two-sample testing, which is a common preprocessing task on the other modalities, is not effective the case of embeddings of text data. We also see that the use of a pre-trained classifier is not efficient for online drift detection.

Chapter 2

Related work

Tsymbal et al [28] describes that two kinds of concept drift that may occur in the real world are normally distinguished in the literature as (1) sudden (abrupt, instantaneous) and (2) gradual concept drift. Lu et al [18] describe a general framework for concept drift detection. Gama et al [11] introduce the concept drift adaptation and present the state of the art techniques and a collection of benchmarks.

Investigation of methods for detecting dataset shift on the modality of computer vision show that the best performance is achieved by the two-sample testing with pre-trained classifiers for dimensionality reduction [24]. Feldhans et al [10] empirically compare drift detectors on document embeddings on two benchmarking datasets with varying amounts of drift. Their results show that multivariate drift detectors based on the KTS Test and LSDD outperform univariate drift detectors based on the KS Test.

Douadi et al [3] compare different types of dissimilarity estimators and metrics theoretically and empirically and investigate the relevance of the single metric components. Kubler et al [15] uses an AutoML two-sample test and achieves competitive performance on a diverse distribution shift benchmark as well as on challenging two-sample testing problems. Zhao et al [32] proposes a general framework that generalizes the JensenShannon divergence and the maximum mean discrepancy family. Cobb and Looveren explore context-aware drift detection [5] and additionally contribute the alibi-detect library [29]

Chapter 3

Drift detection techniques

To formalize the problem, our training dataset is processed through the pretrained model and the extracted embeddings form the dataset x with a distribution $p(x)$. Our test dataset is processed with the same way and produce the dataset x' with a distribution $q(x')$. We want to determine if $p(x) = q(x')$. In hypothesis testing, the null hypothesis would be $H_0 : p(x) = q(x')$ and the alternate hypothesis $H_A : p(x) \neq q(x')$. Unlike other experiments [24], the datasets x and x' are high-dimensional latent space representations of the text data.

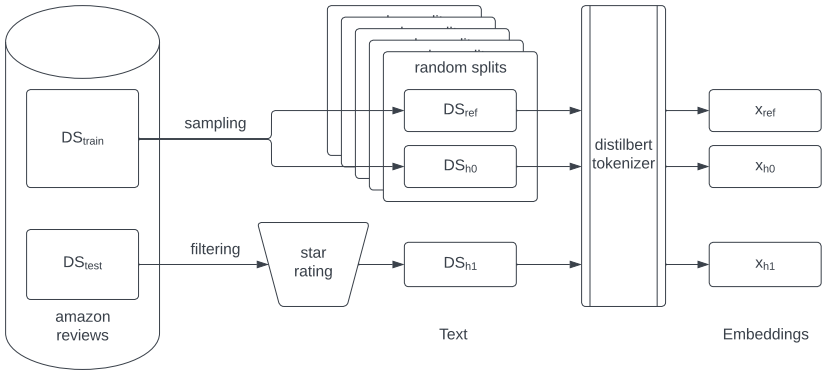

Figure 3.1: Dataset split into reference and tests subsets before the detection of drift

As we randomly sample DS_{ref} and DS_{h0} from the dataset and subsequently generate the embedings x_{ref} and x_{h0}, we do not expect them to have an identical distribution, but close enough to be used for the null hypothesis test. On the other hand, we induce drift in the DS_{h1} dataset by filtering from the test subset the text of a particular use behaviour, reflected into the star rating feature of the dataset. The embeddings x_{h1} should be further away in terms of distance from the embeddings of x_{ref} and x_{h0} and therefore be used for the alternate hypothesis test.

3.1 Kolmogorov-Smirnov (KS) Test

The Kolmogorov-Smirnov (KS) test is a non-parametric test for the null hypothesis that two distributions are identical. The aim of this test is to compare the underlying continuous distributions of two independent samples. As it is a univariate test, we can only compare two samples against each other. From the comparison, we get a vector of $p-values$ for each sample and we can then use the Bonferroni correction to obtain one p-value. The Bonferroni correction is a conservative method of correcting for multiple hypothesis testing [2]. It is used to reject

the null hypothesis if the minimum p-value among all tests is less than α/K (where α is the significance level of the test).

The time it takes to run the Kolmogorov-Smirnov test is proportional to the number of samples. The complexity of the algorithm is $O(n)$ where n is the sample size. This two-sided test attempts to compute the exact 2-sample probability, because our sample sizes are relatively small, below 10.000 samples. [13]. This comparison takes place only in the CPU memory.

3.2 Least-Squares Density Difference (LSDD)

The LSDD detector computes the p-value of the permutation test by using the least-squares density difference as a distance measure between the reference and test data. The time it takes to detect drift is also proportional to the number of samples. The formal definition of LSDD between two distributions p and q on X is:

$$LSDD(p, q) = \int_X (p(x) - q(x))^2 dx$$

It is using the Gaussian RBF kernel to compute the distance between the reference and test data:

$$k(x, x') = \exp(\frac{-(x - x')^2}{2\gamma^2})$$

where γ is the length scale of the kernel. That kernel is using the GPU memory.

3.3 Maximum Mean Discrepancy (MMD)

The MMD detector computes the p-value of the permutation test by using the maximum mean discrepancy as a distance measure between the reference and test data. It is a kernel-based technique for multivariate two-sample test. The time it takes to detect drift is also proportional to the number of samples. The formal definition of MMD between samples of two distributions is:

$$MMD^1 = \frac{1}{m^2 - m}\Sigma_{i=1}^m\Sigma_{j\neq 1}^m k(x_i, x_j) + \frac{1}{n^2 - n}\Sigma_{i=1}^m\Sigma_{j\neq 1}^m k(x'_i, x'_j) - \frac{2}{mn}\Sigma_{i=1}^m\Sigma_{j\neq 1}^m k(x_i, x'_j)$$

where k is the Gaussian RBF kernel we discussed earlier. So it is a two step process, first we compute the kernel matrix, then we compute the MMD. The comparison takes place in the GPU memory.

3.4 Classifier

Following the example of Rabanser et al [24] on the image dataset, we are also trying the classifier method [16]. The output of the classifier is the probability that an input instance belongs to the test set. The notion of distance here is calculated using a KS test on the prediction probabilities between $x_ref - x_h0$ and $x_ref - x_{h1}$. Any binary classification model can be used for this purpose. In our experiment section we discuss the use of the DistilBERT model [25], as a pretrained transformer that we fine-tune with our dataset split into the two classes x_{ref} and x_{h0} or x_{h1}.

Chapter 4

Experiments

4.1 Method

Having fixed size of the reference data h_{ref} at 10k samples, we measure the quality and the runtime of the detectors in different test data h_0 and h_1 sample sizes between 1k and 10k with a step of 1k.

4.2 Setup

The experiments were carried out in a machine with an Intel 12900k CPU, 64GB of DDR5 RAM and one Nvidia RTX3090 24GB GPU. The software used was Python3, PyTorch, Nvidia CUDA/CUDNN, Seldon Alibi-Detect, WILDS, PyArrow, NumPy, Pandas, SciPy, Sacred, PyMongo, TensorFlow2, HuggingFace Transformers and Datasets. The code of the experiments is available at `https://github.com/theofpa/embeddings-store-thesis`.

We have used the Amazon product reviews dataset. [1] The dataset contains 130M+ customer reviews from the Amazon.com marketplace, modified by Ni et al. [20] It has been loaded using the WILDS [14] library. The input x of the dataset is the review text and the label y is the corresponding 1-to-5 star rating. The training and test sets comprise reviews from disjoint sets of users, as follow: The training set has 1 million reviews written before 2013. The test set has 20 thousand reviews written after 2014. This out-of-distribution test dataset is used to evaluate the performance of our methods, as shown in 3.1. We sample the training dataset to generate x_{ref}, as well as the unshifted x_{h0}. We sample the test dataset to generate the drifted x_{h1}.

We sample using random indices of the dataset, seeded by consistent numbers across the experiments using the scikit-learn library [23]. The reported values for the performance of the methods 5.1 is the average performance over 5 random splits. The values of runtime in 5.2 are the average runtime of 10 values, (5 random splits for each of x_{h0} and x_{h1}).

4.3 Implementation

For the KS detector, we've used the implementation of the two-sample Kolmogorov-Smirnov test of the Scipy library [27], following the implementation of the alibi-detect library [29].

For the MMD and the LSDD detector, we've used the Tensorflow [7] implementation of the Gaussial RBF kernel, provided by alibi-detect.

[1] https://s3.amazonaws.com/amazon-reviews-pds/readme.html

For the Classifier, we've loaded the $distilbert-base-uncased$ model using the HuggingFace library [9]. The training and the freezing of the model is done using the TensorFlow library, following the example of alibi-detect.

For all experiments, we fine tuned the BERT-base-cased model using the implementation of Devlin [8]. The pre-trained model was loaded using the Transformer library from HuggingFace.

4.4 Workflow

The execution workflow has the following components:

- A scheduler, given a suite of the experiments to be performed, schedules their execution.
- A worker, receives the execution instructions as configuration parameters, executes the experiment and returns the results.
- A metrics database, that stores the results alongside the configuration metadata of the experiments.
- A reporting tool, that visualises the results of the experiments.

4.5 Reporting

The metrics library used is IDSIA's Sacred [12]. The database which receives the configuration parameters, the results and the metrics described below, is a cloud instance of Mongodb. The configuration parameters of the runs and the reported metrics are stored in two separate collections, due to the nature of their one-to-many relationship. The library used for the reporting tool is Incense [4], that loads the configuration and metrics in a pandas dataframe and visualises them using matplotlib.

Chapter 5

Discussion

The measurement of the performance of the methods is based on the distance between the reference and the test distributions as well as the runtime.

5.1 Quality

We use the notion of distance to measure the effectiveness of the drift detection. That is the distance between $p(x)$ and $q(x')$, so between the embeddings x_{ref} and the embeddings x_{h0} or x_{h1}.

The p-value of the permutation test of the above, is only applicable to the multivariate tests. In the case of multiple univariate we use the bonferroni correction, to get the p-value with the smallest size from the returned vector of p-values.

detector	Classifier		KS		LSDD		MMD	
test_set	h_0	h_1	h_0	h_1	h_0	h_1	h_0	h_1
h_size								
1000	0.01554	0.10278	0.02793	0.05018	0.00047	0.00466	0.00010	0.00437
2000	0.01198	0.10852	0.02075	0.04680	0.00026	0.00482	0.00003	0.00468
3000	0.00995	0.10024	0.01798	0.04392	0.00022	0.00433	0.00000	0.00423
4000	0.01087	0.12723	0.01605	0.04271	0.00015	0.00391	0.00001	0.00420
5000	0.00782	0.11456	0.01482	0.04227	0.00013	0.00401	0.00001	0.00412
6000	0.00768	0.12917	0.01399	0.04132	0.00012	0.00441	0.00001	0.00366
7000	0.00929	0.11459	0.01339	0.04145	0.00011	0.00415	0.00000	0.00391
8000	0.00659	0.11921	0.01264	0.04096	0.00009	0.00392	0.00001	0.00406
9000	0.00607	0.12904	0.01223	0.04086	0.00008	0.00386	0.00002	0.00402
10000	0.00730	0.12758	0.01190	0.04062	0.00009	0.00444	0.00001	0.00404

Table 5.1: Performance of the 4 detectors on different sizes of the test set, without (h_0) and with drift (h_1). Average values from 5 runs with different seeds.

5.1.1 Observations in the quality metric

- The Classifier is the best drift detector in terms of distance between the two cases (h_0-h_{ref} and h_1-h_{ref}).
- MMD is more effective than the other three detectors, into detecting the original distribution (h_0) being part of the same distribution as the reference.
- KS, LSDD and MMD detect similar range of distance between the drifted (h_1) and the reference distribution (h_{ref}).
- All 4 detectors clearly separate the drift in the drifted test set (h_1) as well as detect the original distribution (h_0).
- The size of the sampling (in the ranges 1k-10k that we've tested) does not significantly affect the performance of the detectors.

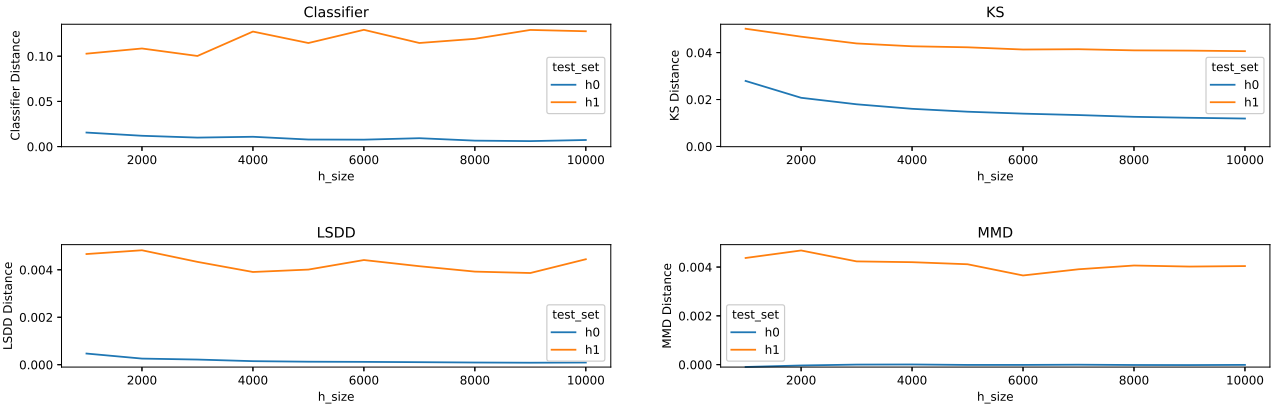

Figure 5.1: Performance of the 4 detectors on different sizes of the test set, without (blue h_0) and with drift (orange h_1).

5.2 Scalability

We use the runtime to discuss what challenges we'll face when we will try to implement the methods in a production ML platform. The runtime in our experiments is the time it takes to detect drift in different test data sizes.

- It includes the distance calculation between $p(x)$ and $q(x')$.
- It includes the tokenization of DS_{test} to x_{h0} or x_{h1}.
- The size of the reference distribution x_{ref} remains 10.000 samples across all runs.
- It does not include the loading of the LMs to the GPU memory.
- It does not include the loading of the text dataset to the CPU memory, nor the copy of the tensors to the GPU memory.
- In the case of KS, MMD and LSDD it does not include the time to tokenize the DS_{train} to x_{ref}.

- In the case of the Classifier, the generation of x_{ref} is included, as we pass both datasets to the training process.
- The values shown are the average of the runtimes of 10 data points: 5 random splits for each of the undrifted h_0 and the drifted test h_1.

The selection of the test size is a key challenge in the detection of sequential changes, due to the false positive which can occur in the absence of change and the quadratic time of the MMD detector. Cobb et al [6] propose a method for setting time-varying thresholds that allows a desired expected runtime to be accurately targeted whilst additionally keeping the false positive rate constant across time steps.

detector	Classifier	KS	LSDD	MMD
h_size				
1000	146.5	13.1	13.1	25.6
2000	165.0	16.6	14.6	31.2
3000	177.0	16.9	15.0	33.9
4000	184.4	16.6	18.3	38.9
5000	198.7	19.5	17.2	42.9
6000	212.8	19.0	20.4	47.6
7000	231.9	21.5	19.7	51.7
8000	243.3	21.5	21.6	58.9
9000	254.7	25.4	22.1	62.8
10000	264.4	25.4	22.9	69.9

Table 5.2: Average runtime in seconds of the 4 detectors on different sizes of the test sets.

5.2.1 Observations in the runtime

- All 4 methods have their runtime changing proportionally to the sample size of the test set.
- The Classifier has a penalty of time due to the nature of the algorithm: The reference set is given as input to a training process during the detection time, something that is not the case on the other methods.
- KS, LSDD and Classifier have a linear time computation complexity.
- MDD has log linear computation complexity, with worst case being quadratic due to the squared exponential kernel of the Gaussian RBF of the MMD^2.

When a classifier is trained, we realize that we need much more processing resources than a typical inference service platform. In the example of an inference serving platform, we have an ecosystem of peripheral services that are deployed around the model server, like the pre and postprocessing transformers, the explainers, the fairness detectors, the adversarial robustness testers, the logging and monitoring stack. All these services are lightweight components compared to the core model server which usually requires a GPU instance to perform what it is

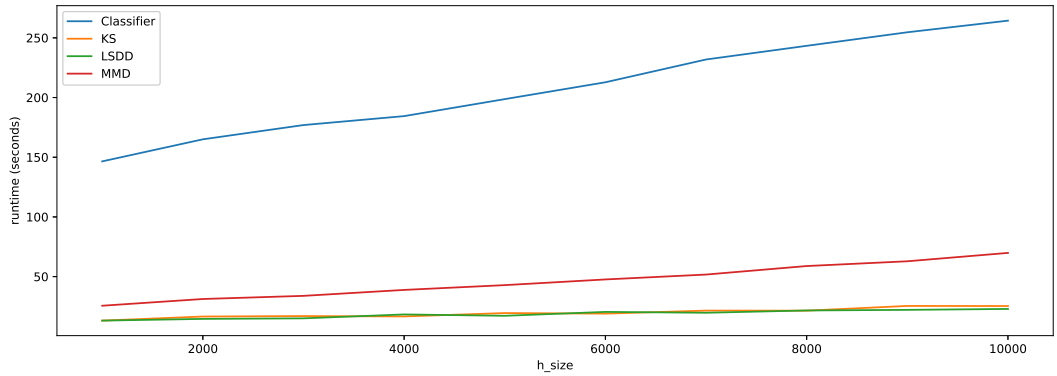

Figure 5.2: Average runtime of the 4 detectors on different sample size of the test set.

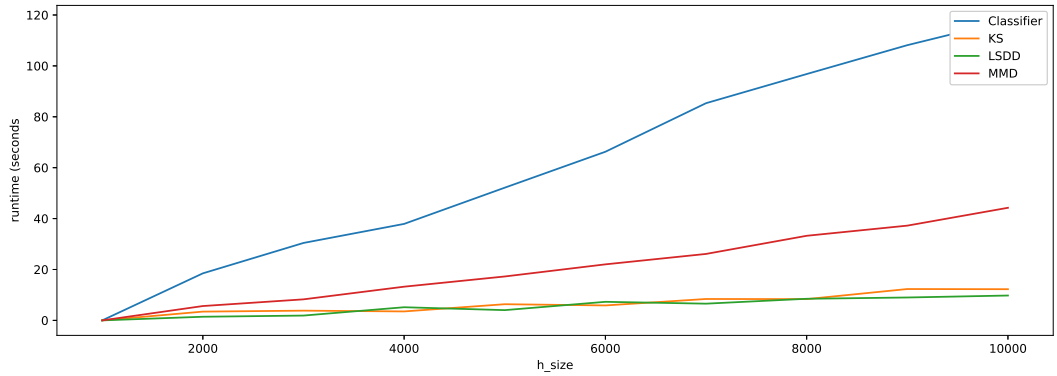

Figure 5.3: Average runtime of the 4 detectors, positioned at the same starting point by subtracting the runtime of the first observation.

meant to: the matrix multiplications between the weights of the model and the input vector of the inference request. With the introduction of a drift detector, we are introducing two components: the drift detector itself and the epheneral storage for the test set.

The GPU instance type of the model server is optimized for inference, with accelerators like the AWS Elastic Inference which allocate a fraction of the VRAM to the model server. The drift detector itself though, would require a GPU instance optimized for training, as it would need a larger amount of VRAM compared to the inference optimized resource allocation, to load the h_{ref} and h_0 or h_1 matrices.

On the topic of the ephemeral storage, the drift detector as a microservice requires as input a test dataset with multiple instances of incoming inference requests, unlike the explainers and transformers which consume single instances of inference requests from an event-based stream. To achieve that, a model serving platform must be able to queue such a bucket of requests and bundle them to make them available to the detector. The batch size of the test set is a critical parameter for the allocation of the resources to store such data. Having a distributed streaming processing system would allow the detector to scale out to parallel, even overlapping tests, based on the batch size number or periods of time. Such as system would handle the offsets of the detectors as consumers and essentially manage the scheduling of the tests.

Traditionally, a training pipeline delivers a model which is then loaded for inference in the model server. In the case of the drift detector we have a second model (the detector's Classifier), sitting next to the model which performs the predictions for our product. It also means that we have a continuous training pipeline, which also refreshes the x_{ref} periodically or based on

a trigger. The traditional data pipeline scheduler are not the best options for such a scenario, and a ml pipeline solution would be optimal [1] for data-driven execution of the pipeline.

Chapter 6

Conclusion

The Classifier method outperforms MMD and other baseline methods like KS and LSDD. It also makes easier the frequently change of h_{ref} as it's loading it every time together with the h_{ref}. It comes with a penalty of extra time required for training the classifier with the reference and test dataset.

Implementing a monitoring solution for a ML platform in the space of drift detection brings two important challenges: the selection of the test set size and the allocation of computation resources. The test size can be empirically defined based on the number of samples or a time period. Recent research [6] [15] is trying to automate such detection configuration. The scaling of the resources for the drift detector should be handled by a platform that is capable of managing the batches of the inference requests in a data-driven fashion.

Bibliography

[1] Denis Baylor, Kevin Haas, Konstantinos Katsiapis, Sammy Leong, Rose Liu, Clemens Menwald, Hui Miao, Neoklis Polyzotis, Mitchell Trott, and Martin Zinkevich. "Continuous Training for Production {ML} in the TensorFlow Extended ({TFX}) Platform". In: 2019 {USENIX} Conference on Operational Machine Learning (OpML 19). 2019, pp. 51–53. ISBN: 978-1-939133-00-7. URL: https://www.usenix.org/conference/opml19/presentation/baylor (visited on 11/03/2021).

[2] J M. Bland and D. G Altman. "Statistics Notes: Multiple Significance Tests: The Bonferroni Method". In: *BMJ* 310.6973 (Jan. 21, 1995), pp. 170–170. ISSN: 0959-8138, 1468-5833. DOI: 10.1136/bmj.310.6973.170. URL: https://www.bmj.com/lookup/doi/10.1136/bmj.310.6973.170 (visited on 06/26/2022).

[3] Tassadit Bouadi, Elisa Fromont, and Eyke Hüllermeier, eds. *Advances in Intelligent Data Analysis XX: 20th International Symposium on Intelligent Data Analysis, IDA 2022, Rennes, France, April 20–22, 2022, Proceedings*. Vol. 13205. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022. ISBN: 978-3-031-01332-4 978-3-031-01333-1. DOI: 10.1007/978-3-031-01333-1. URL: https://link.springer.com/10.1007/978-3-031-01333-1 (visited on 06/26/2022).

[4] Rüdiger Busche. *JarnoRFB/Incense*. June 15, 2022. URL: https://github.com/JarnoRFB/incense (visited on 06/27/2022).

[5] Oliver Cobb and Arnaud Van Looveren. *Context-Aware Drift Detection*. Mar. 16, 2022. DOI: 10.48550/arXiv.2203.08644. arXiv: 2203.08644 [cs, stat]. URL: http://arxiv.org/abs/2203.08644 (visited on 06/26/2022).

[6] Oliver Cobb, Arnaud Van Looveren, and Janis Klaise. "Sequential Multivariate Change Detection with Calibrated and Memoryless False Detection Rates". Aug. 2, 2021. arXiv: 2108.00883 [cs, stat]. URL: http://arxiv.org/abs/2108.00883 (visited on 02/11/2022).

[7] TensorFlow Developers. *TensorFlow*. Zenodo, May 23, 2022. DOI: 10.5281/zenodo.6574269. URL: https://zenodo.org/record/6574269 (visited on 06/27/2022).

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. May 24, 2019. arXiv: 1810.04805 [cs]. URL: http://arxiv.org/abs/1810.04805 (visited on 06/24/2022).

[9] *Distilbert-Base-Uncased · Hugging Face*. URL: https://huggingface.co/distilbert-base-uncased (visited on 06/27/2022).

[10] Robert Feldhans, Adrian Wilke, Stefan Heindorf, Mohammad Hossein Shaker, Barbara Hammer, Axel-Cyrille Ngonga Ngomo, and Eyke Hüllermeier. "Drift Detection in Text Data with Document Embeddings". In: *Intelligent Data Engineering and Automated Learning – IDEAL 2021*. Ed. by Hujun Yin, David Camacho, Peter Tino, Richard Allmendinger, Antonio J. Tallón-Ballesteros, Ke Tang, Sung-Bae Cho, Paulo Novais, and Susana Nascimento. Vol. 13113. Lecture Notes in Computer Science. Cham: Springer Inter-

national Publishing, 2021, pp. 107–118. ISBN: 978-3-030-91607-7 978-3-030-91608-4. DOI: 10.1007/978-3-030-91608-4_11. URL: https://link.springer.com/10.1007/978-3-030-91608-4_11 (visited on 12/09/2021).

[11] GamaJoão, ŽliobaitėIndrė, BifetAlbert, PechenizkiyMykola, and BouchachiaAbdelhamid. "A Survey on Concept Drift Adaptation". In: *ACM Computing Surveys (CSUR)* (Mar. 1, 2014). DOI: 10.1145/2523813. URL: https://dl.acm.org/doi/abs/10.1145/2523813 (visited on 06/26/2022).

[12] Klaus Greff, Aaron Klein, Martin Chovanec, Frank Hutter, and Jürgen Schmidhuber. "The Sacred Infrastructure for Computational Research". In: *Proceedings of the 16th Python in Science Conference*. Ed. by Katy Huff, David Lippa, Dillon Niederhut, and M Pacer. 2017, pp. 49–56. DOI: 10.25080/shinma-7f4c6e7-008.

[13] J. L. Hodges. "The Significance Probability of the Smirnov Two-Sample Test". In: *Arkiv för Matematik* 3.5 (Jan. 1, 1958), pp. 469–486. ISSN: 1871-2487. DOI: 10.1007/BF02589501. URL: https://doi.org/10.1007/BF02589501 (visited on 06/26/2022).

[14] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. "WILDS: A Benchmark of in-the-Wild Distribution Shifts". In: (Dec. 14, 2020). URL: https://arxiv.org/abs/2012.07421v3 (visited on 12/09/2021).

[15] Jonas M. Kübler, Vincent Stimper, Simon Buchholz, Krikamol Muandet, and Bernhard Schölkopf. *AutoML Two-Sample Test*. June 17, 2022. arXiv: 2206.08843 [cs, stat]. URL: http://arxiv.org/abs/2206.08843 (visited on 06/26/2022).

[16] David Lopez-Paz and Maxime Oquab. *Revisiting Classifier Two-Sample Tests*. Mar. 13, 2018. arXiv: 1610.06545 [stat]. URL: http://arxiv.org/abs/1610.06545 (visited on 06/26/2022).

[17] Seldon Technologies Ltd. "Alibi-Detect Documentation". In: (). URL: https://docs.seldon.io/projects/alibi-detect/en/v0.9.1/.

[18] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. "Learning under Concept Drift: A Review". In: *IEEE Transactions on Knowledge and Data Engineering* (2018), pp. 1–1. ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2018.2876857. arXiv: 2004.05785 [cs, stat]. URL: http://arxiv.org/abs/2004.05785 (visited on 06/26/2022).

[19] *Machine Learning Design Patterns [Book]*. URL: https://www.oreilly.com/library/view/machine-learning-design/9781098115777/ (visited on 06/25/2022).

[20] Jianmo Ni, Jiacheng Li, and Julian McAuley. "Justifying Recommendations Using Distantly-Labeled Reviews and Fine-Grained Aspects". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019, pp. 188–197. DOI: 10.18653/v1/D19-1018. URL: https://www.aclweb.org/anthology/D19-1018 (visited on 06/24/2022).

[21] Laurel Orr, Atindriyo Sanyal, Xiao Ling, Karan Goel, and Megan Leszczynski. "Managing ML Pipelines: Feature Stores and the Coming Wave of Embedding Ecosystems". Aug. 11, 2021. arXiv: 2108.05053 [cs]. URL: http://arxiv.org/abs/2108.05053 (visited on 10/13/2021).

[22] Theofilos Papapanagiotou. *Model Monitoring*. Prosus AI Tech Blog. Dec. 31, 2021. URL: https://medium.com/prosus-ai-tech-blog/model-monitoring-1849fb3afc1e (visited on 06/25/2022).

[23] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[24] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/846c260d715e5b854ffad5f70a516c88-Abstract.html (visited on 11/20/2021).

[25] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. *DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter*. Feb. 29, 2020. arXiv: 1910.01108 [cs]. URL: http://arxiv.org/abs/1910.01108 (visited on 06/26/2022).

[26] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. "Automating Large-Scale Data Quality Verification". In: *Proceedings of the VLDB Endowment* 11.12 (Aug. 2018), pp. 1781–1794. ISSN: 2150-8097. DOI: 10.14778/3229863.3229867. URL: https://dl.acm.org/doi/10.14778/3229863.3229867 (visited on 06/25/2022).

[27] *Scipy.Stats.Ks_2samp — SciPy v1.8.1 Manual*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ks_2samp.html (visited on 06/27/2022).

[28] Alexey Tsymbal. "The Problem of Concept Drift: Definitions and Related Work". In: (), p. 7.

[29] Arnaud Van Looveren, Janis Klaise, Giovanni Vacanti, Oliver Cobb, Ashley Scillitoe, and Robert Samoilescu. *Alibi Detect: Algorithms for Outlier, Adversarial and Drift Detection*. Version 0.9.1. Apr. 2022. URL: https://github.com/SeldonIO/alibi-detect (visited on 06/26/2022).

[30] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. "Milvus: A Purpose-Built Vector Data Management System". In: *Proceedings of the 2021 International Conference on Management of Data*. SIGMOD/PODS '21. New York, NY, USA: Association for Computing Machinery, June 9, 2021, pp. 2614–2627. ISBN: 978-1-4503-8343-1. DOI: 10.1145/3448016.3457550. URL: https://doi.org/10.1145/3448016.3457550 (visited on 11/08/2021).

[31] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. "AnalyticDB-V: A Hybrid Analytical Engine towards Query Fusion for Structured and Unstructured Data". In: *Proceedings of the VLDB Endowment* 13.12 (Aug. 1, 2020), pp. 3152–3165. ISSN: 2150-8097. DOI: 10.14778/3415478.3415541. URL: https://doi.org/10.14778/3415478.3415541 (visited on 11/13/2021).

[32] Shengjia Zhao, Abhishek Sinha, Yutong He, Aidan Perreault, Jiaming Song, and Stefano Ermon. "Comparing Distributions by Measuring Differences That Affect Decision Making". In: International Conference on Learning Representations. Mar. 17, 2022. URL: https://openreview.net/forum?id=KB5onONJIAU (visited on 06/26/2022).