mlinspect: Lightweight Inspection of Data Preprocessing in Native Machine Learning Pipelines

Stefan Grafberger
TU Munich
stefangrafberger@gmail.com

Sebastian Schelter
AIRLab, University of Amsterdam & Ahold Delhaize
s.schelter@uva.nl

Machine Learning (ML) is increasingly used to automate decisions that impact people's lives, in domains as varied as credit and lending, medical diagnosis, and hiring. The risks and opportunities arising from the wide-spread use of predictive analytics are garnering much attention from policy makers, scientists, and the media [1].

The correctness and reliability of ML models critically depend on their training data. Pre-existing bias, such as under- or over-representation of particular groups in the training data [2], and technical bias, such as skew introduced during data preparation [3], can heavily impact performance. For example, preprocessing operations that involve filters or joins can heavily change the distribution of different groups represented in the training data [4]. Recent ML fairness research, which mostly focuses on using learning algorithms on static datasets [5] is insufficient because it cannot address the root cause of such technical bias originating from data preparation.

We need automated inspection of ML pipelines. Due to time pressure in their day-to-day activities, most data scientists will not spend the time and effort to manually instrument their code or insert logging statements for tracing as required by model management systems [6], [7]. We envision support for data scientists in the form of *automated inspections of their pipelines*, similar to the inspections used by modern IDEs to highlight potentially problematic parts of a program, such as the use of deprecated code or problematic library function calls. We furthermore argue that, to be most beneficial, automated inspections need to *work with code natively written with popular ML library abstractions*.

Lightweight pipeline inspection with `mlinspect`. In this talk, we present `mlinspect`, a library that enables lightweight lineage-based inspection of ML preprocessing pipelines written in Python. The key idea is to extract logical query plans, modeled as directed acyclic graphs (DAGs) of preprocessing operators, from ML pipelines that use popular libraries like pandas and scikit-learn, and that combine relational operations and estimator/transformer pipelines. These plans are then used to automatically instrument the code and trace the impact of operators on properties like the distribution of sensitive groups in the data. In this way, `mlinspect` empowers data scientists to automatically check their ML pipeline code for data distribution issues and provides linting for best-practices. It does not require any changes to the code. Importantly, `mlinspect` implements a library-independent interface to propagate annotations such as tuple lineage across operators from different libraries, and introduces only constant overhead per tuple flowing through the DAG. Thereby, `mlinspect` offers a general runtime for pipeline inspection, and allows us to integrate many issue detection techniques that previously required custom code, such as automated model validation on data slices [8], the identification of distortions with respect to protected group membership in the training data [4], or automated sanity checking for ML datasets [9].

Current State & Future Work. A paper describing `mlinspect` was accepted at CIDR 2021. We provide a prototypical implementation of our proposed approach at https://github.com/stefan-grafberger/mlinspect. The instrumentation of complex example pipelines using pandas and scikit-learn already works, but we do not comprehensively cover all API functions yet. We already offer implementations of representative inspections, including inspections for row-level lineage tracking, sampling of intermediate outputs, and tests for changes in the distribution of protected groups. In the future, we want to implement backends for more libraries (e.g., Tensorflow Transform) and extend our approach to distributed execution for Spark MLlib. We are planning an open-source release.

References

[1] J. Stoyanovich, B. Howe, and H. Jagadish, "Responsible data management," *VLDB*, vol. 13, no. 12, pp. 3474–3489, 2020.

[2] I. Chen, F. D. Johansson, and D. Sontag, "Why is my classifier discriminatory?" *NeurIPS*, pp. 3539–3550, 2018.

[3] S. Schelter, Y. He, J. Khilnani, and J. Stoyanovich, "Fairprep: Promoting data to a first-class citizen in studies on fairness-enhancing interventions," *EDBT*, 2019.

[4] K. Yang, B. Huang, J. Stoyanovich, and S. Schelter, "Fairness-aware instrumentation of preprocessing pipelines for machine learning," *HILDA workshop at SIGMOD*, 2020.

[5] A. Chouldechova and A. Roth, "A snapshot of the frontiers of fairness in machine learning," *Commun. ACM*, vol. 63, no. 5, pp. 82–89, 2020. [Online]. Available: https://doi.org/10.1145/3376898

[6] M. Vartak and S. Madden, "Modeldb: Opportunities and challenges in managing machine learning models," *IEEE Data Eng.*, vol. 41, pp. 16–25, 2018.

[7] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe *et al.*, "Accelerating the machine learning lifecycle with mlflow." *IEEE Data Eng. Bull.*, vol. 41, no. 4, pp. 39–45, 2018.

[8] N. Polyzotis, S. Whang, T. K. Kraska, and Y. Chung, "Slice finder: Automated data slicing for model validation," *ICDE*, 2019.

[9] N. Hynes, D. Sculley, and M. Terry, "The data linter: Lightweight, automated sanity checking for ml data sets," *NIPS MLSys Workshop*, 2017.