# Reconstructing and Querying ML Pipeline Intermediates

Sebastian Schelter, University of Amsterdam, s.schelter@uva.nl

In order to study and increase the reliability and correctness of the predictions made by machine learning (ML) applications, various data debugging techniques have been developed in the last years. These methods aim to reveal dependencies between the inputs (features) and outputs of an ML model. Examples include *Slice Finder* [2], *Gopher* [4], *Fairtest* [6] or *Fairlearn* [1], which identify subsets of the data where a model performs poorly or which negatively impact a model's fairness.

Unfortunately, the majority of these methods are not directly applicable to real-world ML applications with complex data preparation pipelines, as they assume a single, static input dataset with attributes to slice the data, aligned with features and predictions in matrix form. Therefore data scientists typically have to manually construct an appropriate evaluation dataset by rewriting the pipeline. This is tedious, as data scientists already spent the majority of their time writing data wrangling code [8], and also poses the danger of accidentally introducing hard-to-find ML-related bugs into the analysis [7].

**Provenance-based reconstruction and querying of pipeline intermediates**. In previous work [3], we proposed "inspections" for operator-centric debugging of ML pipelines. We generalize these techniques to automatically construct the required evaluation data for holistic debugging of complex pipelines. For that, we leverage a recently proposed model of ML pipelines, which treats them as a dataflow computation with known operations, containing only select-project-join and ML-specific featurization operations [3, 5]. In this model, a pipeline takes multiple relational inputs, internally filters, joins and transforms then into a train relation $R_{\text{train}}$ and a test relation $R_{\text{test}}$, builds their featurized counter parts $X_{\text{train}}$ and $X_{\text{test}}$, together with corresponding labels $y_{\text{train}}$ and $y_{\text{test}}$, and outputs the predictions $y_{\text{pred}}$.

During execution, we track fine-grained tuple-level provenance for the relational and feature-encoding operations in the dataflow. Based on the captured provenance, we generate views over the inputs to reconstruct $R_{\text{train}}$ and $R_{\text{test}}$ and over their captured matrix counter parts. Given these views, we can apply all previously discussed data debugging techniques directly. While the individual debugging approaches differ in their access patterns, the proposed views are sufficient to provide the required data for all of them. Note that some of the approaches like SliceFinder can even be expressed as a series of aggregation queries over these views.

**Prototypical implementation**. We prototype our approach for ML pipelines written as Python scripts, which either leverage pandas and scikit-learn, or pyspark and SparkML. For the former, we compute the provenance with an optimised variant of the approach from mlinspect [3], for the latter, we "piggyback" the computation of provenance polynomials on top of Spark's RDD operations. Our prototype is available at https://github.com/amsterdata/freamon.

We leverage DuckDB to create virtual views over the captured pipeline intermediates by joining them according to the provenance information. The API of our approach allows users to execute an ML pipeline, and materialize a view over the features, labels and predictions for training and testing the model, sliceable by attributes from the relational inputs. We provide example notebooks to showcase how to generate views to debug the predictions of a pipeline which trains a classifier to predict the helpfulness of product reviews. Data scientists can, for example, execute the pipeline, and subsequently generate a view over the test labels and predictions, sliceable by the category and rating attributes from the input tables:

```
# Execute sklearn pipeline, capture intermediates and provenance
view_generator = from_sklearn_pipeline('classify-product-reviews.py')
# Materialize a view over the test labels and predictions,
# sliceable by two attributes from the test input
test_view = view_generator.materialize_test_view(
  sliceable_by=['category', 'rating'],
  with_features=False, with_y=True, with_y_pred=True)
```

This view can directly be leveraged by external data debugging libraries. The data scientists could for example use Fairlearn [1] to compute fairness metrics for the review predictions with respect to the product category and review rating:

```
# Compute fairness metrics from the view via the fairlearn library
fairness_metrics = fairlearn.metrics.MetricFrame(
  metrics={'recall': sklearn.metrics.recall_score},
  y_true=test_view.y, y_pred=test_view.y_pred,
  sensitive_features=(test_view.category, test_view.rating>3)
print(fairness_metrics.by_group)
```

Alternatively, users can directly write aggregation queries over a virtual, non-materialised internal view over all inputs and intermediates for model training and testing. Such queries can for example compute the mean and variance of the cross-entropy loss for different data slices, analogous to SliceFinder [2]:

```
view_generator.execute_query("
 SELECT category, rating>3 AS toprated,
   AVG(cross_entropy_loss(y, y_pred)) AS avg_loss,
   VARIANCE(cross_entropy_loss(y, y_pred)) AS var_loss,
   COUNT(*) as size
 FROM virtual_test_view
 GROUP BY GROUPING SETS ((category,rating>3),(rating>3),(category))")
```

In future work, we plan to generalize this prototype into a system to manage and debug ML pipelines, based on their provenance and materialised intermediates.

## REFERENCES


[1] S. Bird, et al. Fairlearn: A toolkit for assessing and improving fairness in AI.
[2] Y. Chung, et al. Slice finder: Automated data slicing for model validation. *ICDE*'19.
[3] S. Grafberger, et al. Data Distribution Debugging for ML Pipelines. *VLDBJ*'22.
[4] R. Pradhan, et al. Interpretable explanations for fairness debugging. *SIGMOD*'22.
[5] B. Karlaš, et al. Data Debugging with Shapley Importance over End-to-End Machine Learning Pipelines. *arXiv preprint arXiv:2204.11131* 2022.
[6] F. Tramer, et al. Fairtest: Discovering unwarranted associations in data-driven applications. *EuroS&P*'17.
[7] S. Kapoor, et al. Leakage and the Reproducibility Crisis in ML-based Science. *arXiv preprint arXiv:2207.07048* 2022.
[8] Anaconda.com. The State of Data Science 2020. *https://www.anaconda.com/state-of-data-science-2020.*


 13th Annual Conference on Innovative Data Systems Research (CIDR '23). January 8-11, 2023, Amsterdam, The Netherlands.