Croissant: A Metadata Format for ML-Ready Datasets.

Croissant: A Metadata Format for ML-Ready Datasets

Mubashara Akhtar[7], Omar Benjelloun[4], Costanza Conforti[4], Pieter Gijsbers[12], Joan Giner-Miguelez[13], Nitisha Jain[7], Michael Kuchnik[8], Quentin Lhoest[5], Pierre Marcenac[4], Manil Maskey[9], Peter Mattson[4], Luis Oala[3], Pierre Ruyssen[4], Rajat Shinde[10], Elena Simperl[7,11], Goeffry Thomas[4,6], Slava Tykhonov[2], Joaquin Vanschoren[12], Jos van der Velde[12], Steffen Vogler[1], Carole-Jean Wu[8*]

[*]Authors in alphabetical order

[1]Bayer, [2]DANS-KNAW, [3]Dotphoton, [4]Google, [5]Hugging Face, [6]Kaggle, [7]King's College London, [8]Meta, [9]NASA, [10]NASA IMPACT & UAH, [11]Open Data Institute, [12]TUE & OpenML, [13]Universitat Oberta de Catalunya

ABSTRACT

Data is a critical resource for Machine Learning (ML), yet working with data remains a key friction point. This paper introduces Croissant, a metadata format for datasets that simplifies how data is used by ML tools and frameworks. Croissant makes datasets more discoverable, portable and interoperable, thereby addressing significant challenges in ML data management and responsible AI. Croissant is already supported by several popular dataset repositories, spanning hundreds of thousands of datasets, ready to be loaded into the most popular ML frameworks.

KEYWORDS

ML datasets, discoverability, reproducibility, responsible AI

ACM Reference Format:

Mubashara Akhtar, Omar Benjelloun, Costanza Conforti, Pieter Gijsbers, Joan Giner-Miguelez, Nitisha Jain, Michael Kuchnik, Quentin Lhoest, Pierre Marcenac, Manil Maskey, Peter Mattson, Luis Oala, Pierre Ruyssen, Rajat Shinde, Elena Simperl, Goeff Thomas, Slava Tykhonov, Joaquin Vanschoren, Jos van der Velde, Steffen Vogler, Carole-Jean Wu. 2024. Croissant: A Metadata Format for ML-Ready Datasets. In *Workshop on Data Management for End-to-End Machine Learning (DEEM 24), June 9, 2024, Santiago, Chile.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3650203.3663326

1 INTRODUCTION

Machine learning (ML) advances in generative AI, recommendation systems, natural language processing, and computer vision increasingly highlight the critical role of data management in technology breakthroughs. Yet, working with data remains time consuming and painful, due to a wide variety of data formats, the lack of interoperability between tools, and the difficulty of discovering and combining datasets [1, 2]. Data's prominent role in ML also leads to questions about its responsible use for training and evaluating ML models in areas such as licensing, privacy, biases, and more [3].

This paper presents *Croissant*, a metadata format designed to improve ML datasets' discoverability, portability, reproducibility, and interoperability. Croissant makes datasets "ML-ready", by enabling them to be directly loaded into ML frameworks and tools (see Figure 2 for sample code). Croissant describes datasets' attributes, the resources they contain, and their structure and semantics in a way that streamlines their usage and sharing within the ML community (Figure 1), while fostering responsible AI practices.

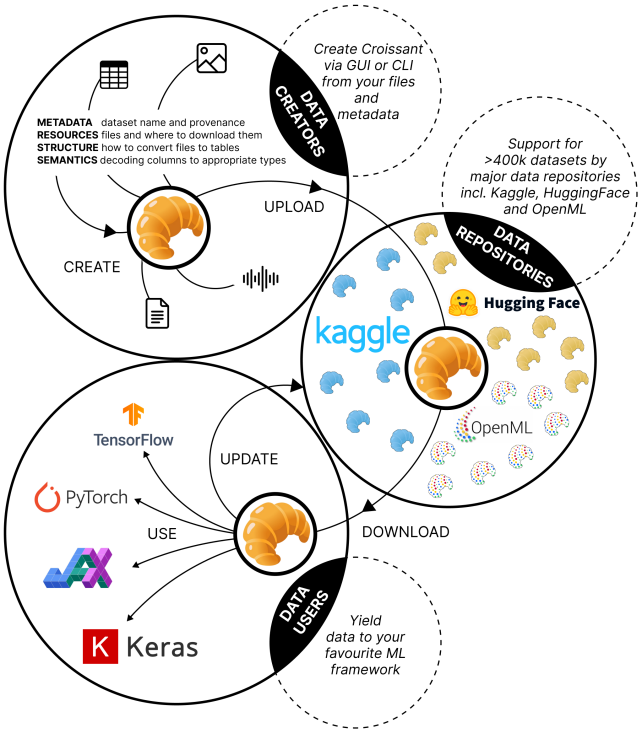

Figure 1: The Croissant lifecycle and ecosystem.

Croissant's goal is to describe most types of data commonly used in ML workflows, such as images, text, or audio. Even though datasets come in a variety of data formats and layouts, Croissant exposes a unified "view" over these resources, and lets users add semantic descriptions, and ML-specific information. The Croissant vocabulary [4] does not require changing the underlying data representation, and can therefore be easily added to existing datasets, and adopted by dataset repositories. Our main contributions are:

- Developing the Croissant metadata vocabulary for ML datasets.
- Demonstrating and analyzing its integration across major data repositories including HuggingFace, Kaggle, and OpenML.
- Developing open source reference implementations for the Croissant format, loaders, and editor.

DEEM 24, June 9, 2024, Santiago, Chile

ACM ISBN 979-8-4007-0611-0/24/06
https://doi.org/10.1145/3650203.3663326

```python
# 1. Point to a local or remote Croissant JSON file
import mlcroissant as mlc
url = "https://huggingface.co/api/datasets/fashion_mnist/croissant"
# 2. Inspect metadata
print(mlc.Dataset(url).metadata.to_json())
# 3. Use Croissant dataset in your ML workload
import tensorflow_datasets as tfds
builder = tfds.core.dataset_builders.CroissantBuilder(jsonld=url, file_format='array_record')
builder.download_and_prepare()
# 4. Split for training/testing
train, test = builder.as_data_source(split=['default[:80%]', 'default[80%:]'])
```

Figure 2: Users can easily inspect datasets (e.g., Fashion MNIST [5]) and use them in data loaders with Croissant. Visit https://github.com/mlcommons/croissant for a full example.

The Croissant project is open source[1], and developed through an open community process as part of ML Commons[2]. The remainder of the paper covers related work on dataset formats (Section 2), an overview of Croissant through a running example (Section 3), current integrations in dataset repositories and tools (Section 4), and concludes with future work directions (Section 5).

2 RELATED WORK

Croissant builds on the work of several communities towards providing a unified view of ML data management.

Dataset vocabularies for effective data management are paramount to all stages of an ML pipeline [6]. Previous efforts to organize datasets have led to the creation of standards like DCAT [7], and the Dataset vocabulary in `schema.org` [8], which make datasets easier to find and use across different platforms. These standards are useful for dataset metadata in general, but do not fully cater to the specific needs of data management in the context of ML. Data Packages [9] and CSV on the Web (CSVW) [10] add more details for handling and distributing datasets but are limited to tabular data. RO-Crate [11] and OAI-ORE [12] provide more general descriptions of dataset contents, but the former focuses more on making scientific data more reproducible, while the latter is concerned with the dissemination and preservation of digital objects. General-purpose data repositories like Dataverse [13] and CKAN [14] provide useful functionality to organize, manage and publish datasets, but interoperability with other systems via shared dataset representations is not their primary focus.

Data formats for storing and processing data, like Apache Arrow [15] and Apache Parquet [16], have brought significant improvements in performance and portability by allowing data to be used efficiently across different computing environments. Safetensors [17] targets the safe handling of tensor data, addressing security concerns. Formats such as Delta Lake [18] add ACID semantics to the Parquet format. Lance [19] and Ibis [20] focus on making data access more efficient. By contrast with these approaches, which dictate a specific, often optimized, data model and layout, Croissant is agnostic to the specific data representation, and describes datasets via a superimposed metadata layer.

[1]https://github.com/mlcommons/croissant
[2]https://mlcommons.org/croissant

Responsible AI (RAI) dataset documentation has become increasingly prevalent for making data transparent in terms of how it was created, what biases it reflects, and other considerations on data provenance. Data statements [21], Datasheets [6], and Data cards [22] were proposed to fill the lack of standardized dataset documentation. These formats are primarily meant as human-readable documentation. Croissant builds on these works by supporting the RAI documentation with a dedicated vocabulary extension [23].

3 A TASTE OF CROISSANT

Croissant is organized around the following four layers:

(1) **Dataset Metadata Layer:** Contains general information about the dataset, such as its name, description and license.
(2) **Resources Layer:** Describes the source data included in the dataset. Croissant introduces concepts like `FileObject` and `FileSet` for handling individual files and groups of files, supporting various data formats.
(3) **Structure Layer:** Describes and organizes the structure of the resources. Data is structured as `RecordSets`, which are roughly equivalent to nested relations. `RecordSets` are sufficient to describe a wide range of data, from text, to binary formats, to tabular and hierarchical data, and support joining across heterogeneous data. Croissant also supports basic data manipulation methods, like JSON Path and regular expressions, for flexible data extraction and transformation.
(4) **Semantic Layer:** Builds on the previous layers to apply ML-specific data interpretations, including custom data types (e.g., bounding boxes) and dataset organization methods (e.g., train/test splits). This layer is designed to be extendable, catering to the evolving needs of the ML community and supporting domain-specific application endpoints (e.g., geospatial analysis).

Together, these layers form a comprehensive dataset definition that not only facilitates the detailed description of datasets but also their practical use in ML projects, ensuring Croissant's adaptability and relevance in various applications. In the remainder of this section, we give the reader an intuition for these layers using examples. A full description of the Croissant format can be found in the Croissant specification [4].

3.1 Dataset metadata

Croissant dataset descriptions (Figure 3) are based on `schema.org/Dataset`, a widely adopted vocabulary for datasets on the Web [8]. Croissant specifies constraints on which `schema.org` properties are required, recommended and optional, and adds a few properties, e.g., to represent snapshots, live datasets, and citation information.

Furthermore, Croissant supports Responsible AI (RAI) via dataset documentation, in line with existing RAI initiatives [6, 21, 22] to promote transparency and accountability in AI. A dedicated RAI extension [23] expands Croissant's functionality, covering key use cases such as data lifecycle, labeling, safety, fairness, traceability, regulatory compliance, and inclusion.

3.2 Describing dataset resources

Croissant provides two primitive classes to describe the resources contained in a dataset: `FileObject` to describe individual files, and `FileSet` to describe sets of files.

```
{ "@type": "sc:Dataset",
  "name": "PASS",
  "dct:conformsTo":
     "http://mlcommons.org/croissant/1.0",
  "description": "PASS is a large-scale image dataset
     that does not include...",
  "citeAs": "@Article{asano21pass,  title = \"PASS: An
     ImageNet replacement...",
  "license": "cc-by-4.0",
  "url": "https://www.robots.ox.ac.uk/~vgg/data/pass/"}
```

Figure 3: Dataset metadata for the PASS dataset.

```
{ "@id": "pass0",
  "@type": "cr:FileObject",
  "contentUrl":
     "https://zenodo.org/6615455/PASS.0.tar",
  "sha256": "0be3a104d6257d83296460b419f82c71",
  "encodingFormat": "application/x-tar"},
{ "@id": "image-files",
  "@type": "cr:FileSet",
  "containedIn": {"@id":"pass0"}
  "includes": "*.jpg",
  "encodingFormat": "image/jpeg"},
  { "@id": "metadata",
  "@type": "cr:FileObject",
  "contentUrl":
     "https://zenodo.org/6615455/pass_metadata.csv",
  "sha256": "0b033707ea49365a5ffdd14615825511",
  "encodingFormat": "text/csv"}
```

Figure 4: Definitions of Resources for the PASS dataset.

In the example of Figure 4, `FileObject` is used to describe an archive file containing images, and a CSV file containing additional features about them. Declarations of object names are highlighted in yellow, with references in orange. `FileSet` describes the set of all images contained in the archive. Beyond this simple example, `FileObjects` can address files by URLs, local paths, or extract them from other `FileObjects`. `FileSets` can selectively target a subset of an archive's contents via include and exclude patterns, combine multiple archives, and address files listed in a manifest. Despite their simplicity, `FileObject` and `FileSet` can describe the resources of all datasets the Croissant community has worked with so far, across a broad range of repositories and data.

3.3 Describing structure with RecordSets

As we saw above, `FileObject` and `FileSet` can describe any types of resource. While this flexibility is required to represent the variety of data used in ML, applications still need a convenient representation to work with. A `RecordSet` represents the contents of any resource as a set of records. It is composed of a set of `Fields`, which define its structure. You can think of a `RecordSet` as a view on top of one or more `FileObjects` or `FileSets`.

To illustrate, Figure 5 shows a `RecordSet` combining images from PASS with additional features from a metadata CSV file. This example illustrates some of the noteworthy features of `RecordSets`:

- Each `Field` in the `RecordSet` defines the source of its data, which may refer to the contents of elements in a `FileSet` (images/image_content), file path information (images/hash, which extracts a portion of the filename using a regular expression), or columns of a tabular `FileObject` (e.g., images/coordinates/latitude).

```
{
  "@id": "images",
  "@type": "cr:RecordSet",
  "key": "images/hash",
  "field": [
    {
      "@id": "images/image_content",
      "@type": "cr:Field",
      "dataType": "sc:ImageObject",
      "source": {
        "fileSet": {"@id": "image-files"},
        "extract": {"fileProperty": "content"}
      }
    },
    {
      "@id": "images/hash",
      "@type": "cr:Field",
      "dataType": "sc:Text",
      "source": {
        "fileSet": {"@id": "image-files"},
        "extract": {"fileProperty": "filename"},
        "transform": {"regex": "([^\\/]*)\\.jpg"}
      },
      "references": {
        "fileObject": {"@id": "metadata"},
        "column": "hash"
      }
    },
    {
      "@id": "images/coordinates",
      "@type": "cr:Field",
      "dataType": "sc:GeoCoordinates",
      "subField": [
        {
          "@id": "images/coordinates/latitude",
          "@type": "cr:Field",
          "source": {
            "fileObject": {"@id": "metadata"},
            "column": "latitude"
          }
        },
        {
          "@id": "images/coordinates/longitude",
          "@type": "cr:Field",
          "source": {
            "fileObject": {"@id": "metadata"},
            "column": "longitude"
          }
        }
      ]
    }
  }
```

Figure 5: Definition of a `RecordSet` that joins images and structured metadata from the PASS dataset.

- `RecordSets` support joining data from multiple sources. To define the join, the images/hash field has a *reference* to the "hash" column of the metadata `FileObject`. This allows other fields (e.g., images/coordinates/latitude) to reference data from that `FileObject`.
- `Fields` can be nested. The images/coordinates field contains two subfields: images/coordinates/latitude and images/coordinates/-longitude. Croissant also supports nesting entire `RecordSets`,

e.g., to add annotations to images, where each image may correspond to multiple structured annotations. For a representative example, please see Croissant's COCO [24] definition[3].

3.4 Semantic layer

Finally, Croissant supports semantic typing of `Field`s (and `RecordSet`s). In the example above, the structured `Field` images/coordinates has the dataType `GeoCoordinates`[4] from schema.org. The subFields images/coordinates/latitude and images/coordinates/longitude are implicitly mapped to the latitude and longitude properties associated with that class, because their names match by suffix. Croissant also supports defining an explicit mapping via `equivalentProperty`.

Semantic typing plays an important role in Croissant. It provides a general mechanism to attach semantics to data by linking to known vocabularies and identifiers. From an ML perspective, semantic typing is used to describe important aspects, such as splits for test, training and validation, as well as label information. Semantic typing is also used to describe commonly used data types, such as bounding boxes. Finally, semantic typing can capture important information for responsible AI, such as gender or ethnicity distributions in datasets in a standardized way.

4 CROISSANT INTEGRATIONS

In parallel with the definition of the Croissant format, we have pursued a number of integrations, with the goals of 1) making Croissant immediately useful to users, and 2) grounding Croissant in the requirements of real-world datasets and tools. We next describe the lessons learned from these efforts.

Data Repositories. Croissant has been successfully integrated into three dataset repositories: Hugging Face datasets, Kaggle datasets, and OpenML, yielding over 400,000 datasets in the Croissant format. Two aspects of the design of Croissant have helped data repositories adopt Croissant with relatively low effort:

- Croissant is an extension of the widely adopted `schema.org/Dataset` vocabulary. Supporting Croissant was therefore a matter of adding additional fields to existing metadata.
- Croissant does not require changing the existing layout of data, but instead describes data as it already exists. Supporting Croissant only required adding metadata, and not changing any data.

In addition, data repositories also had characteristics that eased their adoption of Croissant:

- Most repositories support a small set of normalized data representations. For instance, Hugging Face converts the vast majority of its datasets to Parquet, so conversion to Croissant is primarily about handling that format.
- Repositories already have data type and schema information about the data they host, such as relational schemas for tabular datasets. This schema information can be readily converted to `RecordSet` definitions.

Data Loaders. Croissant's reference implementation is a standalone Python library[5] that supports the validation of Croissant dataset descriptions, their programmatic creation and manipulation, and serialization into JSON-LD. To consume data, the library provides an iterator abstraction that is fast and interoperates with existing data loaders. The TensorFlow Datasets [25] library provides a dataset builder[6] that prepares the dataset on disk in a format compatible with JAX, TensorFlow and PyTorch loaders. Alternatively, frameworks such as PyTorch DataPipes interface with the Croissant library by wrapping the iterator directly. We anticipate that additional optimization opportunities will arise with more varied and larger datasets, perhaps requiring distributed execution as well as more advanced operator scheduling.

Croissant Editor. Since Croissant is primarily a machine readable format (in JSON-LD), users can find it hard to create datasets by hand. We developed the Croissant Editor[7] (also on GitHub[8]), a tool that lets users visually create and modify Croissant datasets. The Croissant Editor provides form-based editing and validation of Croissant metadata, and bootstraps the definition of resources and `RecordSet`s by inferring them from the data uploaded by the user. The editor also integrates with the Croissant Responsible AI extension, and guides users in describing RAI aspects of their datasets.

Dataset Search. In addition to the support from individual data repositories, Croissant is also supported by Google Dataset Search [26]. When a user searches for a query that returns Croissant datasets, a special filter allows them to restrict the results to only Croissant datasets. This functionality allows users to effectively search for Croissant datasets across data repositories and all over the Web.

5 CONCLUSION & FUTURE WORK

In this paper, we introduced Croissant, a metadata format that makes datasets "ML-ready". We expect the Croissant format to evolve based on feedback from users and emerging needs from the rapidly evolving field of machine learning.

The success of Croissant is defined by its adoption in ML research and industry, which depends on the wide availability of Croissant datasets, and out-of-the-box support from ML tools and frameworks. To that aim, we invite dataset repositories and tool developers to join the Croissant community, and add support for Croissant. We strive to develop libraries to make Croissant adoption painless.

Semantically, Croissant defines primitives to link data for existing vocabularies. We expect users to guide further development of the ML-specific aspects of Croissant, beyond the basics that are currently covered, as specific ML problems and solutions will require richer ML metadata.

Finally, we believe that Croissant may also benefit other fields, given the broad range of datasets it can represent. Other communities may adopt Croissant in order to increase the interoperability of data repositories, tools and processing frameworks or platforms, for the benefit of their users. Additional features required by specific domains may be developed as Croissant extensions similar to the one that was developed for Responsible AI. We are starting to explore this approach in the Geospatial and Health domains.

[3] https://github.com/mlcommons/croissant/blob/main/datasets/1.0/coco2014/metadata.json
[4] http://schema.org/GeoCoordinates
[5] https://github.com/mlcommons/croissant/tree/main/python/mlcroissant
[6] https://www.tensorflow.org/datasets/format_specific_dataset_builders#croissantbuilder
[7] https://huggingface.co/spaces/MLCommons/croissant-editor
[8] https://github.com/mlcommons/croissant/tree/main/editor

ACKNOWLEDGMENTS

Joan Giner-Miguelez is supported by the AIDOaRt project, which is funded by the ECSEL Joint Undertaking (JU) under grant agreement No 101007350. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, and Spain. Pieter Gijsbers, Joaquin Vanschoren, and Jos van der Velde would like to acknowledge funding by EU's Horizon Europe research and innovation program under grant agreement No. 952215 (TAILOR) and No. 101070000 (AI4EUROPE).

REFERENCES

[1] Michael Kuchnik, Ana Klimovic, Jiri Simsa, Virginia Smith, and George Amvrosiadis. Plumber: Diagnosing and removing performance bottlenecks in machine learning data pipelines. *Proceedings of Machine Learning and Systems*, 4: 33–51, 2022.

[2] Luis Oala, Manil Maskey, Lilith Bat-Leah, Alicia Parrish, Nezihe Merve Gürel, Tzu-Sheng Kuo, Yang Liu, Rotem Dror, Danilo Brajovic, Xiaozhe Yao, Max Bartolo, William A Gaviria Rojas, Ryan Hileman, Rainier Aliment, Michael W. Mahoney, Meg Risdal, Matthew Lease, Wojciech Samek, Debojyoti Dutta, Curtis G Northcutt, Cody Coleman, Braden Hancock, Bernard Koch, Girmaw Abebe Tadesse, Bojan Karlaš, Ahmed Alaa, Adji Bousso Dieng, Natasha Noy, Vijay Janapa Reddi, James Zou, Praveen Paritosh, Mihaela van der Schaar, Kurt Bollacker, Lora Aroyo, Ce Zhang, Joaquin Vanschoren, Isabelle Guyon, and Peter Mattson. DMLR: Data-centric machine learning research - past, present and future. *Journal of Data-centric Machine Learning Research*, 2024. URL https://openreview.net/forum?id=2kpu78QdeE. Featured Certification, Survey Certification.

[3] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2021.

[4] Omar Benjelloun, Elena Simperl, Pierre Marcenac, Pierre Ruyssen, Costanza Conforti, Michael Kuchnik, Jos van der Velde, Luis Oala, Steffen Vogler, Mubashara Akthar, Nitisha Jain, and Slava Tykhonov. Croissant format specification. Technical report, 2024. URL https://mlcommons.org/croissant/1.0.

[5] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017.

[6] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets, 2021.

[7] Riccardo Albertoni, David Browning, Simon J D Cox, Alejandra Gonzalez Beltran, Andrea Perego, and Peter Winstanley. Data catalog vocabulary (DCAT) - version 3. https://www.w3.org/TR/vocab-dcat-3/, 01 2024. (Accessed on 03/18/2024).

[8] schema.org. Schema.org v26.0. https://github.com/schemaorg/schemaorg/tree/main/data/releases/26.0/, 02 2024. (Accessed on 03/18/2024).

[9] Data packages. https://specs.frictionlessdata.io/. (Accessed on 03/21/2024).

[10] Csv on the web: A primer. https://www.w3.org/TR/tabular-data-primer/. (Accessed on 03/21/2024).

[11] Stian Soiland-Reyes, Mercè Crosas Peter Sefton, Leyla Jael Castro, Frederik Coppens, José M. Fernández, Daniel Garijo, Marco La Rosa Björn Grüning, Simone Leo, Eoghan Ó Carragáin, Marc Portier, Ana Trisovic, RO-Crate Community, Paul Groth, and Carole Goble. Packaging research artefacts with ro-crate. *Data Science*, 5(2), 2022.

[12] Open archives initiative object exchange and reuse. https://www.openarchives.org/ore/. (Accessed on 03/21/2024).

[13] Gary King. An introduction to the dataverse network as an infrastructure for data sharing, 2007.

[14] Ckan. https://ckan.org/. (Accessed on 03/21/2024).

[15] Apache Software Foundation. Arrow columnar format — apache arrow v15.0.1. https://arrow.apache.org/docs/format/Columnar.html, 01 2024. (Accessed on 03/16/2024).

[16] Apache Software Foundation. Apache parquet. https://parquet.apache.org/docs/file-format/, 11 2023. (Accessed on 03/16/2024).

[17] Huggingface. huggingface/safetensors: Simple, safe way to store and distribute tensors v0.4.2. https://github.com/huggingface/safetensors, 01 2024. (Accessed on 03/18/2024).

[18] Michael Armbrust, Tathagata Das, Liwen Sun, Burak Yavuz, Shixiong Zhu, Mukul Murthy, Joseph Torres, Herman van Hovell, Adrian Ionescu, Alicja Łuszczak, et al. Delta lake: high-performance acid table storage over cloud object stores. *Proceedings of the VLDB Endowment*, 13(12):3411–3424, 2020.

[19] Chang She. Benchmarking random access in lance. https://blog.lancedb.com/announcing-lancedb-5cb0deaa46ee-2/, 03 2023. (Accessed on 03/18/2024).

[20] Ibis project. https://ibis-project.org/. (Accessed on 03/21/2024).

[21] Emily M. Bender and Batya Friedman. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604, 2018. doi: 10.1162/tacl_a_00041. URL https://aclanthology.org/Q18-1041.

[22] Mahima Pushkarna, Andrew Zaldivar, and Oddur Kjartansson. Data cards: Purposeful and transparent dataset documentation for responsible ai, 2022.

[23] Mubashara Akhtar, Nitisha Jain, Joan Giner-Miguelez, Omar Benjelloun, Elena Simperl, Lora Aroyo, Rajat Shinde, Luis Oala, and Michael Kuchnik. Croissant rai specification. Technical report, 2024. URL https://mlcommons.org/croissant/RAI/1.0.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[25] TFDS. TensorFlow Datasets, a collection of ready-to-use datasets. https://www.tensorflow.org/datasets, 03 2024.

[26] Dan Brickley, Matthew Burgess, and Natasha Noy. Google dataset search: Building a search engine for datasets in an open web ecosystem. In *The world wide web conference*, pages 1365–1375, 2019.

A PLATFORMS INTEGRATION

Croissant is available on a number of platforms, which we highlight in Figure 6. Each of the shown platforms (i.e., Kaggle, HuggingFace, and OpenML) has an option to export datasets in the Croissant format. Users can then load the dataset by pointing to the downloaded Croissant file.

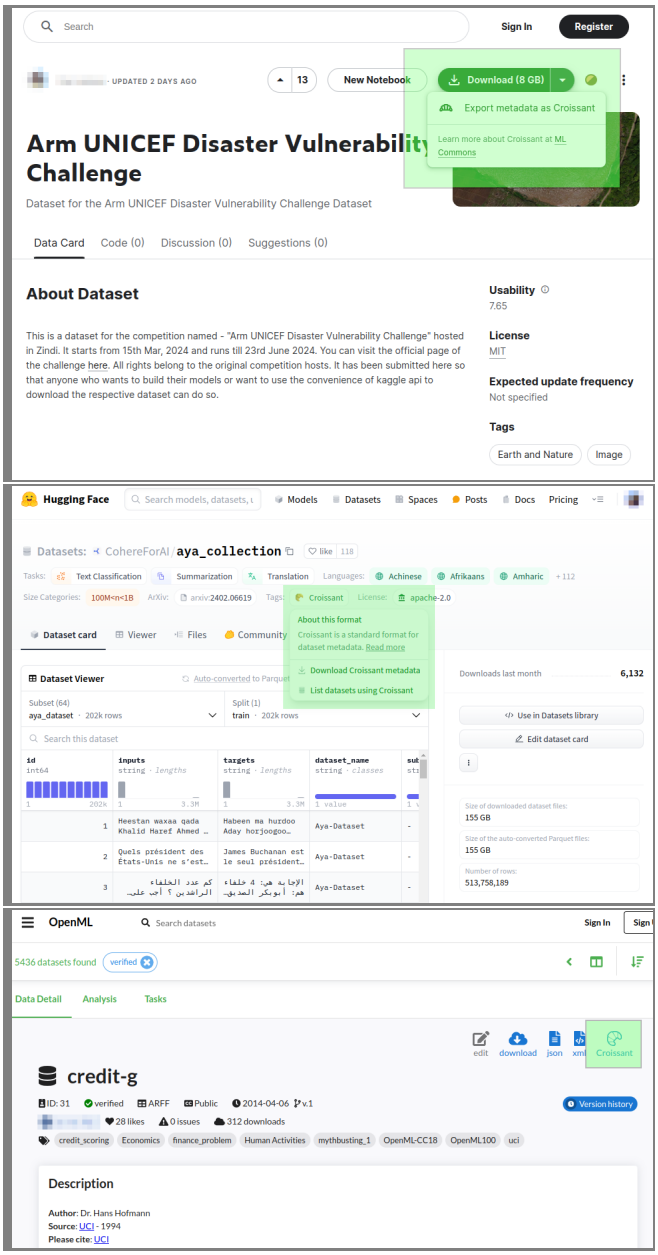

Figure 6: From top to bottom: Croissant integration across Kaggle, HuggingFace and OpenML data repositories. Croissant files for datasets on these platforms can be exported through the GUI, as shown highlighted in green, or programmatically through Croissant APIs.

B CROISSANT EDITOR

The Croissant open-source editor, shown in Figure 7, abstracts away the details of the Croissant syntax via a familiar user interface. Users can drag-and-drop files to start creating a Croissant dataset. The editor infers the resources and structure definitions from the data, and guides them in filling out required and optional fields. The editor can be run locally or accessed online at https://huggingface.co/spaces/MLCommons/croissant-editor.

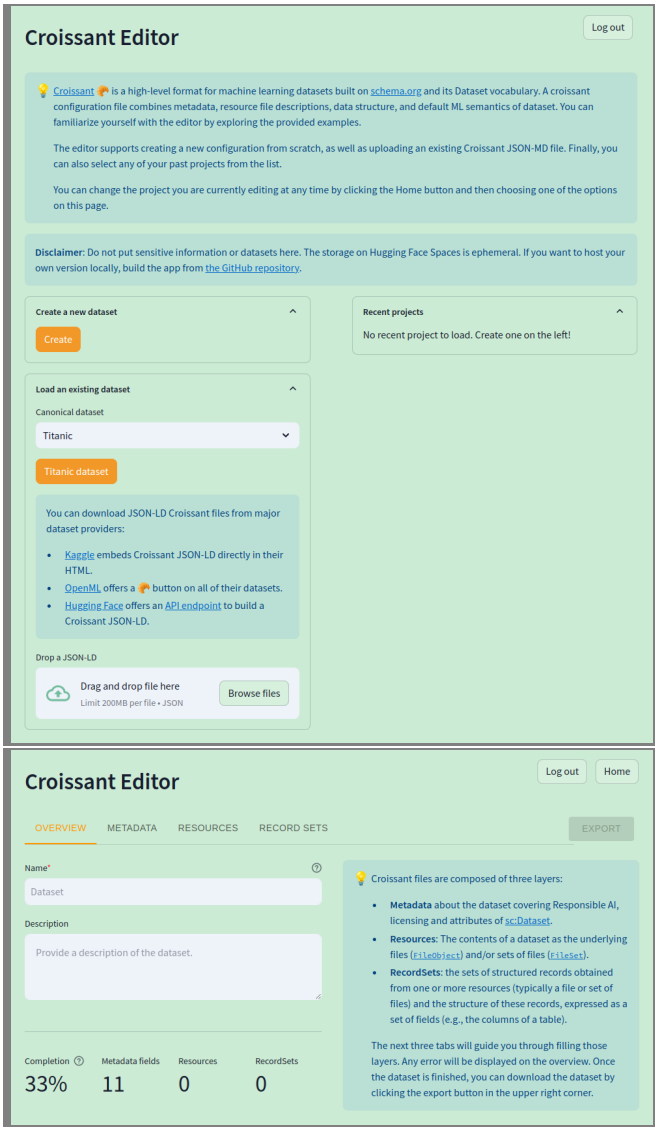

Figure 7: The GUI of the Croissant editor.