

Advanced Data Science

Zuil Pirola



UNIVERSIDADE
LUSÓFONA

How Data Science Research Should Be Reported

Why reproducible reporting matters

- Scientific credibility depends on **transparency and reproducibility**
- Data, code, and methodology should be **openly accessible**
- Enables **validation, reuse**, and further research

Where modern research is shared

- Research paper hubs (e.g., Hugging Face Papers trending section)
- Public **GitHub repositories** accompanying publications
- Increasing expectation of **open science practices**

How Data Science Research Should Be Reported

Core reproducibility principles

- Share datasets whenever **legally/ethically** possible
- Provide **clean, executable code** and **documentation**
- Ensure computational **environments** can be recreated
- Maintain clear project structure and reporting standards

Development Environments in Data Science

Why use an **IDE**?

- Improves productivity with **debugging**, **linting***, and **code navigation**
- Facilitates project **organization** and **version control** integration
- Common choices: VS Code, PyCharm, Jupyter-based workflows

Best practices

- Create one **environment per project**
- Document dependencies (**requirements.txt** / environment.yml)

*<https://peps.python.org/pep-0008/>

Development Environments in Data Science

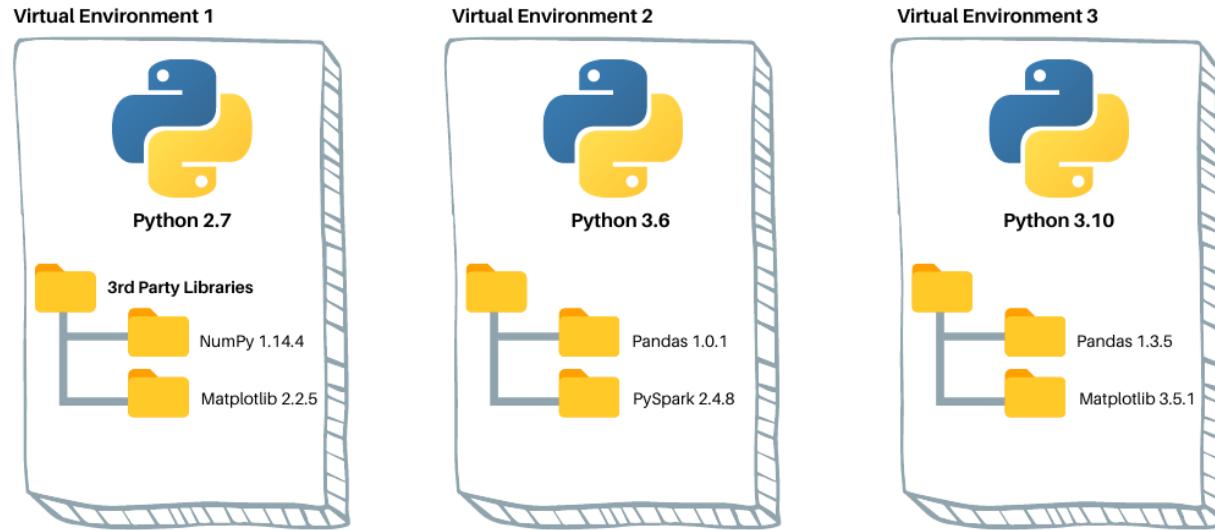
Environment management is essential

- **Avoids** dependency **conflicts** across projects
- Ensures **reproducibility** of experiments

Common tools

- **venv** → lightweight built-in Python virtual environments
- **conda** → environment + package manager widely used in data science

Configuration



dataquest.io

<https://www.dataquest.io/blog/a-complete-guide-to-python-virtual-environments/>

Configuration

Recommended Downloads (IDEs & Tools)

Local development environments for Python:

[!\[\]\(d65d67af0d48c0b8df8416565067ed4b_img.jpg\) PYTHON](#) [!\[\]\(903130ed6058e2c828f13cd7443f1600_img.jpg\) ANACONDA](#) [!\[\]\(b25355660c87d4489d7c3e6a9e50e78c_img.jpg\) VS CODE](#) [!\[\]\(d985d645507ce76ee0109eae98a32c20_img.jpg\) PYCHARM PROFESSIONAL](#)

Installation Guides

- Visual Studio Code
[!\[\]\(94a74132313ec4402774a2dadccdfc29_img.jpg\) VS CODE SETUP](#)
- PyCharm + Jupyter Notebook
[!\[\]\(8728ea602d895703ad57908b6ad88239_img.jpg\) PYCHARM SETUP](#)

Portuguese Version

[!\[\]\(6263c845ed99aeff3215bda781807c1c_img.jpg\) PYCHARM SETUP](#)

Example

The screenshot shows the Hugging Face Trending Papers interface. At the top, there's a search bar with the placeholder "Search any paper with AI" and a magnifying glass icon. To the right of the search bar are buttons for "Daily", "Weekly", "Monthly", and a star icon. A "Trending Papers" button is also present. Below the header, there are two main card-like sections for research papers.

Flavors of Moonshine: Tiny Specialized ASR Models for Edge Devices

Abstract: We present the Flavors of Moonshine, a suite of tiny monolingual speech recognition (ASR) models for edge devices. Monolingual ASR models are often large and complex, making them difficult to deploy in the cloud or on mobile devices due to memory constraints and latency requirements. Previous solutions suggest that multilingual ASR models can be used to reduce the size of individual monolingual models. However, this approach is not always feasible, especially for underrepresented languages. We propose a different approach by exploring cross-lingual phonetic constraints to build smaller, more efficient models. Our results show that for sufficiently small models (27M parameters), we can achieve performance comparable to state-of-the-art full-size models while maintaining a significantly smaller footprint. This is particularly important for edge devices where memory and processing power are limited. We demonstrate that our models can achieve high-quality transcription and recognition rates on a variety of datasets, including English, German, French, Spanish, Italian, and Portuguese. Our results show that our models outperform existing ASR models on these datasets, achieving superior error rates and enabling on-device ASR for underrepresented languages.

Submitted by evanking · 5 authors · Published on Sep 2, 2025

Moonshine: Speech Recognition for Live Transcription and Voice Commands

Abstract: This paper introduces Moonshine, a family of speech recognition models optimized for live transcription and voice commands. Moonshine consists of two models: one based on an encoder-decoder transformer architecture and another based on a rotary position embedding (RPE)-based absolute position representation. The RPE-based model processes audio in a sliding window fashion, allowing it to handle speech segments of various lengths, but with a fixed receptive field. The encoder-decoder model uses a hierarchical structure to capture long-term dependencies. The RPE-based model demonstrates a time efficiency of approximately 10ms per second of audio, while the encoder-decoder model is slightly slower at around 15ms per second. Moonshine also demonstrates a low latency for the encoder-decoder model, with a latency of less than 10ms for a 10-second speech segment while incurring no latency for the RPE-based model.

Upvote 10 · GitHub 3.78k · arXiv Page

Upvote - · GitHub 3.72k · arXiv Page

<https://huggingface.co/papers/trending>

Hands-On Practice: Building a Reproducible Repository (Ames Housing Dataset)

Objective

- Apply reproducibility principles by **creating a structured GitHub repository**
- Work with the **Ames** Housing dataset as a real Data Science **case study**

Dataset

- Ames Housing Dataset (Kaggle)
- Focus on **data loading, initial exploration, and reporting** findings

Hands-On Practice: Building a Reproducible Repository (Ames Housing Dataset)

```
# ames-project/
# |
#   +-- data/
#       +-- raw/
#       +-- processed/
#
#   +-- notebooks/
#   +-- src/
#   +-- reports/
#       +-- figures/
#
#   +-- README.md
#   +-- requirements.txt
#   +-- .gitignore
```

Expected Outcome

- A shared GitHub repository as a course **assignment**
- Clear organization, reproducible environment, and documented insights

The folder descriptions shown in the tree above are available in the Python notebook.

Hands-On Practice: Building a Reproducible Repository (Ames Housing Dataset)

Goal

- Carefully review the dataset features before any modeling step
- **Develop** a clear **understanding** of what **each variable** represents

Your Task

- Briefly describe each column in the dataset (1–2 lines per feature)
- Identify the **variable type** for each feature.

Hands-On Practice: Building a Reproducible Repository (Ames Housing Dataset)

Why This Matters

- **Prevents misinterpretation** of variables
- Helps **detect** inconsistencies or data **quality issues**
- **Guides** preprocessing, visualization, and modeling decisions

Deliverable

- A summarized **feature description** documented in your GitHub repository (README or reports)

Feature Types in Data Science

Numerical Variables

- Continuous: Real-valued measurements (e.g., price, height, temperature)
- Discrete: Countable integers (e.g., number of rooms, transactions)

Categorical Variables

- Nominal: Categories without intrinsic order (e.g., neighborhood, color)
- Ordinal: Ordered categories (e.g., quality rating, education level)

Binary Variables

- Two possible states (0/1, yes/no, true/false)
- Often treated as categorical but sometimes numeric for modeling

Feature Types in Data Science

Temporal Variables

- Dates, timestamps, durations
- Important for trends, seasonality, and time series analysis

Text / Unstructured Variables

- Free text, documents, comments
- Require NLP preprocessing (tokenization, embeddings, etc.)

Spatial / Geographical Variables

- Coordinates, regions, spatial relationships
- Used in geospatial and environmental analysis

Preliminary Data Quality Check

Start Evaluating Data Consistency

- Are there missing values in the dataset?
- Are they random, systematic, or meaningful?

Validate Data Types

- Check if stored types match expected semantics:
 - Dates → proper datetime format
 - Age, price, counts → numeric (int/float)
 - Categories → consistent categorical encoding

Preliminary Data Quality Check

Look for Incompatibilities

- Unexpected strings in numeric fields
- Incorrect formats (e.g., dates as text)
- Mixed data types within the same feature

Why This Step Matters

- Prevents modeling errors and biased conclusions
- Improves data cleaning and preprocessing decisions
- Strengthens reproducibility and reporting quality

Obrigado!