**Explanation of the Diagram**

1. **External ESG Data**:
   * Sources such as Eurofidai or Clarity AI provide raw CSV files or other data exports.
   * The data is typically delivered via direct file download or (optionally) through an API.
2. **Data Ingestion & Local Staging**:
   * This module **only** collects and stores raw files. For reproducibility, you keep unmodified CSVs in a local folder or dataset repo.
   * Minimal checks (row counts, column format) to confirm data integrity.
3. **Data Preprocessing Module**:
   * Implemented in Python (running inside VS Code Jupyter notebooks).
   * Cleans and merges multiple CSVs, handles missing data, performs outlier filtering, and normalizes numeric columns.
   * Outputs a “clean DataFrame” in memory or as an intermediate CSV file.
4. **Ontology Integration (Triple Store)**:
   * You load an **ESG ontology** (in OWL/RDF form) into an open-source triple store like **Apache Jena** or **GraphDB Free**.
   * If you wish, you can also convert your cleaned CSV columns into RDF for storage, or simply store references in the ontology that map “CSV column X → CO2DirectScope1 metric.”
   * Use **SPARQL** queries to find which metrics to select for a particular dimension (e.g., Environmental Risk metrics only).
5. **PCA & ML Model Module**:
   * A Python-based module (using scikit-learn) reads the relevant columns from either the local DataFrame or via SPARQL queries.
   * Applies PCA to reduce dimensionality.
   * Stores the resulting components, factor loadings, explained variance, etc.
6. **Visualization & Analysis Layer**:
   * Still in Python (Jupyter), you generate scree plots, biplots, and bar charts showing which ESG metrics contribute strongly to each principal component.
   * Optionally create an interactive dashboard with libraries like **Plotly**, **Altair**, or **Streamlit**—all open-source.
7. **End User / Researcher**:
   * Interacts with results inside Jupyter notebooks or a minimal web app.
   * Uses the outputs to glean insights: which ESG metrics are driving the largest variance or cluster together in principal components?
   * **Feedback Loop**: The user might revise the ontology or the data cleaning approach as new metrics or frameworks emerge.

**How This Diagram Satisfies the Requirements**

* **Single-Responsibility Components**: Each box in the diagram has a narrow focus:
  + (1) only collects and stages data;
  + (2) strictly cleans/merges data;
  + (3) manages the ontology and queries;
  + (4) handles PCA & ML tasks;
  + (5) visualizes results.
* **Data Flow**: Arrows show how data moves downstream:
  + from external sources to ingestion → preprocessing → ontology store → PCA → final analytics.
* **Interactions Among Models**:
  + The main “model” is PCA, with potential expansions (e.g., partial models for each ESG pillar).
  + The triple store acts as a knowledge repository that organizes ESG concepts and helps select features for PCA.
* **Where Data Is Stored & Retrieved**:
  + Raw files are stored locally in your staging area.
  + The triple store (e.g., Jena Fuseki) holds the ontology and possibly the data in RDF form.
  + The Python environment queries the triple store for relevant subsets to feed the PCA step.
* **User Interface**:
  + As a research project, the primary “UI” is likely your Jupyter notebooks.
  + Optionally, you can add a small web-based interface if you need non-technical stakeholders to explore results.

**Tips for Finalizing**

* **Create a polished version**: Use draw.io or Lucidchart to replicate this diagram, label all arrows with short phrases like “Cleaned DataFrame,” “SPARQL Query,” “Principal Components,” etc.
* **Add detail if needed**: For instance, if you run multiple PCA variants (one for Environmental, one for Social), show them as separate submodules in the “PCA & ML” box.
* **Match your actual naming**: If your project calls it “Module A” or “Ontology Manager,” rename the boxes to match your code structure.

This level of detail and clarity ensures your diagram meets the **machine-learning project** architecture requirements: how data is stored, how your models/analysis modules connect, and how end users can interact with (or view) the final outputs.

A screenshot of a computer program

AI-generated content may be incorrect.