Software Design

SeismoScan

Sponsors: Donna Shillington, James Gaherty, Christine Regalla

Mentor: Scott LaRocca

Alyssa Sombrero

Brady Wisniewski

Noah Carges

Thomas Rotchford

01.28.2026

Version 1

2. Table of Contents

3. Introduction

In the field of geology and seismology, there is a need to understand Earth's structure and natural hazards. The geoscience industry is valued in thousands of dollars, which are invested to support environmental assessments, earthquake preparedness, tsunami mitigation, exploration for renewable and non-renewable resources, water management, and more. Within the vast, data-driven industry, fault identification and tectonic analysis are an essential and fundamental process for academic research and applied to the industry's operation.

Our sponsors, Professors Donna J. Shillington, James B. Gaherty, and Christine Regalla from Northern Arizona University's School of Earth and Sustainability are active researchers in deformation, magmatism, sedimentation, and earthquake seismology. They have published works that contribute to the understanding of crustal formation, plate tectonics, and seismic activity, which relates to the objective of the project they have given us - creating a machine learning model to map faults in bathymetry data. This model will save the sponsor's time and enable more consistent fault mapping, and thus contribute further research on earthquake studies and tectonic plates.

The main problem our sponsors must solve is the time consumption it takes for a research assistant to manually identify faults by the given bathymetry data. This time-consuming process involves visual interpretations and characterization of fault attributes, and thus requires valuable research time and limits productivity. Furthermore, another downside of manual fault interpretation is that it could introduce potential inconsistency in the fault maps in terms of the level of detail of mapping.

We will develop a machine learning model to improve the efficiency of our client's workflow by producing more reliable and accurate data for use in their published research. By significantly reducing processing time, from weeks to minutes. The model will accelerate the client's ability to analyze seismic data in a deeper state, compared to what manual mapping can offer. Thus the machine learning model will make the workflow faster, consistent, and efficient.

By integrating machine learning to the seismic workflows, the project aims to improve the accuracy, speed, and consistency in fault identification. Ultimately, it will enhance the productivity of our sponsor's research team while advancing in the geoscience industry's pursuit of data-driven insights into Earth's fault system.

4. Implementation Overview

The machine learning program will be operated on a command line interface that accepts bathymetry data, NetCDF (Network Common Data Form) and outputs the following files: latitude and longitude text file and a visualization of potential faults as a PNG graph. The program will be implemented under the python language and utilize clustering algorithms, and unsupervised algorithms, from python library, scikit-learn library.

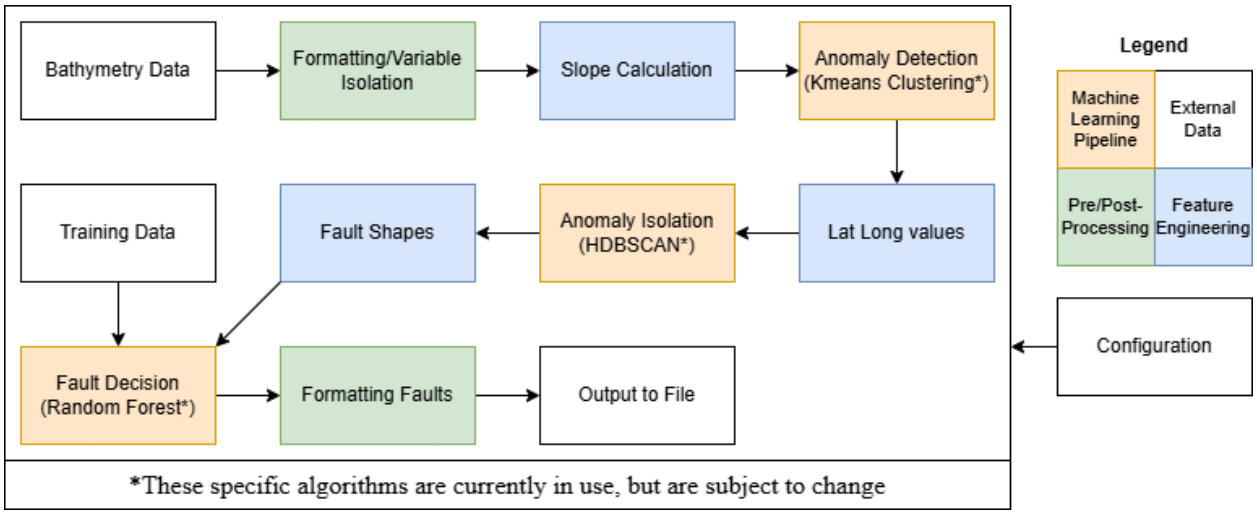
This program will require multiple sets of data. A data set that has been labeled will be split to be used as training data, validation data, and testing data. Along with that other unannotated bathymetry data will be provided in order to simulate usage scenarios. This data will be formatted as NetCDF data and provided to the team by the clients.

There will be two main computational processes done on the backend of this project. The first will be a set of data cleaning and formatting tools in order to remove unnecessary data and calculate useful values, such as the slope measurements, before providing it to the machine learning model. Second will be the machine learning model itself, the k-means clustering algorithm, random forest algorithm, and DBSCAN.

To output the model's predictions, the data will be stored as a .txt file containing latitude and longitude data points. The first value is latitude and the second is longitude. Each point will be separated by a new line. Using the latitude and longitude coordinates, a cluster graph will be generated in .png format that shows where the model makes its fault predictions. The predictions will be stored in a special file directory to organize the outputted data.

5. Architectural Overview

5.1 Architecture Diagram:



All code for this product will be contained within python therefore all communication will be done through internal tools. Along with that the user will interact with this product through a command line interface, as per client request. There are four main components of this product being External Data, Pre/Post-Processing, Feature Engineering, and the Machine Learning Models.

A pipeline approach for this machine learning task was chosen to allow the project to be separated into smaller, and easier problems to solve rather than needing a one-size-fits-all model. The problems that are being solved by separate models are detecting all anomalies on the sea floor, combining adjacent points of anomalies into one larger “feature”, and finally deciding on if that feature is a fault or something else. With that comes feature engineering that is dedicated to each model's problem. Due to the specific tasks of our pipeline we are unable to use a pre-built model and are therefore creating one from scratch.

5.2 External Data

All data that will be imported or exported to this product will be formatted in either a .txt, .grd (netcdf version 4), or .cnf format. In general input data to be analyzed will be formatted in a .grd format, training data may be in a .txt or a .grd format, and the output will be formatted as a .txt file. All configuration data will be formatted in a .cnf file to be identifiable as such.

5.3 Pre/Post-Processing

Pre- and Post-processing will be done specifically on the data coming directly from input files or directly to an output file. For example on the input data it will be isolating actual data from metadata, or post processing would be organizing faults in a way so they are ready for requested output.

5.3 Feature Engineering

Feature engineering is an extension of pre-processing, but is isolating or calculating specific information that will be inputted into any given machine learning model. This might include the slope, lat and longitude points, or elevation data.

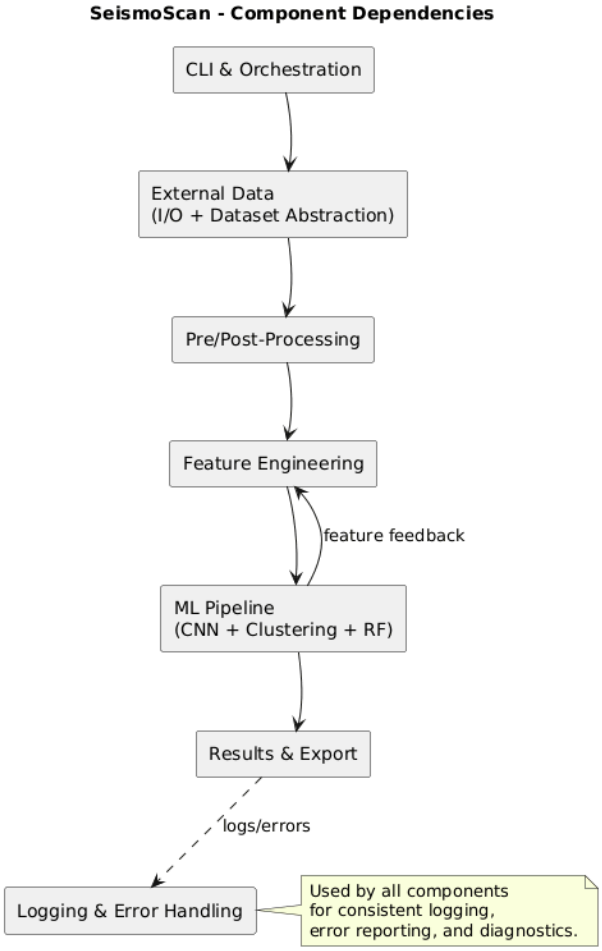
5.4 Machine Learning Pipeline

The “meat” of the product is a pipeline of machine learning models which are planned to be used sequentially, as in the output of one will be used in the feature engineering process of the next, each designed and handpicked with a specific goal in mind.

6. Component Level Design

With the overall pipeline established, the component-level design breaks down how each stage is realized internally. For each component, we describe its responsibilities, internal submodules, and external interfaces (inputs/outputs) to ensure that each major behavior is owned by a single component with minimal overlap. These details provide a concrete blueprint for implementation and integration, while preserving the modular structure described in the system architecture.

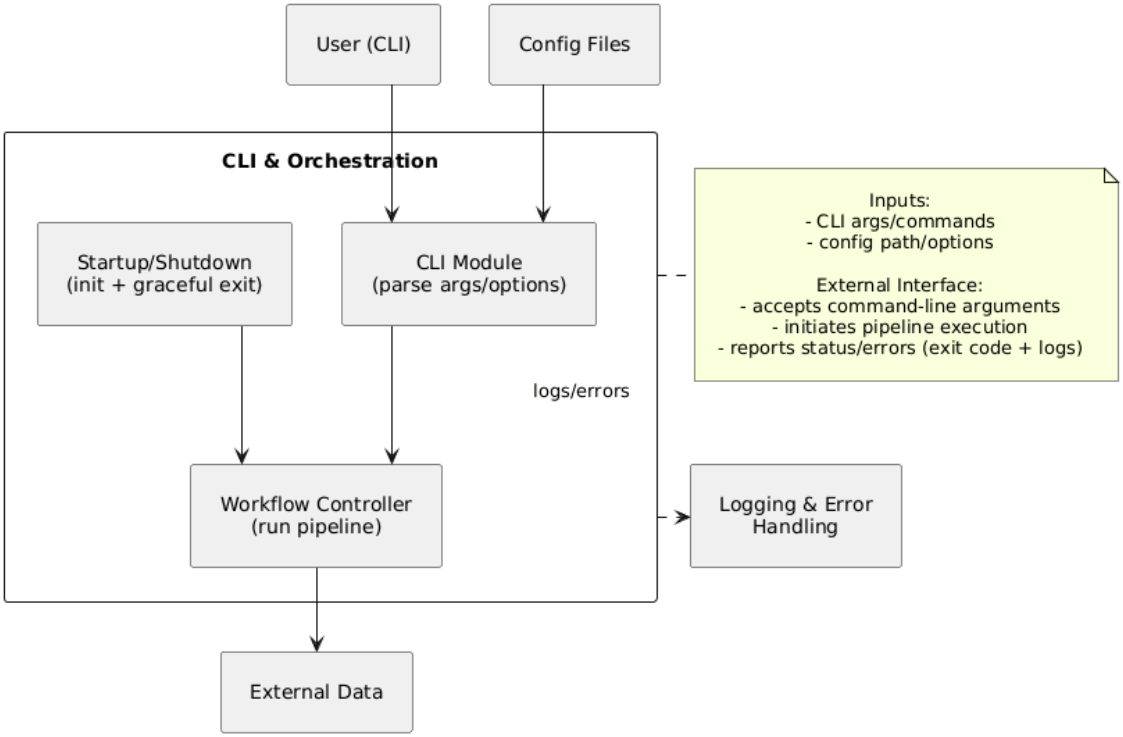
6.1 Overview Diagram:



Solid arrows indicate primary control and data flow between pipeline components. Arrows to Logging & Error Handling and Config & Validation represent shared service usage and cross-cutting concerns rather than pipeline progression.

6.2 CLI and Orchestration

6.2.1 Component Diagram



6.2.2 Responsibilities and Role in the System

The CLI and Orchestration component serves as the entry point for the system as well as the execution controller. It is responsible for coordinating the execution of the processing pipeline and ensuring that all components are invoked in the correct order and with validated inputs.

This component acts as a controller rather than a processing unit by delegating all domain-specific work to downstream components.

6.2.3 Internal Structure

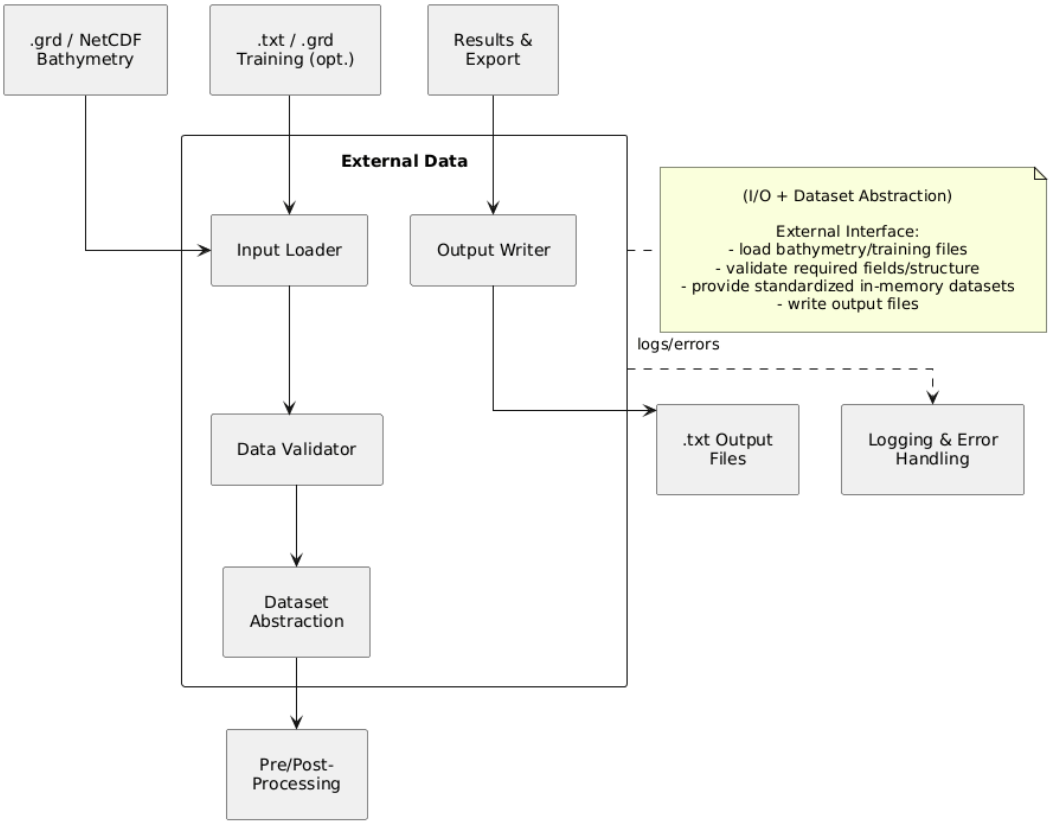
- *Command-Line Interface Module*:
  - Parses user input (file paths, configuration options, and execution modes)
- *Workflow Controller*:
  - Manages the sequence of pipeline execution.
- *Startup and Shutdown Logic*:
  - Initializes shared services and handles termination on failure.

6.2.4 External Interface

- Accepts command-line arguments
- Initiates pipeline execution
- Reports execution status and errors.

6.3 External Data

6.3.1 Component Diagram



6.3.2 Responsibilities and Role in the System

The External Data component manages all interactions with external files and data formats. This is responsible for loading input data, validating file structure, and exporting the final results.

This component abstracts file formats and details so that later components can operate on consistent in-memory data representations.

6.3.3 Internal Structure

- *Input Loader:*
    - Reads bathymetry and training data from .grd and .txt files.

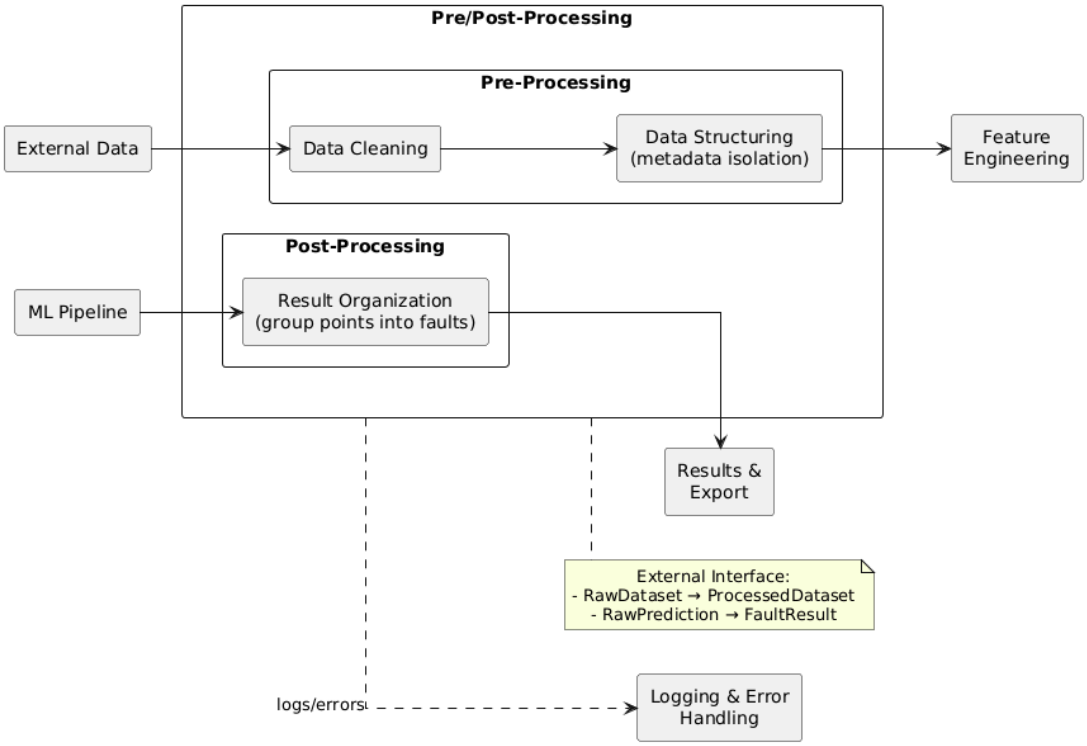
- *Data Validator:*
    - Confirms the presence and validity of data fields.
- *Output Writer:*
    - Writes processed fault results to output files.

6.3.4 External Interface

- Load datasets
- Provide structured datasets to preprocessing
- Write formatted output files

6.4 Pre/Post-Processing

6.4.1 Component Diagram



6.4.2 Responsibilities and Role in the System

The Pre/Post-Processing component prepares the input data for feature extraction and organizes the model's outputs for the final export. This component operates on structured datasets and applies transformations needed for later processing.

Pre-processing includes isolating relevant variables, cleaning data, and normalizing values. Post-processing includes organizing detected fault points into coherent fault structures suitable for output.

6.4.3 Internal Structure

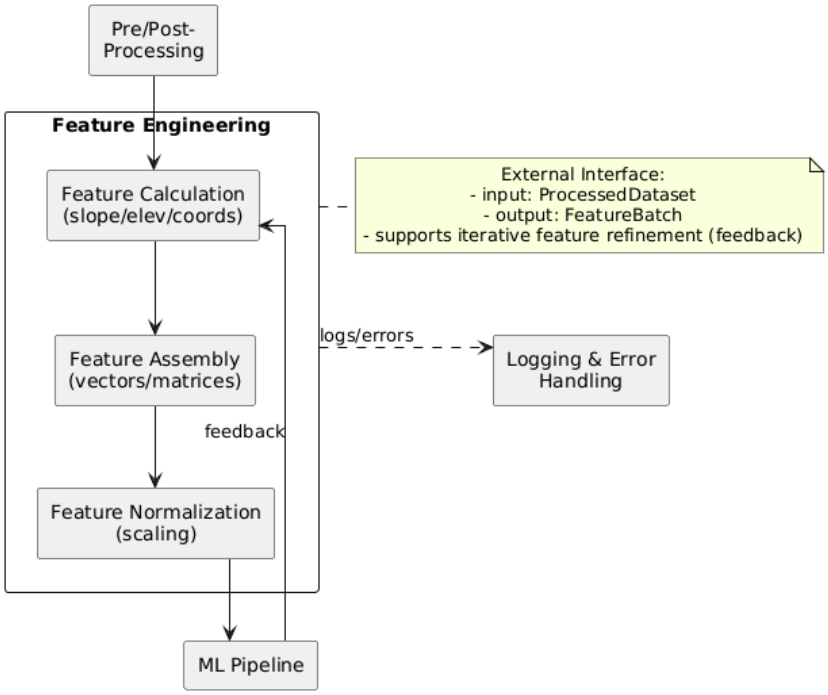
- *Data Cleaning Module:*
  - Handles missing/invalid values and applies normalization.
- *Data Structuring Module:*
  - Separates data from metadata and prepares it for feature extraction.
- *Result Organization Module:*
  - Groups detected fault points into fault segments.

6.4.4 External Interface

- Transform structured datasets into preprocessed representations.
- Organize model outputs into export-ready structures.

6.5 Feature Engineering

6.5.1 Component Diagram



6.5.2 Responsibilities and Role in the System

The Feature Engineering components obtains numerical values from preprocessed data for use by the machine learning models. These features represent relevant physical and spatial characteristics of the input data.

This component is responsible for ensuring that features are consistently calculated and formatted across training and inference workflows.

6.5.3 Internal Structure

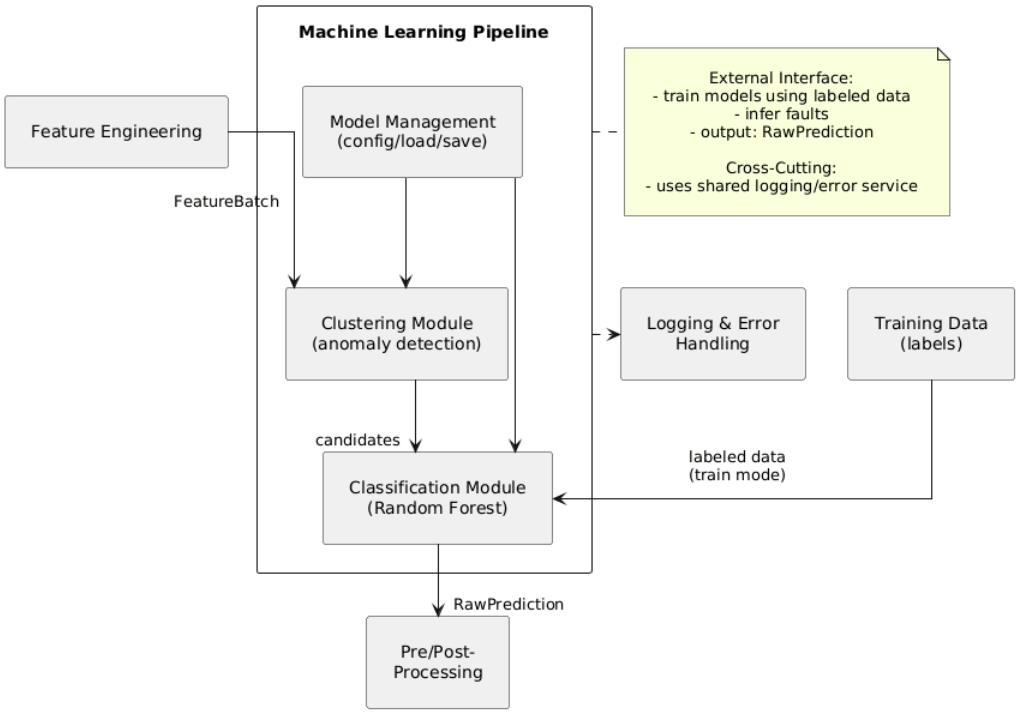
- *Feature Calculation Module*:
  - Computes features like slope, elevation, and coordinates.
- *Feature Assembly Module*:
  - Packages computed values into vectors or matrices.
- *Feature Normalization Module*:
  - Applies scaling as required for later models.

6.5.4 External Interface

- Generate feature representations from preprocessed data.
- Provide feature data to the Machine Learning Pipeline.

6.6 Machine Learning Pipeline

6.6.1 Component Diagram



6.6.2 Responsibilities and Role in the System

The Machine Learning Pipeline component holds all machine learning functionality without our project. This component is responsible for identifying

candidate faults and determining whether those candidates are representing true geological faults.

The pipeline may consist of multiple algorithmic stages, such as unsupervised clustering for anomaly detection and supervised classification for fault validation. From the system's perspective, these stages are one single cohesive component.

6.6.3 Internal Structure

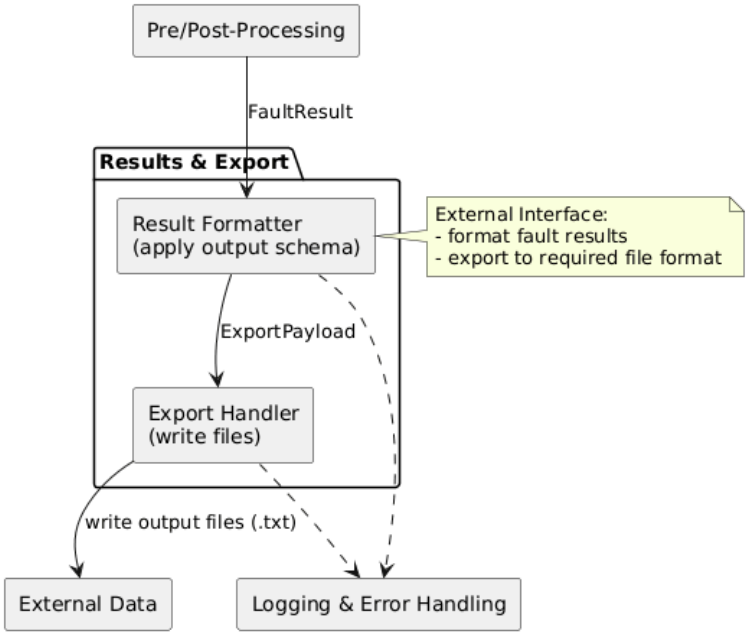
- *Clustering Module*:
  - Identifies fault-like anomalies.
- *Classification Module*:
  - Evaluates candidate faults using supervised learning.
- *Model Management Module*:
  - Manages model configuration and reuse.

6.6.4 External Interface

- Train models using labeled data.
- Perform inference on extracted features.
- Produce structured fault predictions.

6.7 Results and Export

6.7.1 Component Diagram



6.7.2 Responsibilities and Role in the System

The Results and Export component converts machine learning predictions into user-consumable output files. This is responsible for formatting results according to output specifications and writing them.

6.7.3 Internal Structure

- *Result Formatter*:
    - Converts internal representations into required output schema.
- *Export Handler*:
    - Writes formatted results to the output file.

6.7.4 External Interface

- Format fault detection results.
- Write final output files.

6.8 Cross-Cutting Concerns

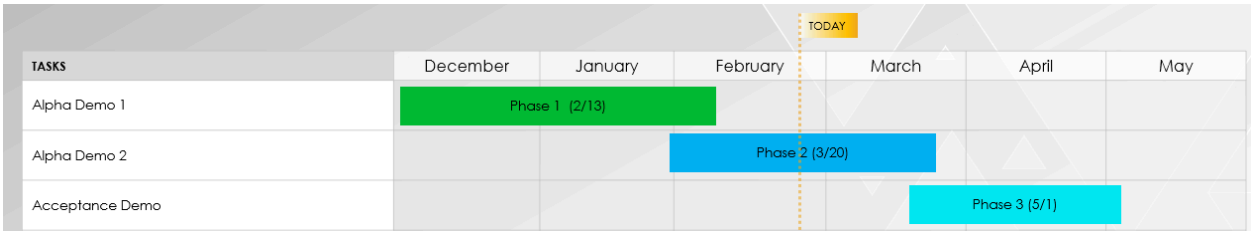
6.8.1 Logging and Error Handling

Logging and error handling are implemented as shared services used by all components. Each component detects and reports errors within its own scope, while centralized logging provides consistent reporting and diagnostics.

6.8.2 Performance and Scalability

The component structure supports efficient processing of large datasets through staged execution and modular design. The architecture allows future optimizations without changes to components interfaces.

7. Implementation Plan



7.1 Milestones

7.10 Alpha Demo 1

This demo turns all of our scripts into one cohesive unit. We also worked to improve our system architecture. The key deliverables of this part are multiple clustering algorithms, providing a config file which allows the user to easily modify key values, creating a robust system for handling all of the output, improving the algorithms, and improving the general infrastructure of the project by proofing it for errors.

7.11 Alpha Demo 2

This Demo will chiefly work on evolving our pipeline into the one we envision for the final product. To do this we will be integrating a supervised model (Random Forest) in order to serve as a final step for the pipeline. The use of this model with supervised learning will allow us to have more concrete parameters for success. During this process we will also make minor improvements to the CLI and pipeline in general by giving the user more power to analyze the data. This demo will be the first to include an actual confidence output. We will take the average distance between points in provided data and use it as a radius to test the distance between our program's identified points and labeled data from our client.

7.12 Final Product

This final iteration of our project will add the finishing touches and deliver a fully integrated research tool for our sponsors. The biggest part of this phase is making sure that the whole pipeline from start to finish works with totally new data and assessing our accuracy on that totally new data.

7.2 Component Implementation Order

1. Input Loader
2. Data Validator

3. Output Writer
4. Data Cleaning Module
5. Data Structuring Module
6. Result Organization Module
7. Feature Calculation Module
8. Feature Assembly
9. Feature normalization Module
10. Clustering Module
11. Model Management Module
12. Classification
13. Result Formatter
14. Export Handler

7.3 Module Integration Strategy

7.30 Phase 1

Parallel Development Opportunities:
CLI and Workflow Controller can be worked on simultaneously with the Input validator.

Key Dependencies:
Workflow Controller depending on the CLI interface existing. All modules downstream of the input loader and data validator.

7.31 Phase 2

Parallel Development Opportunities:
Data Cleaning can be developed simultaneously with Feature calculation. The Result Organization module can also be developed in parallel using fake module output

Key Dependencies:
Data Cleaning and Structuring must work. Feature Analysis depends on Feature Assembly.

7.32 Final Phase

Parallel Development Opportunities:
Clustering Module and Result Formatter. Model Management Module can be built alongside Clustering and Classification Modules

Key Dependencies:

Clustering and Classification require normalization. Export Handler needs formatted results to work.

7.4 Incremental Delivery Plan

7.40 Alpha Demo 1

1. Config System and CLI
2. Preprocessing
3. Feature Extraction
4. Clustering

7.41 Alpha Demo 2

1. Classification Module
2. CLI Improvements
3. Grouped Fault Output

7.42 Acceptance

1. Flexible Input support
2. Visualization
3. Fully Integrated pipeline

7.5 Technical Risks

There are several technical risks that will affect our development process. The biggest is the machine learning itself and engineering the features that may or may not form. We are working in the scope of the coast of Alaska and so the design choices we make for feature engineering will inevitably be influenced by this region. For example, We see a lot of sea mounts in this area and our program always picks them up along with the faults, so we will have to add parameters to screen them out and that may lead to faults in other non-Alaska regions being screened out. To mitigate this and other errors similar, after testing upon the Alaskan dataset, we will make a document which describes the affinity for certain clusters to form and things that we have deliberately excluded. This is our best opportunity for mitigating this since labeled data is very limited and is not nearly comprehensive enough to predict how non-Alaskan regions will impact it. To mitigate this risk, we will apply a combination of feature engineering, model tuning, and cross-validation to identify models that generalize effectively.

A major risk for us is the fact that the multi stage machine learning pipeline may simply not be the most optimal solution and It could be the case that one model outperforms all other permutations of models. If in the process of evaluating models we find this to be the case then we will document it extensively and deliver the best performing model to the client. However, in

order to make this determination we will evaluate multiple different combinations as well as each model individually. They will be measured with the metrics of proximity to the labeled data, as discussed previously, algorithmic efficiency, and memory usage. This way the reason behind our determinations will be clear and based on empirical evidence.

Another issue is scalability. Since all of the algorithms we have implemented at the present moment either have a runtime of O(n^2) or O(nlg(n)) we will have to cap the input size to prevent the program from using too many resources. We will also have to find the most important parameters to optimize the size of the input.

We also plan to take different approaches for feature engineering. Generally, we will test varied inputs ,like slope, curvature, or gradient, in order to assess model performance and see what correlates to greater affinity for the features we want to see.

8. Conclusion

Overall this product will greatly benefit researchers in the geological field, to help with the study of important geological processes that may or may not be affected by faults under the sea floor. Our product will provide a machine learning pipeline that will be supplemented by thorough feature engineering, configuration options to tune the performance as needed, along with pre and post processing tools to make the input and output of the data as seamless and useful as possible.

Using a machine learning pipeline will allow us to more strategically choose our models and algorithms to complete specific tasks, rather than attempting to create a one size fits all solution. This will also provide more support in the testing process as it will be easier to identify and adjust problem spots in our pipeline, rather than having to adjust the entire thing.

Completing one level of this pipeline per sprint will give us the time to focus, adjust, and refine each layer with the attention to detail required to pull off a project of this scale. These design choices and schedule, gives us the best chance at delivering a great product.