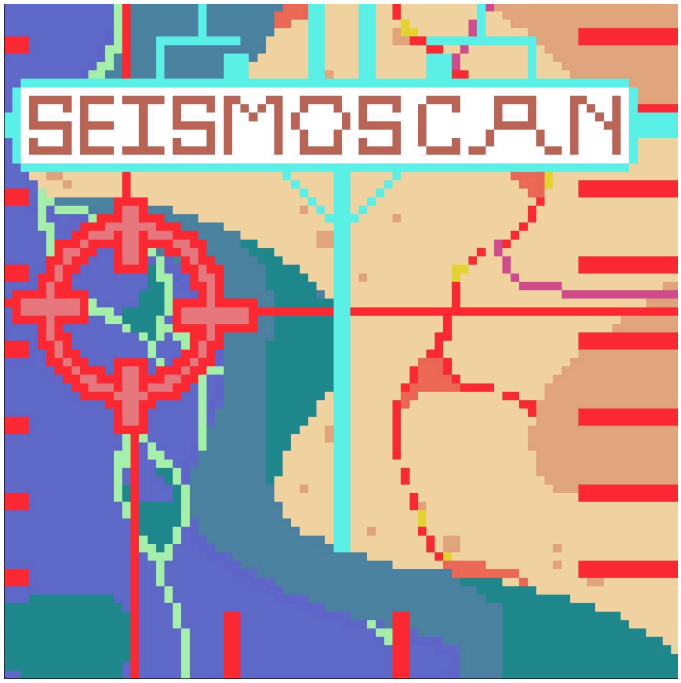
Software Testing

SeismoScan

Sponsors: Donna Shillington, James Gaherty, Christine Regalla



Mentor: Scott LaRocca

Alyssa Sombrero

Brady Wisniewski

Noah Carges

Thomas Rotchford

03.20.2026

2. Table of Contents

3. Introduction

The project is a machine learning model that is used to detect oceanic faults, using a given bathymetry data stored in a NETCDF file format. The model is intended for researchers and professors, who study seismic activities and research. The primary purpose of this project is to produce reliable, accurate, and supportive fault-detected data that will be used in research studies. These outputs are meant to be evidence that advocates for scientific analysis and arguments regarding oceanic seismologic processes. Thus the output the system provides must follow critical attributes of correctness, reliability, usability, and performance.

The model contains a data processing and machine learning pipeline, implemented through Python and executed in a terminal interface on both Linux and Windows operating systems. The data processing pipeline contains the following elements: data intake from NETCDF files, preprocessing to extract the data, clustering algorithms (K-Means, DBSCAN, HDBSCAN, K-Medoids), and Random Forest algorithm to filter out the false positive faults. The project outputs an optional visualization (PNG image) and text files (Latitude and Longitude format .TXT file, Debugging .TXT file) of potential fault locations pinpointed by the model. There is a configuration file (.CNF) to make adjustments to the parameters such as chunk lengths, angles, delimiter, slope offsets and more found in the ‘config’ folder. The critical external dependencies needed to run the model are the Python libraries, PyGMT, NumPy, Matplotlib, and Scikit-learn.

The test scope implemented will be focusing on validating the precision and performance of the fault detection behaviors, including the clustering behaviors, classification accuracy, and output ranges. Within the scope are the machine learning models, configuration system, and output formats. Out of the scope are the terminal environment, Python library implementations, and Windows / Linux operating systems compatibility. The components, within the scope, are prioritized since the primary function of the model is to accurately detect faults, demonstrate configurable model behavior, and produce informative outputs for seismic researchers.

The testing strategy will be devised into multiple approaches:

1.) Accuracy Testing to evaluate how well the system correctly identifies a fault using metrics (confusion matrices, precision, recall, F1 score). Accuracy Testing is planned to be implemented by the acceptance demonstration.

2.) Performance Testing to measure the model’s run time behavior across diverse dataset sizes (small, average, maximum) and configured parameters to test the adaptability of the model.

Performance Testing is planned to be implemented by the acceptance demonstration.

3.) Usability Testing to ensure easier user interaction between the model and non-developer user through configuration and execution pipeline. The Usability Testing is planned to be completed by the acceptance demonstration.

This testing strategy was devised by the project's primary use in scientific research. With faulty data produced by the model, it would create misleading conclusions, thus accuracy testing is prioritized above all testings. Performance testing is included to ensure the system is still usable for maximum datasets . Usability testing to ensure the project can be operable by researchers who don't have an extensive technical background. Overall the testing phase is to guarantee the machine learning model created for the researcher's purpose, can fulfill the objective it was designed to do in an ideal manner.

The following sections will go into detail about the testings the machine learning model will undergo

4. Accuracy Testing

In this project, we use accuracy-based evaluation of ML components rather than traditional unit testing. Instead of testing functions, the primary focus is validating the correctness and performance of the fault detection pipeline.

The dataset is grouped into spatial objects defined as connected components in raster space. Pixels are grouped based on 8-connectivity, meaning each pixel is considered connected to immediate neighbors in all directions. Each object represents a contiguous region of pixels whose size is determined by how the pixels connect spatially. These objects are then randomly assigned to 70% training, 15% validation, and 15% testing, with the training set being used to fit each model, the validation set being used to fine tune the parameters, and the testing set being used to evaluate the accuracy and compare different configurations.

Prediction outcomes are determined by comparing model predictions to ground truth labels. Correct classifications are considered successful predictions, while incorrect classifications are treated as failures.

4.1 Evaluation Metrics:

Model performance will be assessed primarily using a confusion matrix, which includes:

- True Positives (TP): correctly predicted faults
- False Positives (FP): non-faults incorrectly predicted as faults

- False Negatives (FN): missed faults
- True Negatives (TN): correctly rejected non-faults

From the confusion matrix, additional metrics may be considered:
- Precision: measures how many predicted faults are correct
- Recall: measures how many true faults are detected
- F1 Score: balances precision and recall

The best-performing model will be selected based on performance on the test dataset, with an emphasis on balancing fault detection and minimizing false positives.

4.2 Model Configurations:

As stated previously, our project evaluates multiple model combinations, each representing a different approach to generating and classifying candidate fault objects. Each configuration will undergo the same evaluation process.

4.2.1 RF Only

In this configuration, Random Forest is applied directly to candidate objects without additional clustering refinement. Each object is represented by extracted features such as geometric properties and terrain-based attributes, and the classifier predicts whether the object corresponds to a fault.

This model serves as a baseline for comparison. It will allow the team to evaluate how well the classifier performs when solely relying on feature-based learning. It is expected to correctly identify well-defined fault structures but may struggle with noisy or poorly segmented candidate objects. This configuration will help determine whether clustering is necessary to improve object quality.

Accuracy Evaluation:
- Prediction performance on the test dataset

Assessment Metric:
- Confusion matrix, precision, recall, F1 score

Boundary/Edge Cases:
- Compact clusters that do not resemble linear fault structures may still appear significant.

    - The model is expected to reject these objects by identifying their lack of elongation and weak linear characteristics.
- True faults may be split into smaller segments during clustering and grouping.
    - The model should classify these segments as faults if they retain sufficient geometric features.
- Regions with strong elevation or gradient changes may resemble faults but do not correspond to actual fault structures.
    - The model should be able to differentiate these regions by considering multiple features.
- NaN will be considered as a non-fault and will be tested to always be treated as such

4.2.2 DBSCAN + RF

This configuration applies DBSCAN to group nearby points into candidate fault objects based on density. These clustered objects are then passed to the Random Forest classifier for final classification

DBSCAN is used to identify dense regions and separate noise, making it useful for grouping fault-like structures while filtering out scattered points. In the expected output, the model should reduce noise and improve detection of continuous fault structures, but may struggle in areas with varying density and where clusters are not well-defined. This configuration would evaluate whether density-based clustering improves the quality of input objects and overall classification accuracy.

Accuracy Evaluation:

- Prediction performance on the test dataset

Assessment Metric:

- Confusion matrix, precision, recall, F1 score

Boundary/Edge Cases:

- Compact clusters that do not resemble linear fault structures may still appear significant. The model is expected to reject these objects by identifying their lack of elongation and weak linear characteristics.

- True faults may be split into smaller segments during clustering and grouping. The model should classify these segments as faults if they retain sufficient geometric features.
- Regions with strong elevation or gradient changes may resemble faults but do not correspond to actual fault structures. The model should be able to differentiate these regions by considering multiple features.
- NaN will be considered as a non-fault and will be tested to always be treated as such

4.2.3 HDBSCAN + RF

This configuration takes HDBSCAN, which is used to generate candidate objects by identifying clusters of varying density, and follows it with the Random Forest classification. Unlike DBSCAN, HDBSCAN can adapt to changes in density throughout the dataset.

This will be used to determine whether a more flexible clustering approach would improve robustness in complex terrains where fault structures may not have similar densities. The model should be able to handle noisy environments and varying densities in a better manner than DBSCAN, but may also generate additional small clusters that would need to be filtered.

Accuracy Evaluation:

- Prediction performance on the test dataset

Assessment Metric:

- Confusion matrix, precision, recall, F1 score

Boundary/Edge Cases:

- Compact clusters that do not resemble linear fault structures may still appear significant.
    - The model is expected to reject these objects by identifying their lack of elongation and weak linear characteristics.
- True faults may be split into smaller segments during clustering and grouping.
    - The model should classify these segments as faults if they retain sufficient geometric features.
- Regions with strong elevation or gradient changes may resemble faults but do not correspond to actual fault structures.

    - The model should be able to differentiate these regions by considering multiple features.
- NaN will be considered as a non-fault and will be tested to always be treated as such.

4.2.4 K-Means/K-Medoids + RF

This configuration uses clustering methods K-Means and K-Medoids identify candidate points, raster-space grouping to group these points into candidate objects, and then those objects are used in the Random Forest classification. The difference between the two clustering methods is that K-Means uses the mean of points within a cluster, while K-Medoids selects an actual data point as a cluster's center.

These methods would determine whether partition-based clustering can effectively organize candidate points into meaningful structures prior to object formation and classification. Comparing K-Means and K-Medoids would also allow us to assess how different strategies would impact the quality of resulting candidate objects and overall accuracy. We expect that K-Means will produce more uniform clusters but may be sensitive to noise and outlier. K-Medoids, on the other hand, we expect to be more robust to noise. After raster-space grouping, both methods should produce structured candidate objects suitable for classification.

Accuracy Evaluation:

- Prediction performance on the test dataset

Assessment Metric:

- Confusion matrix, precision, recall, F1 score

Boundary/Edge Cases:

- Compact clusters that do not resemble linear fault structures may still appear significant.
    - The model is expected to reject these objects by identifying their lack of elongation and weak linear characteristics.
- True faults may be split into smaller segments during clustering and grouping.
    - The model should classify these segments as faults if they retain sufficient geometric features.
- Regions with strong elevation or gradient changes may resemble faults but do not correspond to actual fault structures.

    - The model should be able to differentiate these regions by considering multiple features.
- NaN will be considered as a non-fault and will be tested to always be treated as such

5. Performance Testing

Performance testing will focus on how the pipeline's runtime changes with different configurations and dataset sizes. While performance is a very important part of many pieces of software, this is a product that is not willing to trade any accuracy for an increase in speed performance. It is still important to note what these runtimes may be to help prepare users for what to expect when running this pipeline.

5.1 Evaluation Metrics:

- Datasets: This will be evaluated on raw data size in MB
- Runtime: This will be evaluated on time it takes to complete a task
- Memory Usage: This will not be tested as the data will be chunked in such a way where data load will not overload any modern machine that this Model Pipeline is intended to run on.

5.2 Datasets:

For testing datasets we will use three different datasets dubbed as small, average, and maximum sized data sets. The small dataset will be around 1MB, average will be 500MB, and maximum at 1000MB size.

5.3 Model Configuration:

Each configuration will be tested on a machine with 64GB of memory and a 12 core 4700 MHz processor to ensure consistent and fair results. This will identify generic runtime principles, which will be reported as Linear, Exponential, or Logarithmic as examples. Each test will be run on similar configuration options, with each option being the “preset” configuration the project will ship with.

Due to some randomizations involved each model pipeline will be tested 5 times on each file size with the Median result for each section being the final result.

5.3.1 RF Only

Runtime evaluation:

- Runtime of the pipeline in its entirety, including preprocessing and all visualizations
- Runtime of Random Forest model alone

Evaluation metric:

- Runtime based on scaled data set sizes (Linear, exponential, etc)

5.3.2 DBSCAN + RF

Runtime evaluation:

- Runtime of the pipeline in its entirety, including preprocessing, visualizations, Verbose output, and
- Runtime of DBSCAN clustering model alone
- Runtime of Random Forest model alone

Evaluation metric:

- Runtime based on scaled data set sizes (Linear, exponential, etc)

5.2.3 HDBSCAN + RF

Runtime evaluation:

- Runtime of the pipeline in its entirety, including preprocessing, visualizations, Verbose output, and
- Runtime of HDBSCAN clustering model alone
- Runtime of Random Forest model alone

Evaluation metric:

- Runtime based on scaled data set sizes (Linear, exponential, etc)

5.2.4 K-Means/K-Medoids + RF

Runtime evaluation:

- Runtime of the pipeline in its entirety, including preprocessing, visualizations, Verbose output, and
- Runtime of K-Means/K-Medoids clustering model alone
- Runtime of Random Forest model alone

Evaluation metric:

- Runtime based on scaled data set sizes (Linear, exponential, etc)

6. Usability Testing

Clients will have a hands-on period with a prototype and no team member present to allow the clients to provide feedback on the interface of the project. They will provide feedback on both the Configuration Editing script, along with the Main script that runs the pipeline.

7. Testing Workflow and Quality Control

We shall aim to address any issue that arises during testing with clear documentation. We will categorize them by severity as either high, medium, or low.

7.1 Addressing Bugs:

High severity bugs will be top priority and must be fixed before any significant deadlines. These issues are things like our pipeline failing or in general anything that prevents the system from creating interpretable outputs. It is unique to our project that our system's fault identification issues could be considered bugs. However solving these issues is the crux of our whole project so it will only be considered a high severity issue if it significantly changes our output for the worse. Medium issues for use would be performance issues, since our runtime is mostly dominated by the cs models and our client didn't give us strict performance requirements. Low severity issues would be things like formatting issues or slightly obtuse debug file outputs since these do not impact the workings of the system.

7.2 Cycle Criteria:

For a development cycle to be considered complete we must have resolved all high severity issues, the pipeline must run successfully with all the config options we have included, and our output files must generate successfully . For a major milestone we will aim to only have low severity issues

To start a development cycle we must have dedicated features to work on and appropriate testing data from our client.

8. Conclusion

Our project aims to help geologists in their research of how faults affect many earth processes. In order to provide the best possible product our pipeline will go through extensive testing of all available configurations. This will include accuracy testing, which is the most important piece of the puzzle. A confusion matrix of true positives and negatives, along with false positives and negatives will be the focal point of this testing. Less importantly will be efficiency testing, with the main focus assuring runtimes are not extremely unreasonable. Per the client runtime is generally not a concern however worst case on a big data set should be completed running over a weekend. We will use runtimes of our most similar machine to the clients in order to test this and report them as part of our documentation process to set expectations. Any bugs found during testing will be tracked, assigned priority based on severity, and resolved before major milestones. We believe that this testing process will put the project in the best position to help our clients research as much as possible.