Real-Time Earthquake Monitoring

Mikołaj Malec 298828

1. Executive Summary

High-Level Description

The application provides a global, scalable view of earthquake activity by aggregating and clustering seismic events into meaningful regional insights. Instead of overwhelming users with millions of individual records, it visualises regional patterns, trends, and highlights only the most significant earthquakes. Real-time and recent events are presented to support situational awareness without compromising analytical clarity. The platform enables analysts, seismologists, and emergency operations officers to understand seismic activity quickly, consistently, and at the appropriate level of detail. ML-based clustering transforms high-volume, noisy seismic data into stable analytical units that enable spatial comparison, temporal trend analysis, and contextual interpretation of new events.

Progress Since PM2:

Since the previous milestone, the project has moved from architectural design to a fully functional implementation of both the real-time and batch processing pipelines. Key advancements include:

- historical data acquisition,
- ML training pipeline,
- region and cluster assignment stored in the serving layer,
- speed layer implementation.

Updates to PM2 Report

Revisions to Business Assumptions

There have been **no changes** to the business assumptions outlined in the PM2 report. The target audience remains defined as Analysts, Seismologists, and Emergency Operations Officers. The project scope continues to focus on the three core use cases:

- Understanding where earthquake activity is concentrated (Spatial Analysis).
- Understanding how seismic activity changes over time (Trend Analysis).
- Understanding current real-time events (Situational Awareness).

Revisions to Technical Assumptions & Architecture

There have been **no changes** to the technical foundation established in the previous milestone. The system continues to utilise the Lambda Architecture, employing Apache NiFi for ingestion, Apache Kafka for messaging, Apache Spark for processing (Speed and Batch layers), and HDFS/Hive for storage. The analytical approach remains focused on unsupervised learning (clustering) rather than predictive modelling, as justified in PM2.

Data Acquisition Strategy

Apache NiFi was **expanded** to support acquisition of the historical data (for example previous year data). A primary challenge in this process was managing the strict data retrieval limits imposed by the external providers. The USGS API caps responses at approximately 1,000 events per query and the Terraquake API structures historical data by month and limits the number of events per page, necessitating a complex pagination strategy to retrieve a full month's activity without gaps.

To overcome the USGS limitations the system implements a granular Day-by-Day Iteration strategy. The NiFi flow accepts user-defined Start and End dates and goes sequentially iterating through these 24-hour intervals.

For the Terraquake API, the system employs a nested loop approach. The logic operates on two levels: an outer temporal loop that iterates through the user requested time range month-by-month, and an inner pagination loop that inspects response headers for continuation flags. For each month, the system iteratively requests subsequent pages until the API indicates the end of that month's data.

The output of both APIs using already existing architecture to be digested.

Other
No new tools or data sources have been introduced. The project still relies on the integration of USGS and Terraquake APIs.

Exploratory Data Analysis (EDA)

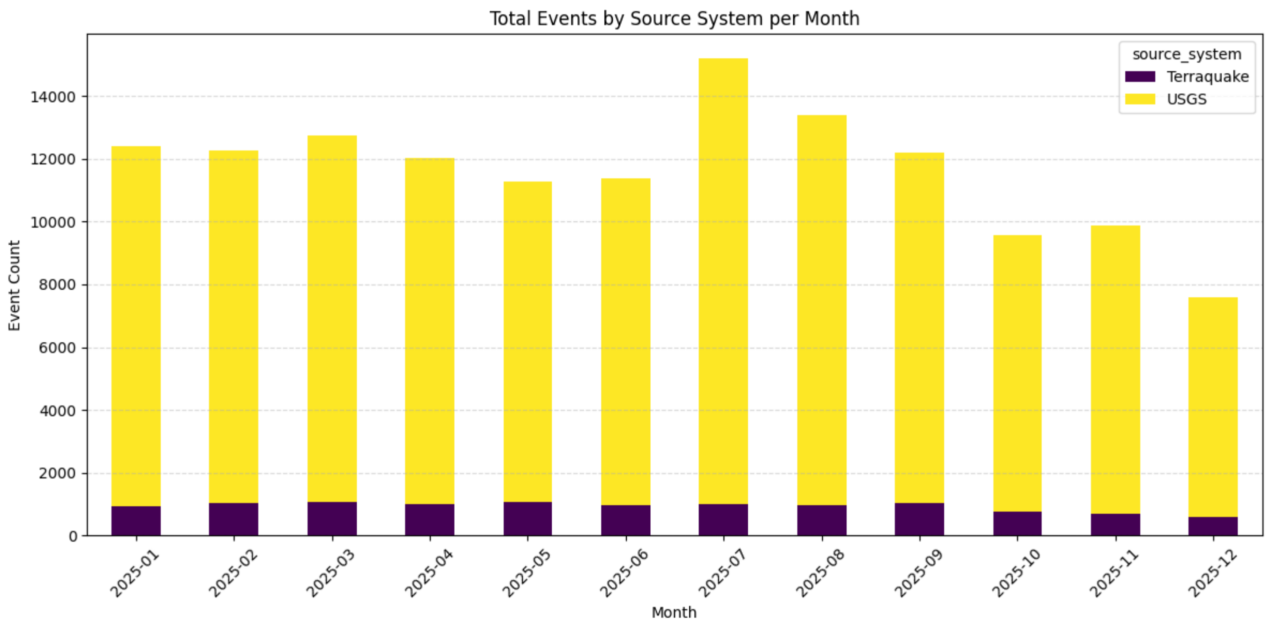
Clean dataset

This dataset is created after unification of the raw datasets from different sources. Serves as the "Single Source of Truth" for the system, on which later ML and data aggregation is created.

Data Volume and Sources

The dataset contains a total of 140,008 unique seismic events:

- USGS is the primary contributor, accounting for 128,977 events (92.1%).
- Terraquake contributes 11,031 events (7.9%), acting as a supplementary source.
- On average, the system records between 350 and 450 events per day.

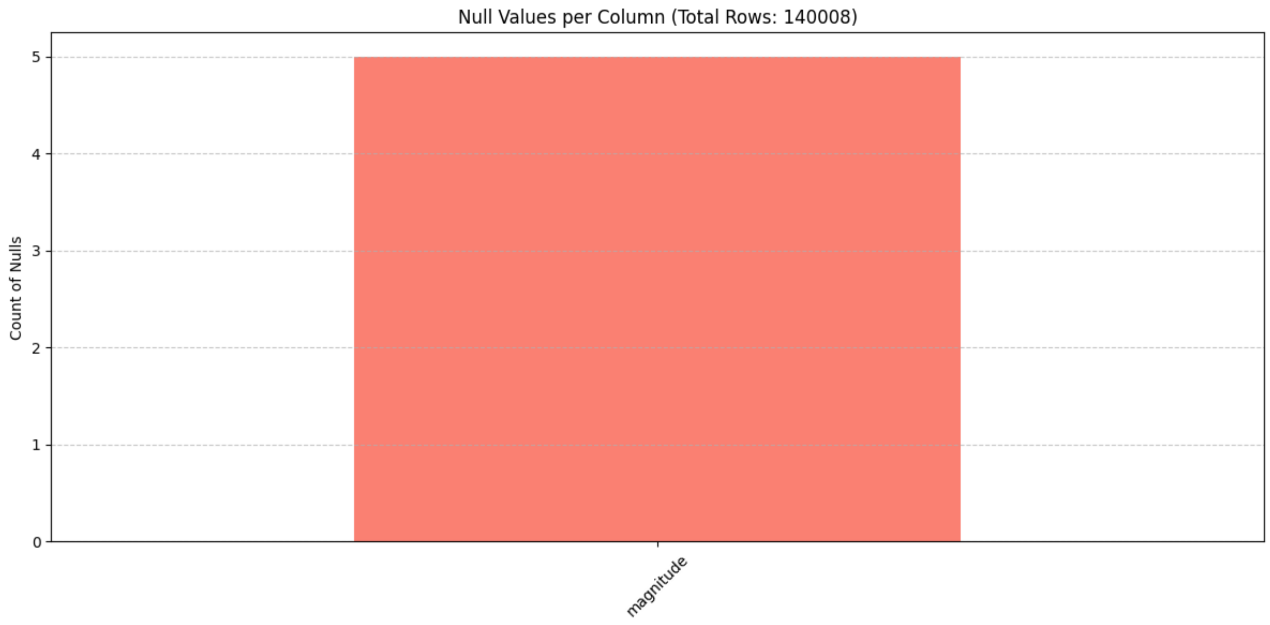


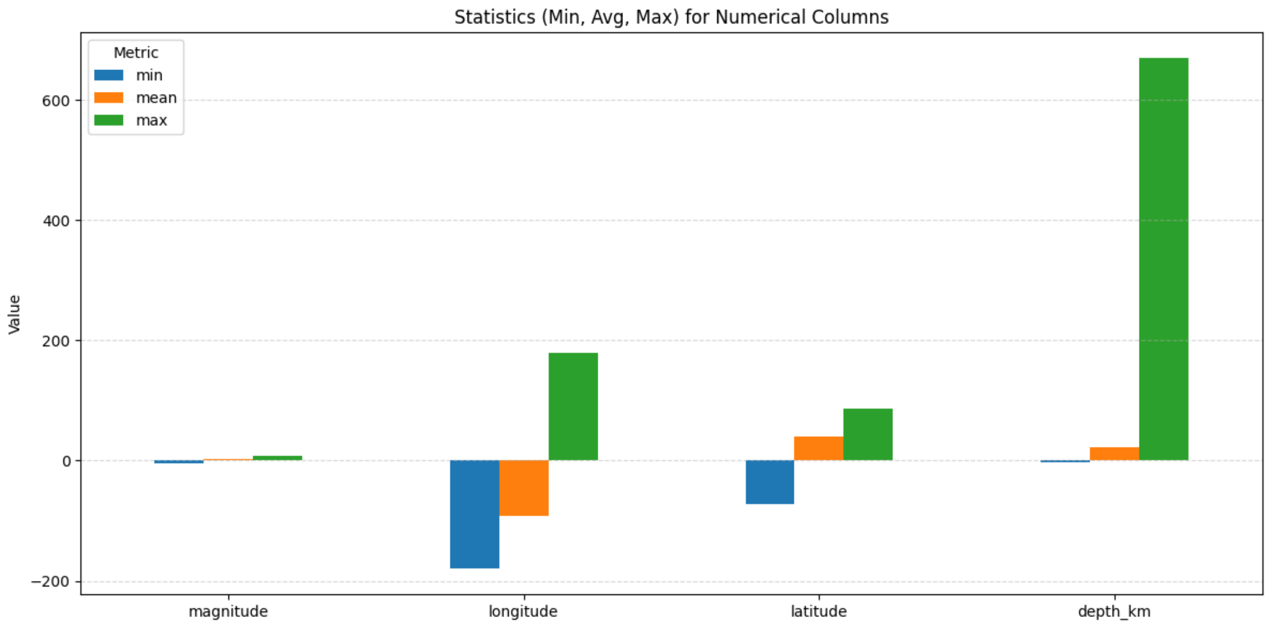
Data Quality and Completeness

The data quality in the Clean Layer is exceptionally high, validating the effectiveness of the ingestion and cleaning pipelines.

- Null Values: The dataset is effectively complete. Key columns (time_utc, latitude, longitude, depth_km) have 0 null values.
- Magnitude Completeness: Only 5 records out of 140,008 are missing magnitude data (< 0.003%), which is statistically negligible for clustering tasks.

- Spatial Integrity: Latitude and Longitude values fall within valid ranges (-90 to +90 and -180 to +180), confirming that no coordinate parsing errors occurred during unification.

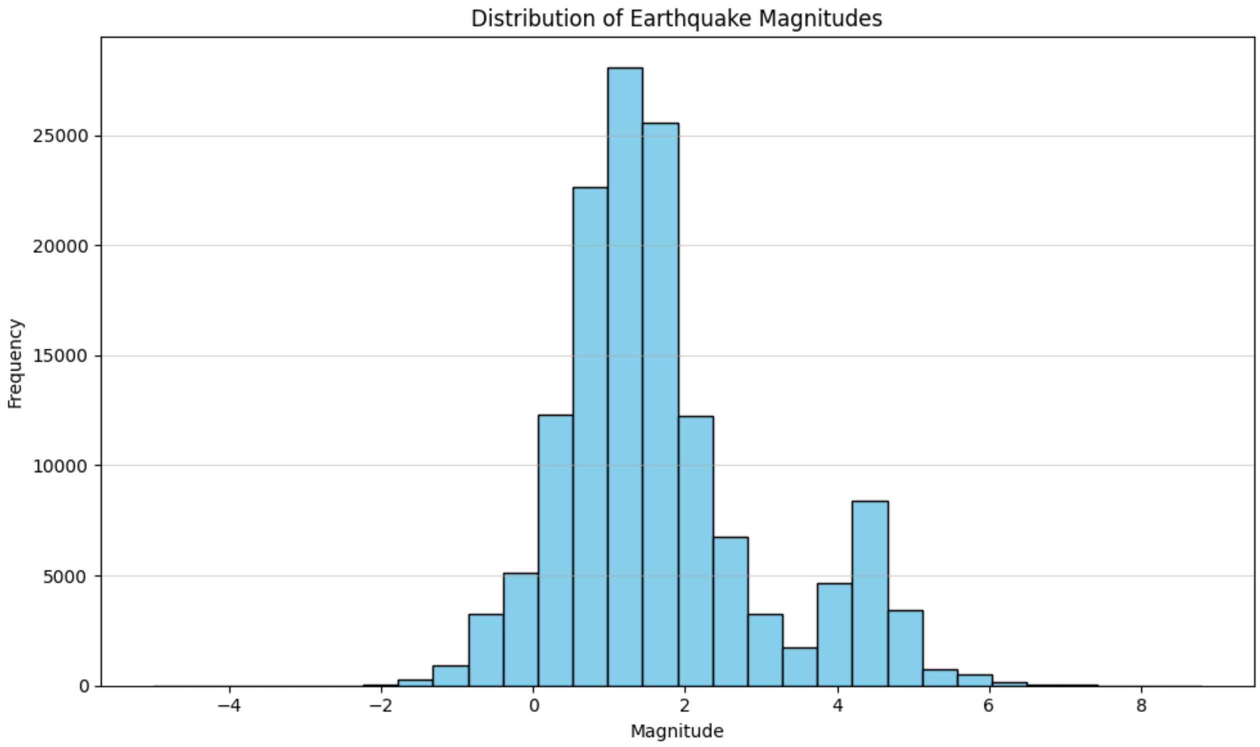




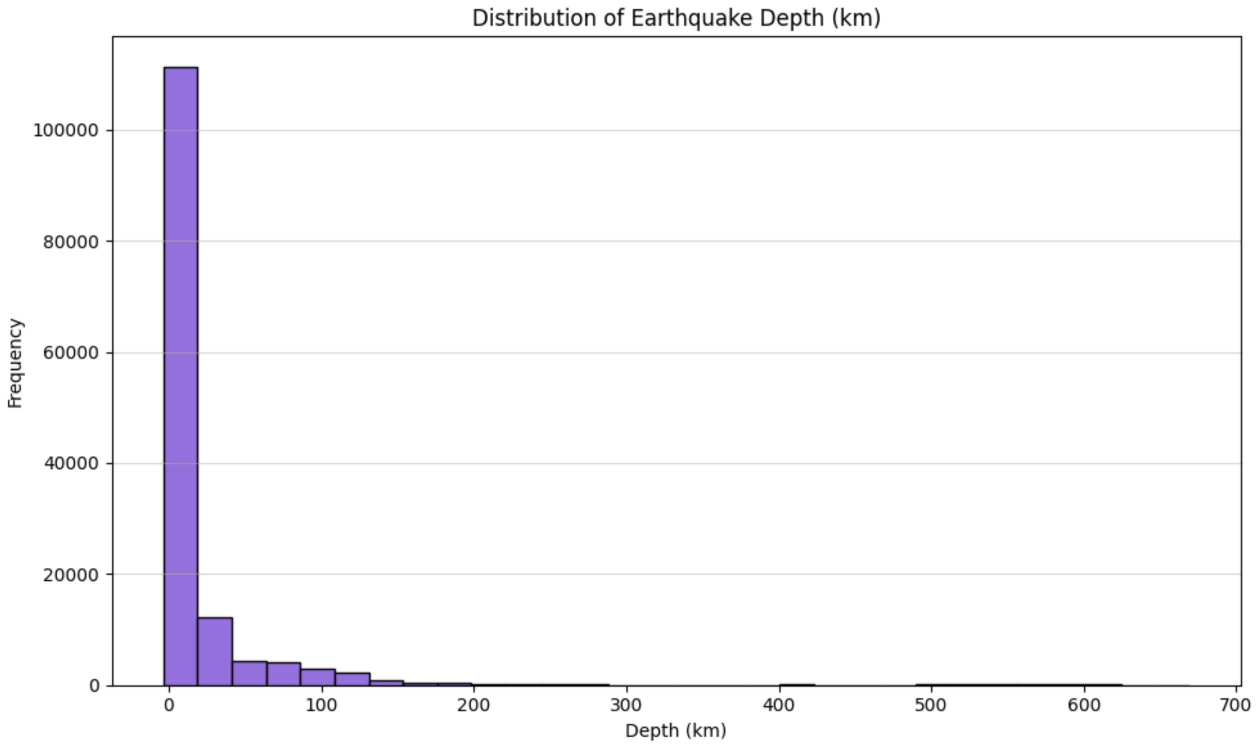
Statistical Distributions

Magnitude:

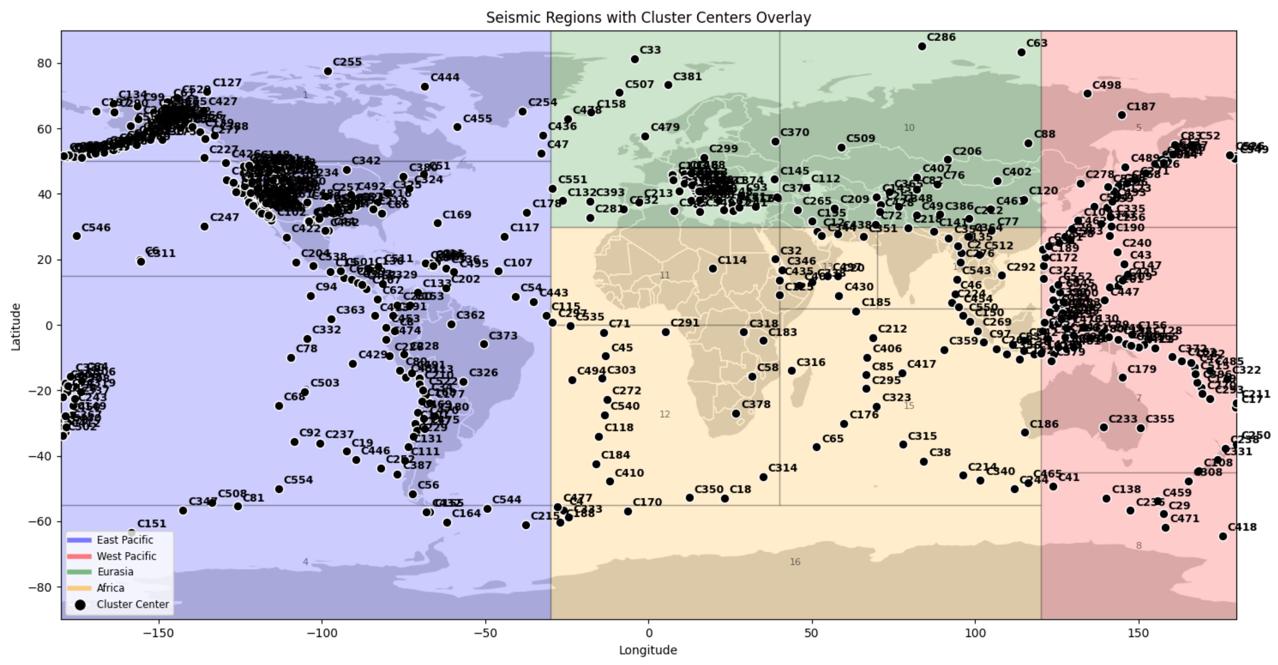
The presence of negative magnitudes is expected for very small "micro-earthquakes" detected by local sensors. The maximum of 8.8 indicates the system successfully captured major global events. Mean 1.67, suggesting that the vast majority of recorded activity consists of minor tremors, a standard distribution in seismology.



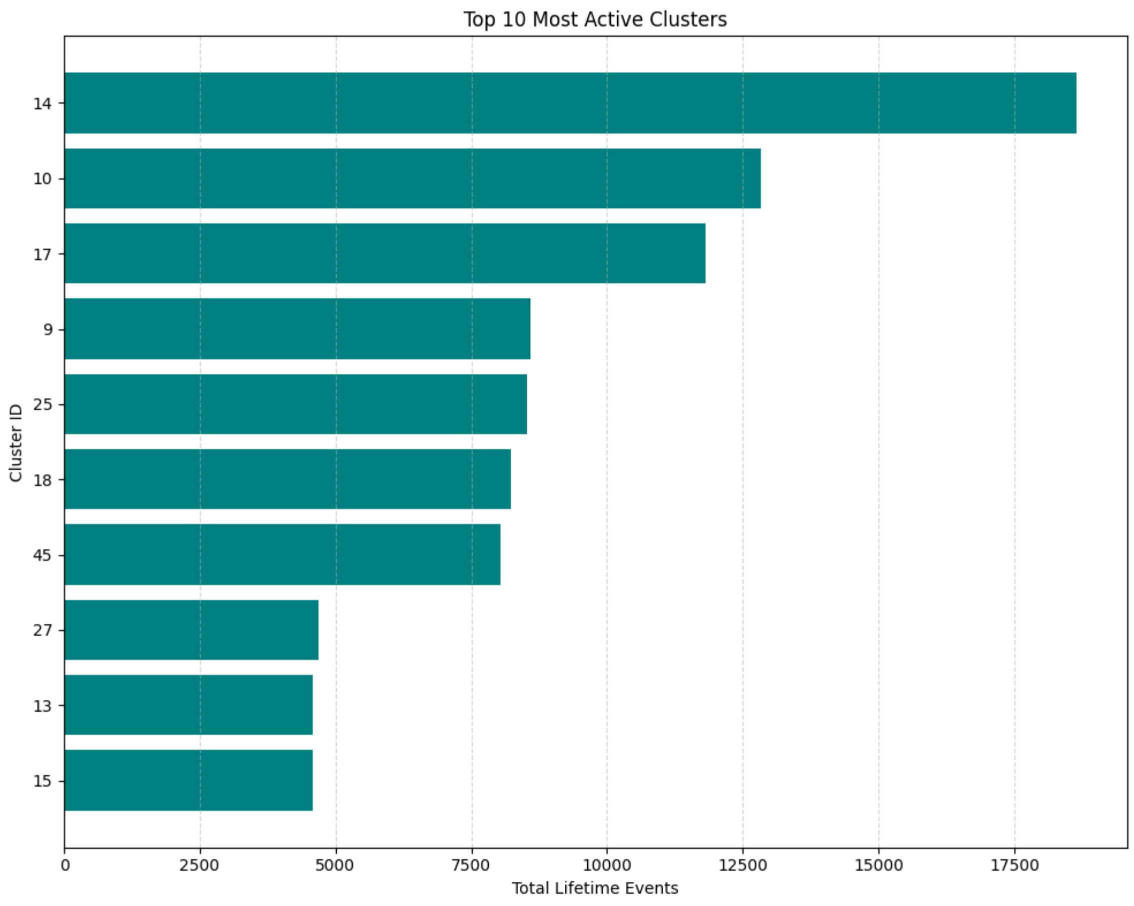
Depth:
The max depth of ~670 km confirms the dataset captures deep-focus earthquakes occurring in subduction zones (transition zone/lower mantle boundary). Negative depths indicate events occurring above the reference geoid (e.g., within mountains or volcanoes).

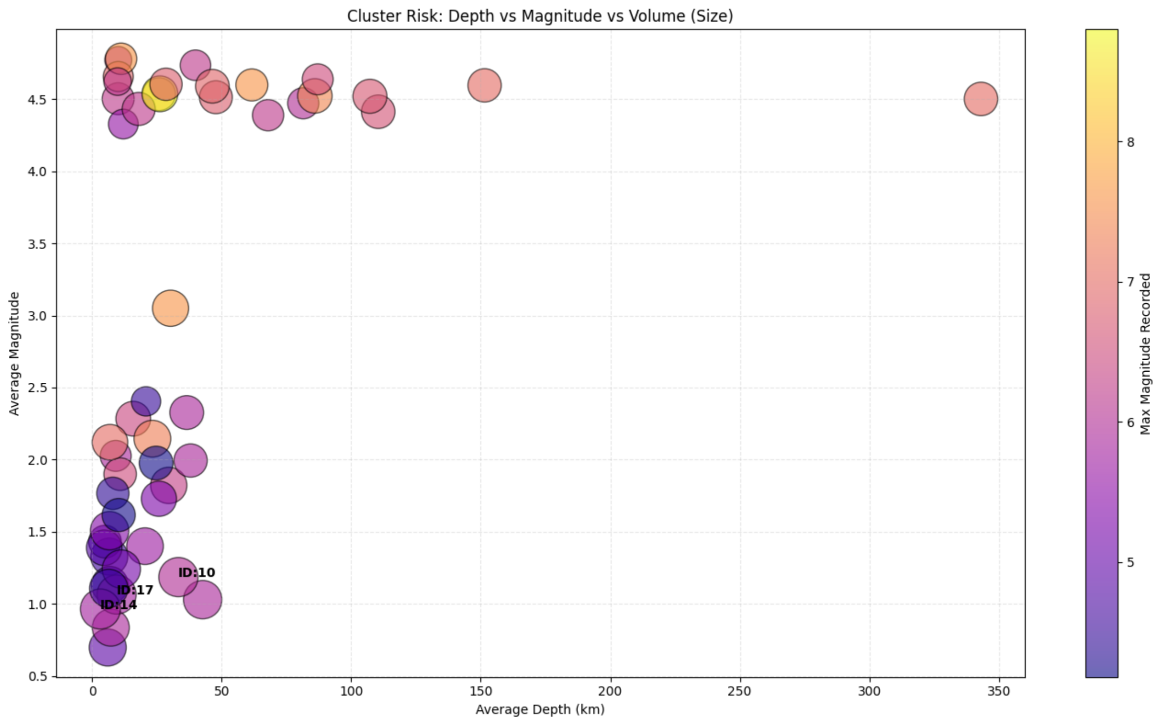


Clustering and regions map



Small analysis of clusters





System Architecture

The system implements a Lambda Architecture, designed to handle massive quantities of data by taking advantage of both batch-processing and stream-processing methods. This approach ensures a balance between latency, throughput, and fault tolerance. The architecture consists of five distinct layers:

1. Input Layer (Ingestion & Routing)

- Technologies: Apache NiFi, Apache Kafka
- Function: Apache NiFi acts as the entry point, scheduling API calls to USGS and Terraquake. It splits the data flow:
  - Hot Path: Sends standardised and cleaned JSON events to Apache Kafka topics to feed the Speed Layer.
  - Cold Path: Writes raw response files directly to HDFS (Raw dataset) for historical archiving and batch processing.
- Output: A real-time stream in Kafka and raw files in HDFS.

2. Speed Layer (Stream Processing)

- Technologies: Apache Spark Structured Streaming, Apache Cassandra
- Function: Real-time event records enriched with region and cluster assignment, stored in Apache Cassandra for fast retrieval.
- Output: Queryable views (Real-time Views)

3. Batch Layer (Batch Processing)

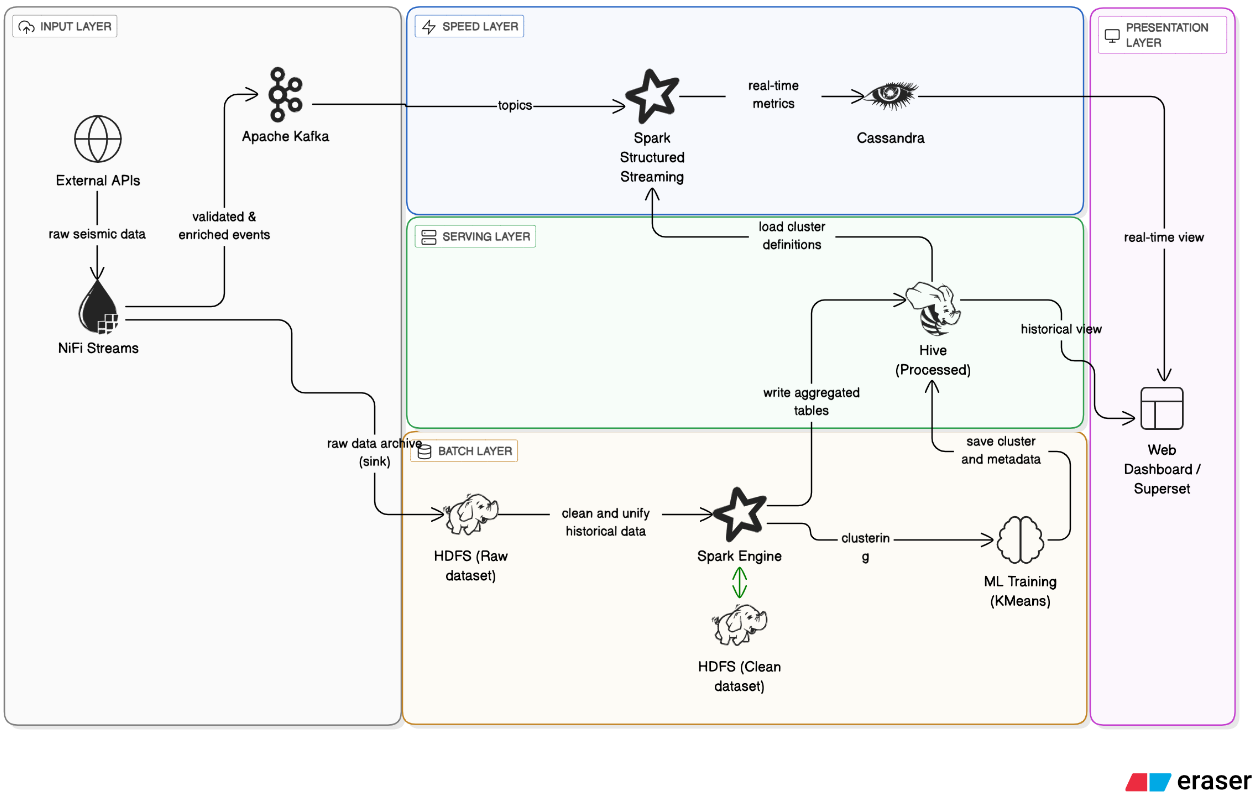
- Technologies: Apache Spark, PySpark, HDFS, Apache Hive
- Function: Managing the master dataset and heavy computational tasks.
  - ETL: Cleans, deduplicates, and merges raw data from HDFS into the Clean dataset.
  - Machine Learning: Retrains the K-Means clustering algorithm on the full Clean dataset to identify evolving seismic zones.
  - Aggregates Statistical Views of the Clean dataset based on the fixed regions and based on the clusters from ML training.
- Output: Aggregated Statistical Views (Processed), and the ML Clustering Model.

4. Serving Layer

- Technologies: Apache Hive, Apache Cassandra
- Function: Serves complex analytical queries on historical trends and regional statistics.
- Output: Queryable views (Historical Views).

5. Presentation Layer

- Technologies: Apache Superset
- Function: The visualization interface where data is consumed. It connects to the Speed Layer and Serving Layer to display:
  - Real-time Dashboard: Live map of events from Cassandra.
  - Analytical Dashboard: Long-term trend lines and cluster analysis from Hive.
- Output: Interactive dashboards providing actionable insights for Analysts, Seismologists, and Emergency Officers.



Data Preprocessing
Data preprocessing is executed through two parallel workflows to satisfy the competing requirements of historical depth and real-time speed. The **Batch processing** manages the rigorous cleaning and aggregation process, promoting data from the **Raw** stage to the **Clean** and **Aggregated** stages for deep analysis. Conversely, the **Speed processing** utilises a simplified, low-latency transformation pipeline to ingest and enrich streaming events for the real-time dashboard, bypassing heavy aggregation steps.

Batch processing

Ingestion Validation (Apache NiFi: Source to Raw)
Before data enters the HDFS Raw Layer, Apache NiFi verifies that every incoming event contains a valid **creation_time**. Since the storage strategy relies on partitioning data by date (dt), events missing this temporal reference cannot be correctly stored. Records failing this check are not discarded silently; they are routed to a Log processor.

Unification (PySpark: Raw to Clean)
The system first harmonises the divergent JSON structures from both APIs. Any provider-specific extra columns are dropped, and the data is cast to a strict common schema to ensure consistency. The final Clean Layer preserves only the following dimensions:
- id (String)
- magnitude (Double)
- mag_type (String)
- place (String)
- time_utc (BigInt)
- longitude (Double)
- latitude (Double)
- depth_km (Double)
- source_system (String)

- dt (Date)

Deduplication (PySpark: Raw to Clean)
There is high probability, that there will be duplication of the data stored in Raw Layer, also merging two independent data sources is also a challenge.

First, inner-source deduplication is performed independently within each data source. Duplicate records are removed by relying on each provider's own unique event identifiers (id), ensuring that no source contains repeated entries for the same reported event.

Second, cross-source deduplication is applied to identify and merge records that refer to the same real-world earthquake but originate from different providers. Because USGS and Terraquake use completely different identifiers, direct ID matching is not possible. Instead, the system uses a fuzzy matching approach based on a "bucket" strategy. Each event is assigned to a time bucket by dividing its timestamp by 60,000 milliseconds (one minute), so events occurring within the same minute are considered potential matches. In parallel, a geographic bucket is created by multiplying the latitude and longitude by 10 and flooring the result, grouping events that occur within approximately 0.1 degrees (about 11 kilometres) of each other. If an event from USGS and an event from Terraquake fall into the same time and geographic buckets, they are treated as the same physical earthquake. In cases where such a match is found, the USGS record is retained as the source of truth.

Aggregation Logic (PySpark: Clean to Aggregated)
The final stage of preprocessing occurs during the generation of the Aggregated Layer, where individual events are summarised by day into analytical statistics based on Clustering and the Region.

Before any aggregation takes place, the system performs a strict validity check on spatial coordinates. Since both the K-Means Clustering and the Region Spatial Join rely entirely on geospatial positioning, any record missing latitude or longitude are excluded from the final tables.

The system aggregates the validated data into two distinct statistical tables, each serving a specific analytical purpose:
- Fixed Region Aggregation: The system performs a spatial join between the earthquake events and a static region definitions table. This logic uses coordinate bounding boxes (Latitude Min/Max, Longitude Min/Max) to assign events to standard geographical zones (e.g., "Eurasia"). This ensures consistent, baseline reporting for known geological areas.
- Dynamic Cluster Aggregation: The system aggregates data based on the output of the machine learning process. Events are grouped by their assigned cluster_id, which represents a dynamically detected seismic hotspot.

Speed processing

Lightweight Unification (NiFi Jolt)
Upon ingestion, the stream undergoes an immediate structural transformation using a JoltTransformJSON processor in NiFi. This step standardises the JSON structure (removing nested objects) and drops extra or irrelevant columns immediately to reduce payload size. The output is dropped to the Kafka.

Stream Cleaning & Parsing (PySpark Structured Streaming)
The Kafka stream is consumed by PySpark, which applies robust parsing logic to handle variable timestamp formats (time_utc can appear as epoch milliseconds or string datetimes depending on the source), filters out records where day or id is null and assigns region and cluster based on the coordinates.

Storage & Deduplication (Apache Cassandra)
The enriched stream is written to Apache Cassandra. Cassandra is configured with the event id as the Primary Key, this inherently solves the problem of stream duplicates. If the same event ID

arrives twice, Cassandra simply overwrites the existing record, ensuring data consistency without complex read-before-write logic.
Since USGS and Terraquake assign different IDs to the same physical earthquake, Cassandra will store both records. In the context of the real-time dashboard, this is an acceptable design choice rather than a critical error. In fact, displaying both alerts provides researchers with valuable comparative data, allowing them to visualise how different agencies detect and estimate the magnitude of the same event in real-time.

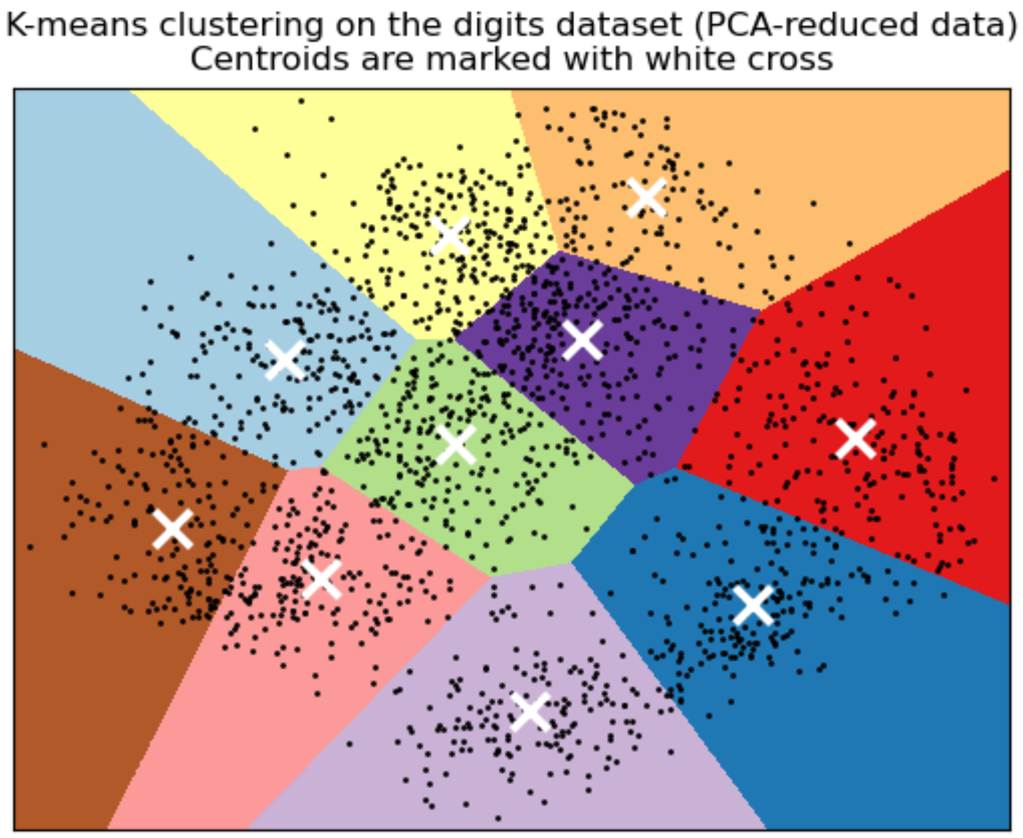
Analytical Module

The Analytical Module is responsible for identifying persistent seismic zones using unsupervised machine learning. The system utilises K-Means Clustering due to its efficiency with large datasets and interpretability.

Algorithm Overview: K-Means
The concept was first proposed by Hugo Steinhaus in 1956, though the standard iterative refinement technique used today is widely known as Lloyd’s Algorithm (proposed by Stuart Lloyd in 1957). The term "K-Means" was later coined by James MacQueen in 1967.

In the domain of seismology, K-Means is extensively used to partition earthquake catalogs into distinct tectonic regimes. Research has successfully applied this method to delineate shallow seismic source zones in complex areas. For instance, Weatherill & Burton (2009) demonstrated the efficacy of K-Means in modelling the highly fragmented fault systems of the Aegean region, while Novianti et al. (2017) utilised the algorithm to cluster earthquake epicentres in Indonesia to improve hazard mapping.

K-Means is exceptionally well-suited for large-scale distributed systems like Apache Spark. The algorithm is highly parallelizable because the most computationally intensive step, calculating the distance between every data point and the cluster centroids, can be performed independently for each point. In a distributed environment, worker nodes calculate these distances in parallel (Map phase) and then aggregate the results to update centroid positions (Reduce phase).



Example of k-means clustering
Image source: https://scikit-learn.org/stable/_images/sphx_glr_plot_kmeans_digits_001.png

Feature Engineering: Spherical Projection
A core challenge in global geospatial clustering is the distortion caused by standard Euclidean distance on a sphere (the Earth). Using raw Latitude/Longitude degrees works poorly near the poles and the International Date Line. For example, longitude -179° is numerically far from +179°, but geographically, they are neighbors. The Solution is 3D Vectorisation. The system converts spherical coordinates (Lat/Lon) into 3D Cartesian coordinates (x, y, z) on a unit sphere prior to model training.

- $x = \cos(lat) \cdot \cos(lon)$
- $y = \cos(lat) \cdot \sin(lon)$
- $z = \sin(lat)$

The clustering algorithm operates on these x, y, z vectors. This ensures that the distance calculated between points accurately reflects their physical proximity on the Earth's surface, solving the "Date Line" and "Polar Distortion" problems.

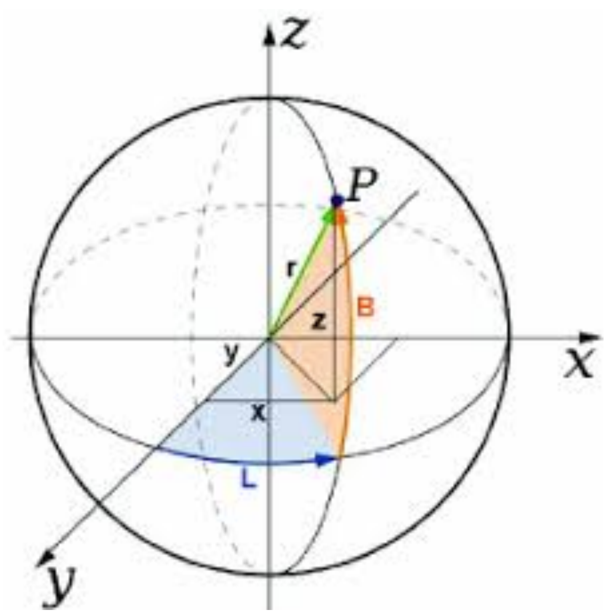


Image source: https://www.researchgate.net/figure/Three-dimensional-cartesian-x-y-z-and-spherical-r-radius-B-latitude-and-L_fig1_261217485

Training Strategy

The primary challenge of K-Means is that it requires the number of clusters k to be defined in advance. It does not "discover" the number of natural groups on its own. Additionally, the algorithm is sensitive to the initial random placement of centroids, which can lead to suboptimal local minima. To mitigate these issues, the analytical module implements automated k-tuning and stability trials.

- Iterative Hyper-parameter Tuning: The script iterates through a range of k (number of clusters) from 300 to 600 in steps of 20. This allows the system to empirically determine the optimal granularity for global seismic zones rather than guessing a fixed number.
- Stability Trials: To counter the sensitivity to initialisation, the algorithm runs 3 independent trials with different random seeds for each value of k. The trial with the lowest Within-Cluster Sum of Squares (WCSS) is selected as the representative model for that k.

Model Evaluation: Calinski-Harabasz Index

Standard PySpark evaluators (like Silhouette) are computationally expensive (O(N^2)). The system implements a custom, efficient calculation of the Calinski-Harabasz (CH) Index. The CH Index evaluates cluster validity by maximising inter-cluster dispersion (separation) while minimizing intra-cluster dispersion (compactness). The system compares the CH Index across all tested k values and automatically selects the model with the highest score as the "Global Best Model."

Post-Processing & Metadata Generation

Once the optimal clusters are identified, the system transforms the abstract mathematical output back into actionable geographical metadata:

- Centroid Back-Projection: The cluster centers (originally x, y, z vectors) are projected back to the Earth's surface to derive readable Latitude/Longitude coordinates.
- Radius Calculation: The system calculates the Great Circle Distance from the cluster centroid to the farthest point within that cluster to define the region's physical radius (max_r_km).

Project Planning

With the core backend infrastructure, machine learning pipeline, and data ingestion flows successfully implemented, the final phase of the project focuses on the Presentation Layer, system integration testing, and preparing the final deliverables. Also, more experiments to find the optimal number of clusters can be done (wider range and more granular).

Referenced Papers

- Weatherill, G., & Burton, P. W. (2009). "Delineation of shallow seismic source zones using K-means cluster analysis, with application to the Aegean region." Geophysical Journal International
- Novianti, P., et al. (2017). "K-Means cluster analysis in earthquake epicenter clustering." International Journal of Advances in Intelligent Informatics
- Steinhaus, H. (1956). "Sur la division des corps matériels en parties." Bulletin de l'Académie Polonaise des Sciences, Cl. III, Vol. IV, No. 12, pp. 801–804. Link to Paper (PDF)
- Lloyd, S. P. (1957). "Least square quantization in PCM." Bell Telephone Laboratories Paper. (Later reprinted in IEEE Transactions on Information Theory, 28(2):129-137, 1982)
- MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations." Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, pp. 281-297. University of California Press, Berkeley
- Caliński, T., & Harabasz, J. (1974). "A dendrite method for cluster analysis." Communications in Statistics - Theory and Methods, 3(1), 1-27