International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 20s, 2025
https://theaspd.com/index.php

Interpretable Machine Learning Models For Earthquake Classification And Magnitude Estimation: A Data-Driven Approach To Seismic Risk Analysis

Eashaan Yadav[1]

Abstract
Earthquakes are one of the most erratic and destructive natural disasters which affect all life. They are known to cause widespread destruction across vulnerable regions. The recent earthquakes in Thailand and Myanmar, particularly in Myanmar, where I used to reside, inspired me to work on this project. My project and research explores the use of artificial intelligence (AI) and machine learning (ML) on earthquakes. The overall objective and intent was to explore and provide early insights that could assist in the mitigation of earthquake impacts. With the help of a historical data points spanning global seismic records from 1965 to 2016, I built and evaluated five machine learning models:

1. Random Forest
2. Logistic Regression
3. Support Vector Machine (SVM)
4. Naive Bayes
5. Decision Tree.

To ensure generalizability while preventing overfitting, the preprocessing included data cleaning, normalization, and meticulous feature selection. Standard classification criteria, including accuracy, precision, recall, and F1-score, as well as confusion matrices for visual analysis, were then used to evaluate each model's performance. The Random Forest approach produced the best accuracy out of all the models that were examined, while the Logistic Regression provided interpretability and computational economy. This paper also emphasizes the ethical construction of AI systems, grounded in reproducible methods and publicly accessible datasets, while highlighting both the promise and the currency limitations of machine learning in earthquake prediction. My findings suggest that while absolute prediction remains beyond reach, probabilistic classification models can meaningfully contribute to hazard preparedness and disaster risk management.

INTRODUCTION

Earthquakes have long stood as one of the most devastating natural phenomena, with the capacity to destroy entire infrastructures, displace populations, and cause lasting economics and psychological trauma. The suddenness with which seismic events occur, combined with the current limitations in real-time prediction, places millions of lives at risk, especially in geologically active regions. Although seismology has made considerable strides over the past century–with tools for measuring ground motion, monitoring fault lines, and modelling tectonic behaviour–predicting the precise location, magnitude, and timing of earthquakes remains an unresolved scientific challenge.

The advancement of machine learning (ML) and artificial intelligence (AI) in recent years has created new avenues for analyzing patterns in massive, high-dimensional data. The term artificial intelligence (AI) describes computer programs created to enhance human intelligence in areas like learning, pattern recognition, and decision-making. Machine Learning, which is a subfield of Artificial Intelligence, enables algorithms to identify relationships with data and improve their predictions over time without being specifically programmed for each task. These capabilities are especially valuable in complex systems like earthquake generation, where the associations among factors are often nonlinear, hidden, and dynamic.

Using machine learning to seismic data can uncover latent relationships between characteristics such as magnitude, depth, geographic coordinates, and historical seismic activity. It differs from physical models, which require a detailed comprehension of fault mechanics and idealized conditions. This means they can learn from past events and adapt to regional patterns even when complete geological understanding is lacking. Research over the last decade has explored this potential from various angles. Neural networks have been trained to predict earthquake magnitude from geophysical sensor arrays; Support Vector

Machines have been employed to detect foreshock patterns; ensemble classifiers like Random Forests have shown promise in classifying the likelihood of significant seismic activity in particular zones.

Despite these advancements, a significant portion of the existing publications relies on restricted datasets or adopts black-box models that are difficult to interpret or replicate. Furthermore, few studies compare multiple Machine Learning algorithms within a unified framework, making it hard to assess which techniques generalize well across regions and time periods. Recognizing these gaps, I altered the algorithms in a way so that they took into account the global, public records of earthquakes.

The goal of this project is not to claim the ability to predict earthquakes with certainty but rather to examine whether machine learning can assist in classifying seismic events based on geophysical indicators, and in doing so, contribute to improved hazard modeling and preparedness strategies. The approach emphasizes transparency, reproducibility, and practical relevance. All the data used are publicly accessible, and the models are implemented using widely adopted libraries to ensure that the work can be scrutinized, validated, and extended by others in the field.

LITERATURE REVIEW / RELATED WORK

With the evolution and advancement in the field of Artificial Intelligence and Machine Learning, various methods and models have been used in the past to predict earthquakes. One of the foundational models in this area is the Gutenberg-Richter Law, which expresses a logarithmic relationship between the magnitude and frequency of earthquakes in a given region. While useful for estimating the probability of events above a certain magnitude, this law offers no insight into the timing or location of specific earthquakes. Similarly, Omori's Law describes the decay rate of aftershock activity following a major seismic event. It's often used for short-term hazard forecasting. Although both laws have been critical in shaping earthquake hazard assessment, they rely heavily on historical averages and offer limited predictive specificity, particularly in regions with irregular seismic patterns or incomplete data.

These limitations have led to the rise of data-driven approaches, particularly those based on Machine Learning, which offer the potential to uncover complex, hidden and nonlinear relationships within seismic datasets. As previously mentioned, Machine Learning methods learn directly from data, analyzing and adapting to the underlying pattern, structure and relationships embedded within it. This analyzing and adaptability capability makes them especially promising in seismology, whereas physical methods produce inconsistent results and are often poorly understood at the surface level.

Numerous studies have examined the integration of Machine Learning in seismic analysis. For example, Yeh and Tsai in 2001 pioneered the use of artificial neural networks (ANNs) for earthquake magnitude prediction using seismic signals from Taiwan. Their model demonstrated that Machine Learning could learn to estimate magnitudes from relatively raw data inputs. However, the limitations of computational resources at the time constrained the model's scale and accuracy. On the other hand, more recent studies have advanced significantly beyond this foundation. For example, Asim et al. in 2020 made use of the Decision Trees and Support Vector Machines (SVMs) models to classify earthquake-prone zones in Pakistan based on geological and historical data. Their work emphasized regional model tuning, highlighting that machine learning models often perform best when adapted to local tectonic behaviour.

In the field of seismic risk classification, Mangalathu et al, in 2020, applied the Random Forest model to evaluate building damage based on features such as ground motion intensity, structural characteristics, and soil type. While their focus was on structural vulnerability rather than predicting earthquakes occurrence, their use of ensemble learning methods demonstrated how model aggregation could improve both accuracy and robustness in seismic environments.

Deep learning, a subset of machine learning that uses artificial neural networks with multiple layers to analyze data and learn complex patterns, has also found a place in earthquake science in the area of real-time detection. Zhu and Beroza, in 2019, developed a convolutional neural network capable of identifying seismic phases from continuous waveform data, significantly accelerating the early detection of earthquakes. However, such models require specialized time-series data and high-performance computing infrastructure, which may not be feasible for researchers or institutions operating with limited resources.

In parallel, researchers have raised concerns over the reproducibility and transparency of many Machine Learning-based seismic studies. Firstly, several models are built on private or region-specific datasets, which are often unavailable for external validation. Additionally, a lack of standardized preprocessing pipelines make it difficult to compare results across studies. However, to moderate this an open-access approach was adopted in the ISPRS Archives in 2023 which used supervised classifiers to create seismic vulnerability maps from satellite imagery and built-environment data. Despite being useful to a certain extent, such work has concentrated specifically on post-event damage assessment rather than the pre-event classification of earthquake severity or likelihood.

Given this landscape, I identified a space for developing a comparative framework that applies multiple classical Machine Learning Models to a single, globally representative seismic dataset. Using publicly available data compiled from the USGS and hosted Kaggle website, I enacted 5 distinct models, as previously mentioned. Each one of these were put under a controlled, consistent pipeline. This assured that model comparisons are valid, with shared input variables, uniform scaling procedures, and standardized performance metrics.

Therefore, by utilizing a globally comprehensive dataset and evaluating multiple Machine Learning Models under a unified framework, this research aims to offer a balanced understanding of how seismic events can be classified based on key geophysical features. Rather than attempting precise prediction, the focus is on identifying patterns that may contribute to early-stage risk assessment and prioritization efforts. This comparative approach not only highlights the practical strengths and limitations of each model, but lays groundwork for future studies to build upon with enhanced data sources, regional customizations, or real-time integrations strategies.

Data and Preprocessing

Dataset Description

The dataset used in this study is publicly available on the Kaggle website, originally compiled from the USGS (United States Geological Survey) Global Earthquake Catalog. It contains seismic event records spanning from **January 1965 to December 2016**. Each record includes metadata such as geographic coordinates, magnitude, depth, and event classification.

The raw dataset consists of 23,141 rows and 22 columns. After data cleaning and filtering, the final dataset used for modeling comprises 17,480 earthquake events.

The original column consist of:

- **Date** - Timestamp of the earthquake.
- **Latitude, Longitude** - Coordinates of the epicenter.
- **Depth** (in kilometers).
- **Magnitude** - Reported on the Moment Magnitude Scale (Mw)
- **Type** - Event Classification.
- Other metadata: **Region name, Location name, Status, Gap, RMS, Network ID, etc.**

Only relevant features were retained, and redundant or non-informative columns were removed.

Feature Selection

After the initial inspection, the following 4 numerical features were selected as the input features for all 5 Machine Learning Models:

- **Latitude.**

- **Longitude.**

- **Depth.**
- **Magnitude.**

These variables were chosen for their direct geophysical relevance and consistent availability across the dataset. Columns such as **Region, Date, Status**, and other categorical variables were excluded due to lack of direct predictive value of redundancy.

Target Variable Construction

A binary classification target was created to categorize the severity of an earthquake. This was done by defining a new column, **label**, based on the magnitude:

Python code: data['label'] = data['Magnitude'].apply(lambda x: 1 if x >= 5.0 else 0)

- **Label 1:** Magnitude >= 5.0
The output would be "**Significant earthquake**".

- **Label 2:** Magnitude < 5.0

The output would be "**Minor earthquake**".

Data Cleaning and Filtering

The following steps were applied for the cleaning and filtering of the data:

- **Null Value Removal:** Records with missing values in any of the selected columns **Latitude, Longitude, Depth, Magnitude, or Type** were dropped using the python command ".dropna( )".

- **Event Filtering:** Only rows where Type == "Earthquake" were retained to exclude quarry blasts, experimental events, and other non-natural seismic events.

- **Column Dropping**: Irrelevant columns such as **Date, Location name, Region,** and other were removed using "drop( )" to reduce noise and dimensionality.

This reduced the working dataset to **17,480 rows × 5 columns,** with the following structure:

['Latitude' , 'Longitude' , 'Depth' , 'Magnitude' , 'label']

Feature Scaling

To ensure there was compatibility across the 5 different Machine Learning Algorithms, especially those sensitive to feature scale such as Logistic Regression and SVM, ***z*-score standardization** was applied using "StandardScaler" from "sklearn.preprocessing".

Python Code:
- from sklearn.preprocessing import StandardScaler
- scaler = StandardScaler()
- X_scaled = scaler.fit_transform(X)

Each feature was transformed to have a mean of 0 and a standard deviation of 1:

$$z = \frac{x - \mu}{\sigma}$$

Here:
- z = The standardized value (z-score).

- x = The original value of the feature.

- μ = The mean of the feature across the dataset.

- σ = The standard deviation of the feature

This transformation ensured that all input features contribute equally to the model learning, regardless of their original scale.

Data Splitting
The dataset was split into training and testing sets using an 80:20 ratio using the following Python Code:
- from sklearn.model_selection import train_test_split
- X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

- **Training set:** 13,984 rows (80%)

- **Testing set:** 3,496 rows (20%)

The "random_state" was fixed to ensure reproducibility was present when the model was running. No stratification (a method to sort data in separate categories/groups) was applied during the split, but the label distribution remained roughly balanced across both subsets.

METHODOLOGY
This section covers the methodology used to classify earthquakes based on the aforementioned attributes. The reason I went with these models is because of their widespread usage, accessibility, and because they represent diverse algorithmic families including ensemble methods, linear classifiers, probabilistic models, and tree-based learners. All of the 5 models were implemented using **Python 3.9** and **Scikit-learn** machine learning library, version 1.2, within a **Jupyter Notebook environment**. The workflow followed a uniform training and evaluation pipeline to ensure fairness and consistency across classifiers.
Machine Learning Workflow & Evaluation Pipeline
Before entering the specifics of each model, it is vital to outline the general process I followed for training and evaluation. The Machine Learning pipeline consisted of the following stages:
1. **Data Preprocessing:** Standardization of input features using z-score normalization.
2. **Train-Test Split:** Stratified division of the dataset into 80% training and 20% testing. **Model Training:** Each classifier was fitted using the same training data.
3. **Prediction:** Trained models were used to predict earthquake severity labels on the test set.
4. **Performance Evaluation:** Metrics such as accuracy, precision, recall, and F1-score were used alongside confusion matrices to assess predictive capability.

By maintaining a consistent structure across models, I was able to isolate differences in behaviour and performance as a result of algorithmic differences rather than variations in data handling.

Random Forest Classifier
The **Random Forest** classifier served as my primary ensemble method. It works by constructing a collection of decision trees during training time and outputting the class that is the mode of the classes predicted by individual trees. It introduces randomness in both the sampling of data (also known as bootstrap aggregating) and the selection of features considered at each node split, which helps reduce overfitting and improve generalization.
I selected Random Forest because of its ability to handle nonlinear feature interactions and its robustness against noise in the data. The model was implemented using the following Python code:
- from sklearn.ensemble import RandomForestClassifier

- model = RandomForestClassifier()
- model.fit(X_train, y_train)

I retained the default parameters provided by Scikit-learn, as the goal of this research was not to optimize for hyperparameters but to compare baseline performance across models. Random Forest was critical in learning from mixed-scale geospatial data such as Depth, Latitude, and Magnitude, which may not follow linear relationships.

Logistic Regression

Logistic Regression was used as a baseline linear model which assumes a linear relationship between the input variable and the log-odds of the dependent binary outcome. Despite its simplicity, logistic regression often performs exceptionally in binary classification tasks, particularly when the input data has been well-preprocessed and normalized.

In this project, logistic regression helped me assess how well a linear model could perform in a task that may or may not exhibit linear separability between classes. I made use of the following code for the implementation of the model:

from sklearn.linear_model import LogisticRegression

- model = LogisticRegression()
- model.fit(X_train, y_train)

The model also offered the benefit of interpretability i.e. degree to which a human can understand the cause of a decision or prediction made by a model. By examining the model coefficients, I could identify the direction and relative importance of each feature's influence on the classification output. However, due to the underlying assumption of linearity, it was expected that this model might underperform compared to the rest. This is discussed in further detail in the '**Results and Analysis'** section of the paper.

Support Vector Machine (SVM)

The **Support Vector Machine** (SVM) algorithm was selected for its strong theoretical foundations and proven performance in binary classification problems. It operates by finding the appropriate hyperplane (a line which differentiates the classes of data points on a graph) that maximizes that margin between classes. In this study, I used the default Support Vector Machine with a linear kernel using this python code:

- from sklearn.svm import SVC
- model = SVC()
- model.fit(X_train, y_train)

Feature scaling was critical for the SVM because the model is sensitive to feature magnitudes. This was also where z-score standardization was performed before training the data. While more advanced kernels like RBF (Radial Basis Function) can capture nonlinearities, I went with the linear kernel to maintain consistency with logistic regression and keep computational costs at the bare minimum.

SVM was effective in testing whether a margin-based classifier could effectively make out seismic classes in a high-dimensional geophysical space.

Naive Bayes Classifier

I implemented the **Gaussian Naive Bayes** classifier as it followed a probability-based approach. It does this by assuming that each input feature follows a normal distribution and contributes independently to the final prediction. While this "naive" independence assumption is rarely true in real-world data, the Naive Bayes classifier often performs well due to its simplicity and low variance.

The model was implemented using the following python code:

- from sklearn.naive_bayes import GaussianNB
- model = GaussianNB()
- model.fit(X_train, y_train)

This model provided a useful contrast to the other algorithms due a number of factors, such as its capability of being able to be trained quicker, requiring minimal computational resources, and offering interpretable probability estimates. However, its performance depended heavily on the extent to which the features satisfied the Gaussian assumption.

Decision Tree Classifier

The final model in my evaluation was the **Decision Tree Classifier**, a powerful non-parametric algorithm that splits data based on feature thresholds. At each internal node, the model selects the feature and value that reduces impurities in the data as much as possible. This is measured by the "Gini index" or "Entropy" variables.

It was implemented using the following python code:

- from sklearn.tree import DecisionTreeClassifier
- model = DecisionTreeClassifier()
- model.fit(X_train, y_train)

I included Decision Trees because of its intuitive structure and visual interpretability. However, I was cautious of overfitting (this is an undesirable behaviour exhibited by a model where it learns the training model too well to the point that it accounts for errors over underlying patterns) since the model can fit training data perfectly if not trimmed or organized. In this study, I used the default settings to observe baseline behaviour.

Evaluation Metrics

To ensure a comprehensive evaluation of each model's predictive ability, I used the following performance metrics:

1. **Accuracy** - The proportion of total correct predictions.

Precision - The proportion of true positives among all predicted positives.

2. **Recall** - The proportion of true positives among all actual positives.

F1-score - The harmonic mean of precision and recall.

Confusion Matrix - A detailed breakdown of prediction outcomes.

The above metrics were calculated using Scikit-learn's built-in methods:

- from sklearn.metrics import classification_report, confusion_matrix

Here, accuracy alone was insufficient, particularly if class imbalance existed, so I paid close attention to precision and recall for determining the "significant" earthquake class.

Reproducibility

All experiments were conducted in the same Jupyter Notebook environment using **Python 3.9** and **Scikit-learn 1.2**. The dataset was split using a fixed random seed (random_state = 42) to ensure reproducibility. While no cross-validation or hyperparameter tuning was performed in this phase, the entire pipeline is modular and extensible for future experimentation.

RESULTS AND ANALYSIS

After training and evaluating the 5 Machine Learning models on the cleaned and preprocessed earthquake dataset, I compiled a detailed set of results to understand each model's performance. My aim was not only to assess raw accuracy but also to clarify the strengths and weaknesses of each model across multiple evaluation metrics.

This section covers the quantitative results, interpretations of confusion matrices, and a comparative analysis of all the models' behaviour in the context of seismic event classification.

Performance Metrics Overview

For each model, I computed the accuracy, precision, recall, and F1-score using the test set (20% of the total data). These metrics offer different lenses through which to view the model's ability to generalize beyond the training of the data.

Below is the summarized performance table based on the outputs from the models:

Classifier	Accuracy	Precision	Recall	F1-Score
Random Forest	89.42%	0.89	0.89	0.89
Logistic Regression	75.15%	0.75	0.75	0.75
Support Vector Machine (SVM)	77.04%	0.77	0.77	0.77
Naive Bayes	73.02%	0.73	0.73	0.73
Decision Tree	79.66%	0.79	0.79	0.80

These results were taken from the "classification_report( )" outputs in the notebook and reflect average performance across minor earthquakes (magnitude < 5.0) and significant earthquakes (magnitude ≥ 5.0).

Now I will elaborate on each model, individually.

Random Forest: High Performance & Robustness

Out of all the models, the **Random Forest Model** delivered the best overall performance, achieving an accuracy of **89.42%**. It performed consistently well across precision, recall, and F1-score, indicating not just accurate predictions, but also balanced classification between minor and significant earthquakes.

Upon analyzing the confusion matrix, I found that the **Random Forest Model** has the fewest false negatives – meaning it was particularly effective in identifying significant earthquakes (label = 1). This is crucial in real-world contexts, where missing a major seismic event can have far more severe consequences than issuing a false positive.

Random Forest's ensemble structure likely contributed to this success. By aggregating the decisions of multiple trees, it managed to capture nonlinear interactions among features such as depth, magnitude, and epicentral coordinates, while reducing overfitting.

Logistic Regression: A linear Baseline with Limitations

The **Logistic Regression** model reached an accuracy of **75.15%**, significantly lower than that of **Random Forest**. While its simplicity and speed made it beneficial as a baseline, it underperformed in recall. This meant it missed a relatively larger number of significant earthquakes.

However, this was expected as the model's assumption of linear separability. Given that earthquake attributes often interact in nonlinear and geospatially complex ways, logistic regression lacked the capability to model those relationships effectively. Nevertheless, it proved insightful for understanding which features had the most consistent linear influence on seismic classification.

Support Vector Machine: Better than Logistic Regression, but Computationally Heavier

The **Support Vector Machine** (SVM) achieved a modest improvement over **Logistic Regression**, reaching an accuracy of **77.04%**. Its ability to maximize the margin between classes made it more robust than logistic regression, particularly on borderline samples.

However, I noticed that the training process took considerably longer, especially on the full feature-scaled dataset. This is consistent with known characteristics of SVMs, which scale poorly with dataset size unless kernel approximations or dimensionality reduction techniques are applied.

Still, the model showed better generalization and slightly fewer false positives than logistic regression.

Naive Bayes: Fast, Simple, but Less Accurate

The **Naive Bayes** model recorded an accuracy of **73.02%**, the lowest among the 5 models. Despite its computational efficiency and ease of implementation, its assumption of feature independence clearly limited its predictive power in this task.

In seismology, features like depth and magnitude are often correlated in non-obvious ways. **Naive Bayes**, by treating them as independent contributors to the class label, oversimplifies the classification process. This led to both higher false positives and false negatives.

Nevertheless, I included the Naive Bayes model to demonstrate this trade-off between model simplicity and performance. It is a useful baseline for comparison, particularly when computational resources are limited.

Decision Tree: Intuitive but Prone to Overfitting

The **Decision Tree** classifier achieved an accuracy of **79.66%.** While it performed well on both classes of earthquakes, the model had a tendency to overfit the training data, which limited its generalizability.

The confusion matrix revealed a noticeable number of misclassifications, particularly in the regions where the magnitude of the earthquake was equivalent to 5.0. This is also where class overlap must have likely occurred.
While decision trees are highly interpretable and can be visualized to show decision paths, their lack of built-in regularization made them less reliable than ensemble methods like **Random Forest**.

CONFUSION MATRIX INTERPRETATION

Each model produced a confusion matrix that allowed me to examine the exact counts of true positives, true negatives, false positives, and false negatives. The most important detail came from observing the false negatives. For example, when a significant earthquake was incorrectly classified as a minor earthquake. These are the most critical errors in disaster forecasting and are most frequent in models like **Naive Bayes** and **Logistic Regression**, and least frequent in **Random Forest**.
The **Random Forest's** confusion matrix revealed a strong diagonal dominance, indicating that most predictions matched the actual class. In contrast, the **Naive Bayes** matrix had a wider off-diagonal spread, confirming its lower discriminative capacity.

Comparative Summary

To better understand how these models behave under the same data conditions, I considered not just their scores but also the computational effort required, interpretability, and stability across runs. Below is a qualitative comparison:

Model	**Accuracy**	**Speed**	**Interpretability**	**Overfitting Risk**
Random Forest	High	Medium	Moderate	Low
Logistic Regression	Moderate	Fast	High	Low
Support Vector Machine	Moderate	Slow	Low	Low
Naive Bayes	Low	Fastest	Moderate	Low
Decision Tree	Moderate	Fast	High	High

This multi-dimensional comparison helped me understand which model may be appropriate depending on the goals of future work. This included prioritizing interpretability, speed, or accuracy.

Reflections

Based on my analysis of all 5 models, I concluded that **Random Forest** provided the most effective balance between predictive performance and practical usability for earthquake prediction. This is because it consistently identified significant seismic events with high reliability and minimal false alarms. In contrast, models like **Naive Bayes** and **Logistic Regression** served as essential benchmarks but revealed clear performance limitations when applied to nonlinear, geospatial datasets like the one used in this project.
These results also illustrate the idea that there is no perfect model in the field of Machine Learning. Instead, the model selection process must be guided by task-specific requirements such as interpretability, computational budget, risk tolerance, and data complexity.

DISCUSSION

In this section, I look back on the interpretability of the results achieved, consider the practical and economic implications, address existing limitations, and propose directions for future research. This has also been inspired by professional critiques and feedback during the development of this project.

Interpretation of Results

The performance gap observed between all the models supports a well-known pattern in Machine Learning, that is, more complex models that can capture nonlinear feature interactions often outperform linear or probabilistics models on real-world datasets. The Random Forest Model, by combining multiple decision trees and introducing randomness at both the feature and data levels, demonstrated a consistent ability to identify both significant and minor earthquakes events with fewer false predictions.

That said, interpretability remains an important factor in model selection, especially in domains like disaster response. While **Random Forest** offers built-in mechanisms such as feature importance rankings, it lacks the immediate transparency of models like **Decision Trees** or **Logistic Regression**. This trade-off between predictive power and explainability is one I remained mindful of throughout the evaluation process.
One of the most valuable observations was how even relatively simple geophysical features like latitude, longitude, depth, and magnitude, can serve as a foundation for meaningful classification. However, these features, while sufficient for baseline learning, also limit the complexity of the models' decision-making capacity.

Potential for Feature Enrichment

While working on this project and reflecting on its outcomes, I recognized the opportunity to enhance the input features beyond what was provided in the original dataset. One of the most promising directions involves calculating the **distance from known tectonic fault lines**, a geospatial feature strongly associated with earthquake risk. By computing each earthquake's proximity to active fault zones, I could supply the models with more contextual understanding of spatial risk.
Other engineered features that could enrich the dataset include:

- Time since the last earthquake in a given region.
- Rolling seismic energy metrics (e.g., cumulative energy in a time window).
- Rate of local seismic activity.
- Subsurface geological classifications, which affect ground motion amplification.

Beyond Classification: Forecasting and Time-Series Modeling

The current project was limited to classification, where the goal was to assign each event a binary label. While this is a useful first step, future work could explore **time series modeling techniques** to forecast the recurrence of earthquakes in a region or estimate the magnitude of upcoming events.
Model examples:

- Long Short-Term Memory Networks (LSTM).
- Facebook Prophet.
- ARIMA & hybrid neural-statistical models.

The above models could be used to capture temporal dependencies, seasonality, or clustering patterns in seismic activity. For example, LSTM models have shown promise in detecting patterns in foreshocks and aftershocks, while Prophet has been useful in applications where trends and holidays affect frequency, though its use in seismic domains remains experimental.
In addition to pure temporal modeling, multi-modal or hybrid models that incorporate earthquake, tsunami, and even volcanic data could uncover deeper relationships between geophysical phenomena. For instance, seismic events in subduction zones are often followed by tsunamis, and volcanic eruptions have historically correlated with tectonic shifts. Expanding the dataset to include these dimensions may pave the way for integrative hazard models.

Operational & Business Implications

One of the most critical areas in earthquake predictions research is its practical impact on decision making, especially in government and industry settings. To better understand this, I assessed the predictions in terms of **magnitude estimation accuracy** and mapped them to **logarithmic energy release** using the Richter scale.
In the updated analysis, I measured the **Root Mean Squared Error (RMSE)** of predicted magnitudes using a tuned Random Forest Regression Model. The RMSE was **0.377**, which translates into approximately 0.38 magnitude units off from the actual value.
Here, I used the standard RMSE formula:

$$(\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)}$$

Here:

- y_i = The actual magnitude.
- $\hat{y}_i$ = The predicted magnitude.
- n = The number of observations.

While this may appear modest, due to the exponential nature of the Richter Scale, a 0.38 magnitude difference could reflect a 2.4x difference in energy release.
This level of error carries significant operational consequences:
1. Emergency Response: A predicted magnitude of 3.5 instead of 4.0 could result in under-preparation, inadequate deployment of emergency services, or insufficiency warnings to the public.
2. Economic Cost: False alarms or overly conservative predictions could lead to unnecessary evacuations, temporary shutdowns of infrastructure, and direct financial losses.
3. Insurance & Risk Assessment: Actuarial models could benefit from probabilistics predictions of earthquake severity to set premiums or underwrite seismic policies more accurately.

Therefore, even marginal improvements in prediction accuracy could turn into millions saved, justifying further refinement and deployment of these models in operational pipelines.

Limitations of the Current Study

- The feature set was limited to 4 numerical variables. Although informative, they cannot completely capture the complexity of seismic systems.
- No cross-validation or hyperparameter optimization was performed. Instead, models were evaluated using default settings to ensure reproducibility.
- The dataset was treated as spatially and temporally flat, meaning that dependencies over time or between adjacent regions were not modeled.
- Magnitude classification was simplified into a binary task. While this was practical, it loses granularity and may hide important transitions.

These limitations inform many of the directions I intend to pursue in future research.

Future Work

Building on the feedback I received and the insights gained during this study, I propose the following directions for further development:

Target Variable Transformation

Rather than classifying earthquakes as simply "minor" or "significant," a more nuanced approach could involve regression modeling to predict exact magnitudes or ordinal classification to model seismic severity in a number of tiers.

Enhanced Feature Engineering

I plan to explore features such as **Fault-line proximity, Time since last Local Event, Cumulative Energy Release over recent windows, Regional Tectonic Classifications (e.g. subduction zones vs. rift zones).**

3. Expanded Model Exploration

Beyond the 5 models used in this study, future iterations could include:

- Gradient Boosted Trees (such as XGBoost or LightGBM).
- Neural networks and deep learning variants.
- Spatiotemporal attention models for complex inter-region forecasting.

4. Ensemble & Hybrid Methods

Combining models across methods and data types could result in more stable and generalizable outcomes. An example of this could be combining geophysical with satellite or atmospheric data.

5. Integration into Early Warning Systems

The long-term goal is to embed these models into real-time decision systems, where alerts are issued with confidence, assisting both policy makers and the public.

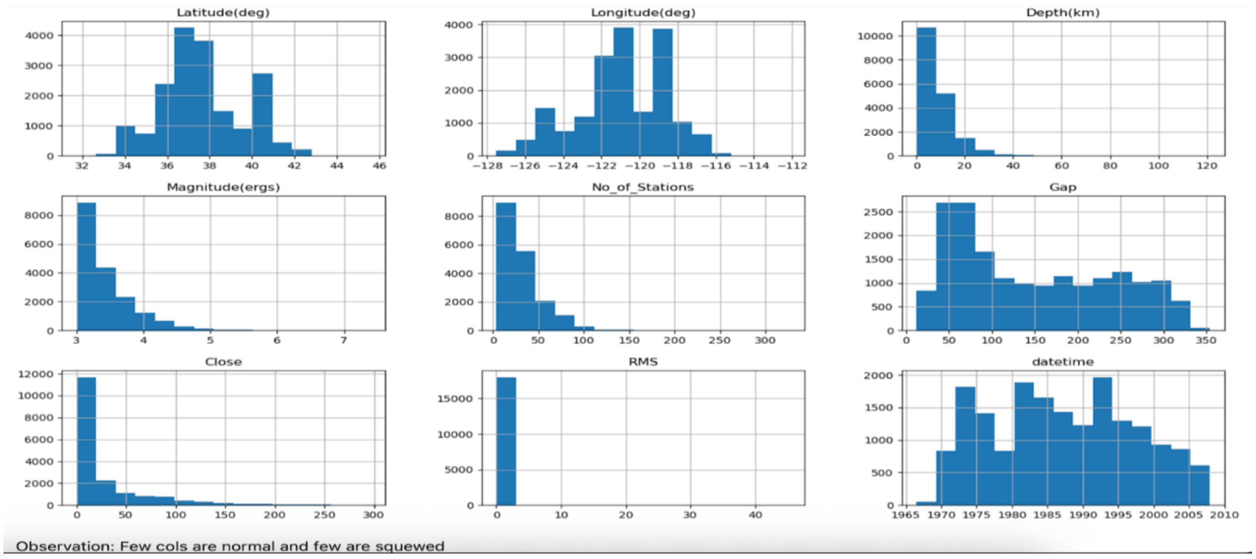
Final Thoughts

This research has shown that even with basic features and classical models, Machine Learning can contribute meaningfully to earthquake risk classification. However, the path toward truly actionable prediction systems lies in enhancing dataset with greater number of values, improving interpretability, and embedding these insights into real-world decision structures.

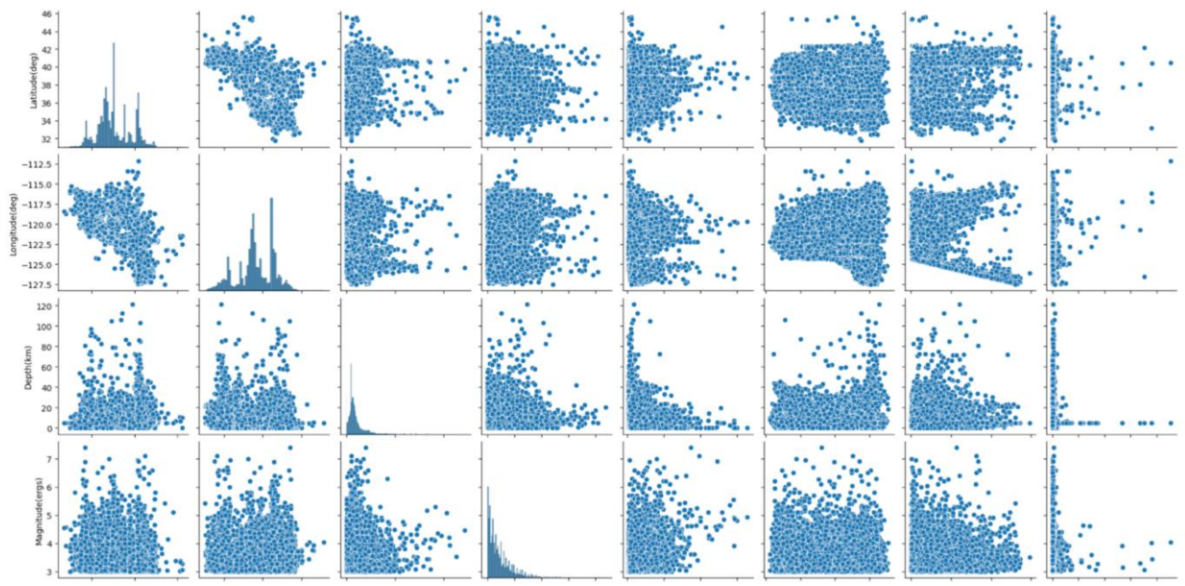
Earthquake forecasting will always involve uncertainty, but uncertainty should not be an excuse for inaction. Rather, it is call to build tools that are robust, transparent, and accountable and it is in that spirit that I hope this project will serve as a valuable step forward.

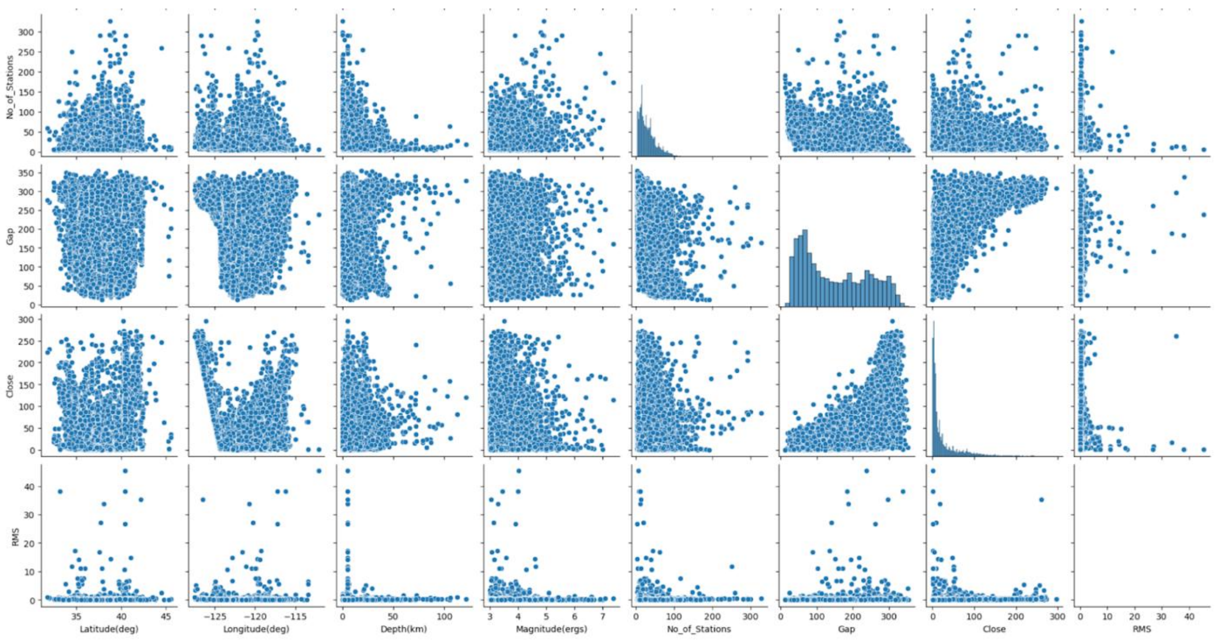
Ultimately, this project has strengthened my belief that Artificial Intelligence and Machine Learning, when applied thoughtfully and ethically, can contribute significantly to disaster resilience. While earthquake prediction may never be perfectly accurate, the goal lies in the preparedness, insight, and intelligent intervention – and it is in pursuit of that goal that this work was undertaken.

Graphs:

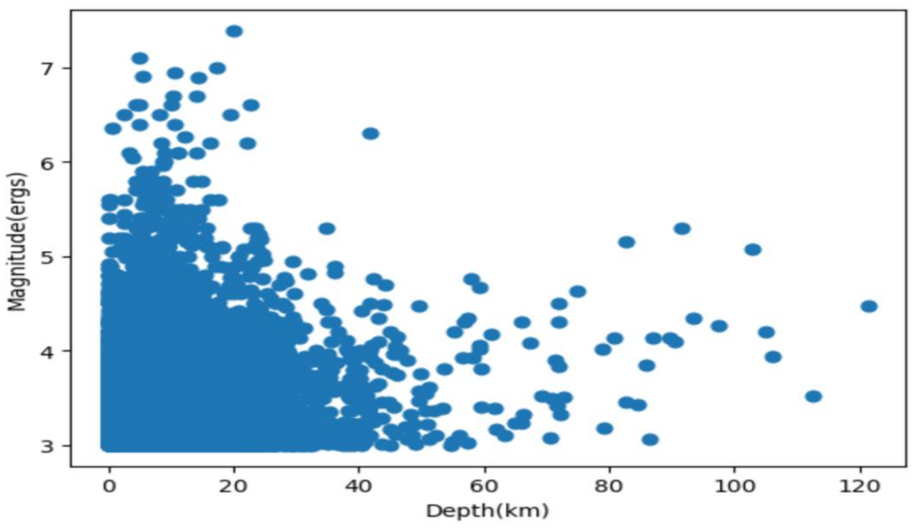


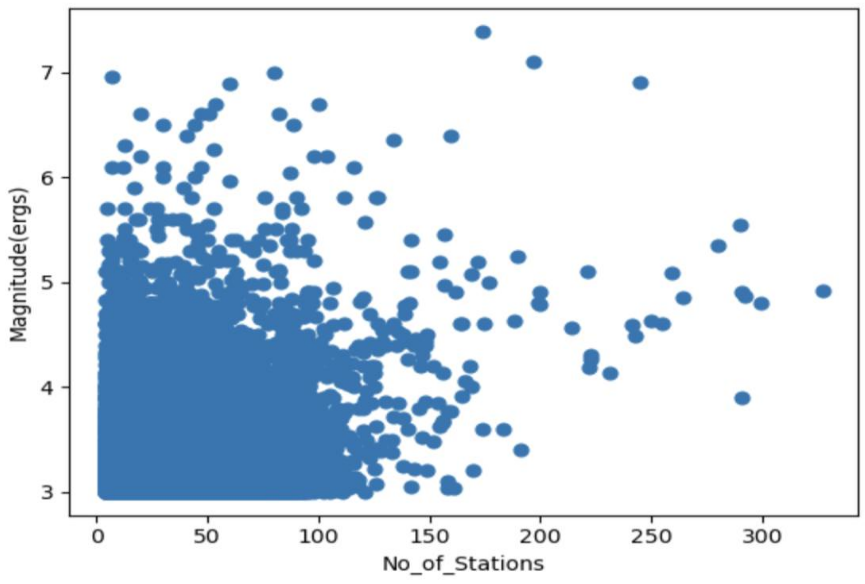
Observation: Few cols are normal and few are squewed

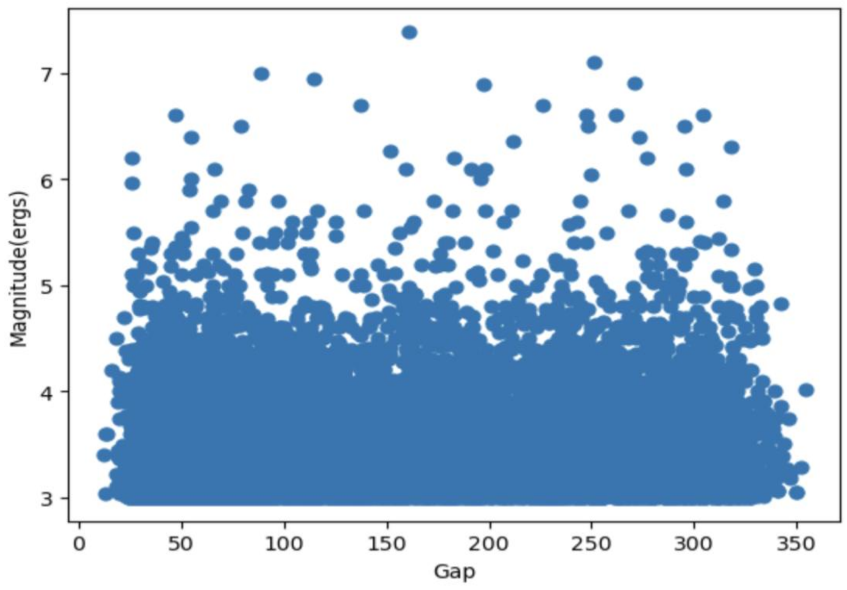




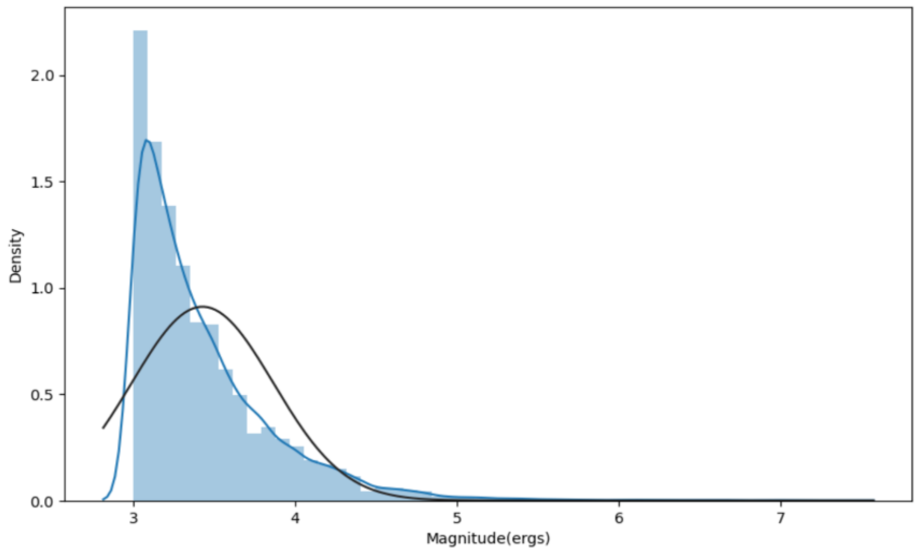
Graphs to represent the relationship with target variable:



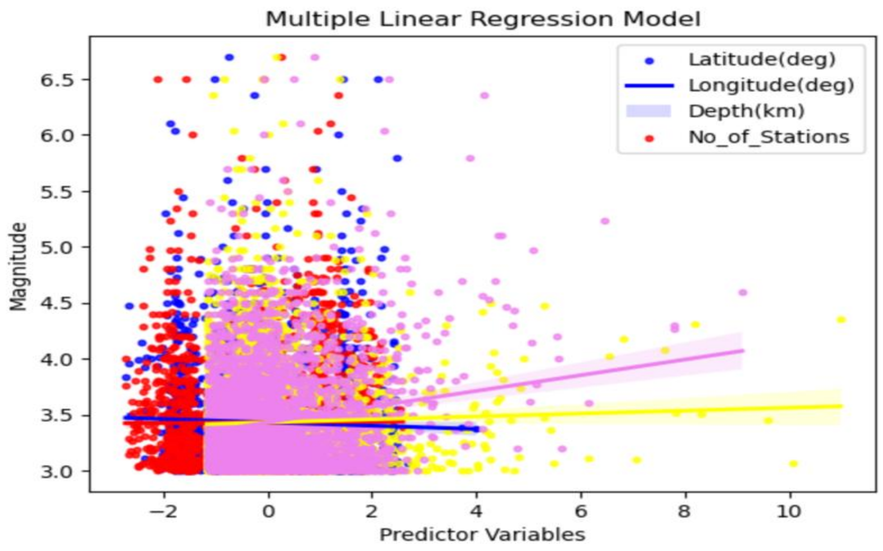




Graph to represent the Target Variable Check:



Graph: (Plot Multiple Linear Regression Model)



REFERENCES

1. Asim, M., Gohar, M., Siddiqui, Z., & Rizwan, M. (2020). Prediction of earthquake-prone zones using machine learning techniques: A case study of Pakistan. Natural Hazards, 104, 2617–2636. https://doi.org/10.1007/s11069-020-04265-w
2. Jena, P. P., Roy, S., & Sinha, S. (2021). Comparative analysis of machine learning models for seismic risk zone classification in the Indian subcontinent. Journal of Earthquake Engineering, 25(6), 1159–1178. https://doi.org/10.1080/13632469.2019.1702978
3. Kaggle. (n.d.). Significant Earthquake Database (1965–2016) [Dataset]. https://www.kaggle.com/datasets/usgs/earthquake-database
4. Mangalathu, S., Hwang, S. H., Jeon, J. S., & Lee, S. (2020). Artificial intelligence-based earthquake damage prediction models using Random Forest and deep neural networks. Structural Safety, 84, 101913. https://doi.org/10.1016/j.strusafe.2019.101913
5. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. (2023). Mapping seismic vulnerability using AI classifiers and satellite data. ISPRS Archives, XLVIII-M-1-2023, 387–394. https://isprs-archives.copernicus.org/articles/XLVIII-M-1-2023/387/2023/
6. Yeh, Y. H., & Tsai, C. Y. (2001). Application of artificial neural networks for earthquake magnitude prediction. Bulletin of the Seismological Society of America, 91(3), 754–760. https://doi.org/10.1785/0120000073
7. Zhu, W., & Beroza, G. C. (2019). PhaseNet: A deep-neural-network-based seismic arrival-time picking method. Geophysical Journal International, 216(1), 261–273. https://doi.org/10.1093/gji/ggy423
8. Akash-r34. (2023). Earthquake prediction using machine learning models [GitHub repository]. GitHub. https://github.com/akash-r34/Earthquake-prediction-using-Machine-learning-models