Timely Early Earthquake Prediction With Machine Learning

Jay Dattoo Dale, Renita Raymond Dsilva, and Sally Wei

Department of Applied Data Science, San Jose State University

DATA 270: Data Analytic Processes

Dr. Eduardo Chan

May 17, 2023

Abstract

Current seismology relies on either calculating statistics manually or monitoring known active regions of seismic activity to estimate earthquake likelihoods. These forecasting methods commonly involve estimating likelihood of a single, high magnitude earthquake within the timeframe of the next thirty years. Because of this multi-decade timespan, it is difficult for organizations and individuals to apply these predictions to long-term disaster preparation planning. This exploratory project applies three models, artificial neural network, random forest, and support vector machines, to develop predictions that narrow the timeframe from thirty years down to one decade. All three models are trained on a subset of the Southern California Data Earthquake Data Catalog, which includes over 420,000 seismic events that have been analyzed for magnitude, depth, location, and sensor quality within Southern California from 1932 to the present day. For this project, data prior to 1980 was excluded to avoid inconsistencies arising from the analog-to-digital archival process implemented in the 1970s. The data was aggregated over a rolling window of one, five, and ten years to produce the maximum magnitude and measures of central tendency and spread over each time period, then used to predict the maximum magnitude of earthquakes for each year. To assess the practicality of these models, each of these models was assessed for runtime and accuracy. The outcome of the project concludes that the SVM model has the highest accuracy, shortest runtime and the project includes a discussion of known limitations and directions for future research.

Timely Early Earthquake Prediction With Machine Learning

Earthquakes are considered one of the most devastating and destructive natural calamities. If an earthquake's magnitude is exceptionally strong, it can utterly destroy a city and destroy everything in its path. The earthquake's aftershocks are likewise quite devastating. Moreover, they can happen at any time, making prevention extremely challenging. Recovery from the earthquake's devastation might take months or even years. The ability to pre-emptively forecast earthquakes greatly increases the effectiveness of managing the risk of potential earthquakes.

Models for predicting earthquakes are proposed for development according to specific localities and will need to take into account for its predictions records of past earthquakes, geological data, earthquake activity patterns, and so on. The main objective of such an earthquake prediction model is to warn civilians and higher authorities of potentially dangerous earthquakes sufficiently in advance, allowing the appropriate precautionary measures to be taken. This will reduce deaths, minimize injuries, and prevent or guard against the destruction of infrastructure.

Forecasting earthquakes accurately is a difficult task that requires up-to-date information, where recent earthquake activity may require the complete remodeling of existing predictive models for the region (Jordan & Jones, 2010). It also requires the consideration of regional infrastructure, population density, and geological factors. Machine learning and neural networks can provide an avenue to building a flexible, responsive model out of such complex factors.

The goal of this project is to develop several prototype models for predicting earthquakes in advance using the machine learning models of support vector machines (SVM), artificial neural networks (ANN), and random forests to return accurate results in a timely manner. SVMs

are learning algorithms that work in real time and are used for classification and regression tasks that allow them to detect earthquake signals and can be trained on past seismic data to predict upcoming earthquakes. Artificial neural networks remove human-involved training in favor of network-learned parameters and can be used for classification, clustering, and other tasks. Random forests combine and average results across multiple decision trees to produce a more accurate result than any one single tree.

The approach taken in this project is to test three models that have previously produced good results in earthquake prediction and reduce their calculation times to a usable time frame for real time earthquake prediction. The expected contribution of this project would be to improve the performance of a set of machine learning models for predicting earthquakes accurately in a timely manner, a task that is currently challenging in the field due to the computational complexity of popular methods, such as convolutional neural networks or multi-step processes involving multiple models. This would allow accurate results to be produced in a timely manner, allowing predictions to reach people faster and help them evacuate buildings and prepare areas that are at risk of damage. This could potentially save lives and reduce the damage caused by earthquakes. These models can be applied to develop an early warning system for disaster management, infrastructure planning and for insurance companies.

Project Requirements

Several functional and AI-based requirements for earthquake prediction systems can be evaluated and measured. The specifications for the project outlined in this paper are as follows.

Functional Requirements

The final produced project aims to produce a rapid, lightweight, easily accessible model or set of models consuming real-time data to make earthquake predictions. The final prototype

program and its included models must be able to update in real-time or near-real-time with updated data from its data sources and supported earthquake catalogs. Thus, there is an emphasis on automation, including for retraining the model in the event of a large earthquake that changes the distribution of the original set of data the model was trained on. As a result, there needs to be an automatable data pipeline from the raw earthquake catalog or catalogs used or supported by the final application. The data pipeline must automate the process from the updating of the data catalog, to the data cleaning, pre-processing, and transformation steps, to produce a finished dataset ready for model consumption.

The final product and suite of models must be easily accessible from any device. This suggests making it available on the cloud or on the Internet, as well as allowing it to be downloaded and used offline on a local user device. As a result, not only does the data pipeline need to be automated, it needs to also be able to run on a simple device with little computing power, meaning the pipeline must be efficient in its reduction of data and computational requirements to lower the strain on the end-user's device.

The models' results, presentability, and usage methods must be easily understood and accessible to anyone in the area, regardless of technical or professional background, ranging from researchers, higher authorities, government, emergency or first responders, and the rescue team, down to the general public.

Aside from the portability required of the final pipeline, the process of cleaning data must be robust enough to account for possible changes in the source datasets, including changes like the unexpected addition or removal of features, changes in labeling or naming conventions, or even changes in storage location or accessibility of the original dataset. This can be achieved

through a monitoring system that checks for the availability of the necessary data and its features, and sends an alert in the case of a change of availability.

Finally, the final program should be able to integrate as a part of another, larger program, or to be able to be used in other software applications, for example as a standalone library. The application should provide an API with which to access its models, results, training, and testing results. This API should be publicly available to be used and built upon by third parties, and it should be robust to serve the traffic requested, or else to be able to be self-hosted. This requires the program providing access to the models to be robust and lightweight enough to not become a bottleneck in these other third-party applications, such as emergency response or disaster preparation programs which may be complex and have integrations with many similar, smaller programs.

AI-based Requirements

As mentioned earlier, an important part of this project is flexibility and responsiveness with regards to a constantly-updating source dataset of earthquake catalogs. As such, a flexible, automated pipeline will be built using python and the data-processing libraries pandas and numpy. These are powerful libraries for managing and manipulating large sets of data, and integrate easily or are often built upon by existing python machine learning libraries, which implies compatibility of the data pipeline and the machine learning models. This compatibility will simplify the pipeline, reducing the complexity involved in managing the pipeline and for making future updates to it.

The chosen programming language python is lightweight, flexible, and popular for both scripting and data processing. The lightweight nature of the language, which neither requires complex boilerplate nor enforces difficult-to-learn, opinionated coding schemas, means the final

pipeline from data source to machine learning models will be easier to maintain even as it grows more complex over time. This is important as there will be many sources of change requiring updates to the pipeline, including support of additional earthquake catalogs, changes to the original catalogs that must be handled for backwards-compatibility, and the addition of new models as well as version control for all of the above. The popularity of python as a programming language means it has a large wealth of useful libraries to pull from for tasks like data manipulation of complex or unusual data types, including raw seismic waveforms. Additionally, despite its lightweight nature, python also includes support for key functions like file manipulation and file management, in addition to complex usages like machine learning.

Building the pipeline entirely in python as a single program allows the entire structure to be provided as a single library of scripts. This can be leveraged as an API for other programs, or as a standalone, independent software, publicly downloadable for any supported end-user device. Python's popularity means a greater number of supported devices. Additionally, the entire program can be version controlled with git version control on GitHub to track changes and updates. Major working releases can be checkpointed through GitHub's built-in release structure, and it even supports syncing changes between datasets and models. This will allow for easy management of a complex, growing suite of models, pipelines, and supported data sources, as well as tracking its history of changes across separate contributors.

The support vector machine and random forests will be built using the python library scikit-learn.

The neural network models will make use of machine learning libraries supporting neural networks like TensorFlow and PyTorch. The two libraries provide similar capabilities, but TensorFlow offers built-in support for batching data, while PyTorch provides a python-like

structure for creating neural networks. As such, the PyTorch library will be used to define the structure of the neural networks and handle the training and testing processes, while TensorFlow will be leveraged for its TensorDataset class which groups raw data into logical datasets for further manipulation, and its DataLoader class, which abstracts and handles batching of data for iterative training.

All of the models produced will be deployed to the public using the same structure, via REST API, to make them openly available on the Internet. Therefore, there will not be a guarantee of the kind of end user device that will be used to access the model. To account for the variance in power of such devices attempting to access or use the model, the final generated models must be lightweight enough to run on auxiliary, edge devices with low computing power and storage capacities.

In addition, the models introduced should be able to be evaluated and validated using standard, common, easily recognized measures like the F1 score, precision, recall, and accuracy. This will allow the final performance of the model to be as easy to understand and as accessible as possible for the audience.

As the model will be based on real-time data from a live-updating earthquake catalog, the model needs to be able to detect and filter for appropriate features from its datasets, which may change over time.

Data Requirements

The data should include a history of earthquake events, and must include the time, depth, calculated magnitude, and coordinates of the earthquake at a minimum. Likely candidates are earthquake catalogs compiled by public agencies, which are trusted and can be expected to be maintained over long periods of time with a high quality of readings. The selected earthquake

catalogs must be available for non-commercial use and should be available to the public, as well as providing a method to access it repeatedly as it updates. The data sources should expect to have long-term support and ideally come from trusted organizations, which can be expected to continue to update the data sources in the near future, or have a long track record of maintaining the records.

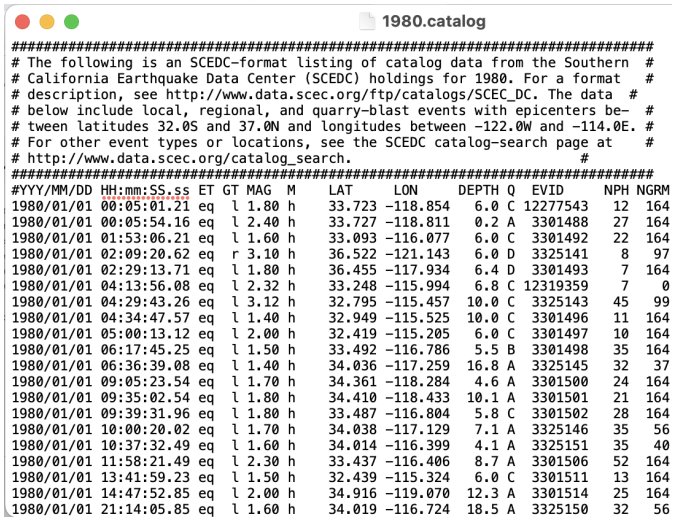
The selected earthquake catalogs should include a large enough corpus of historical data so as to provide enough data for model training. Large amounts of seismic data that can be utilized to train and test the system are needed in order to construct an earthquake prediction system. The data should have enough data for the selected, target regions over the targeted period of time. The data source will be insufficient even if it has a large corpus of data, if its data is spread thin in the targeted area over the selected time period.

The earthquake catalogs should include real-time data and live updates. It should include good coverage of the areas monitored in the catalog, so as to prevent the catalog from having an incomplete record of earthquake events in the region. An incomplete record could lead to skew in the dataset and thus a skew in the final, trained models, lowering the reliability of the models and the level of trust in the dataset.

The data must also be in a readable format for data processing, and complex, undocumented, or unsupported storage formats should be avoided to prevent the data source from becoming unmaintainable or unsupportable by the final program in the future. Datasets including raw waveform data or data collected from satellites or GPS must first be converted and analyzed for earthquake epicenter, depth, magnitude, and coordinates before they can be used for this project. Figure 1 below shows a sample of the SCEDC raw dataset which fulfills all of these requirements.

Figure 1

Sample of Raw SCEDC Catalog File for 1980



1980.catalog

```
################################################################################
# The following is an SCEDC-format listing of catalog data from the Southern  #
# California Earthquake Data Center (SCEDC) holdings for 1980. For a format   #
# description, see http://www.data.scec.org/ftp/catalogs/SCEC_DC. The data  #
# below include local, regional, and quarry-blast events with epicenters be-  #
# tween latitudes 32.0S and 37.0N and longitudes between -122.0W and -114.0E. #
# For other event types or locations, see the SCEDC catalog-search page at    #
# http://www.data.scec.org/catalog_search.                                    #
################################################################################
#YYY/MM/DD HH:mm:SS.ss ET GT MAG  M    LAT     LON     DEPTH Q  EVID     NPH NGRM
1980/01/01 00:05:01.21 eq  l 1.80 h    33.723 -118.854   6.0 C 12277543   12  164
1980/01/01 00:05:54.16 eq  l 2.40 h    33.727 -118.811   0.2 A  3301488   27  164
1980/01/01 01:53:06.21 eq  l 1.60 h    33.093 -116.077   6.0 C  3301492   22  164
1980/01/01 02:09:20.62 eq  r 3.10 h    36.522 -121.143   6.0 D  3325141    8   97
1980/01/01 02:29:13.71 eq  l 1.80 h    36.455 -117.934   6.4 D  3301493    7  164
1980/01/01 04:13:56.08 eq  l 2.32 h    33.248 -115.994   6.8 C 12319359    7    0
1980/01/01 04:29:43.26 eq  l 3.12 h    32.795 -115.457  10.0 C  3325143   45   99
1980/01/01 04:34:47.57 eq  l 1.40 h    32.949 -115.525  10.0 C  3301496   11  164
1980/01/01 05:00:13.12 eq  l 2.00 h    32.419 -115.205   6.0 C  3301497   10  164
1980/01/01 06:17:45.25 eq  l 1.50 h    33.492 -116.786   5.5 B  3301498   35  164
1980/01/01 06:36:39.08 eq  l 1.40 h    34.036 -117.259  16.8 A  3325145   32   37
1980/01/01 09:05:23.54 eq  l 1.70 h    34.361 -118.284   4.6 A  3301500   24  164
1980/01/01 09:35:02.54 eq  l 1.80 h    34.410 -118.433  10.1 A  3301501   21  164
1980/01/01 09:39:31.96 eq  l 1.80 h    33.487 -116.804   5.8 C  3301502   28  164
1980/01/01 10:00:20.02 eq  l 1.70 h    34.038 -117.129   7.1 A  3325146   35   56
1980/01/01 10:37:32.49 eq  l 1.60 h    34.014 -116.399   4.1 A  3325151   35   40
1980/01/01 11:58:21.49 eq  l 2.30 h    33.437 -116.406   8.7 A  3301506   52  164
1980/01/01 13:41:59.23 eq  l 1.50 h    32.439 -115.324   6.0 C  3301511   13  164
1980/01/01 14:47:52.85 eq  l 2.00 h    34.916 -119.070  12.3 A  3301514   25  164
1980/01/01 21:14:05.85 eq  l 1.60 h    34.019 -116.724  18.5 A  3325150   32   56
```

The data should include maintained metadata about quality that can be checked for potential causes of inaccuracy. Such metadata includes instrument types, installation locations, reading quality, and the time and date of record. It should also include notes about changes in the collection method of the raw data, should the dataset be maintained over a long period of time, so as to note changes in the data across certain time frames.

The current selected data source is the Southern California Earthquake Data Catalog, which includes 88 MB of data and over 80,000 rows of individual events tracked over more than forty years. Each year of data is saved into individual files, with later files trending larger than smaller files. As such, the expected storage requirements are at a minimum around 90 MB of storage for the raw dataset, plus growing requirements for each additional year of data as the catalog continues to grow and the program continues to pull and consume new data. The storage must also account for saving of the final transformed, processed data that will be used for training or retraining each of the models implemented.

Project Deliverables

Project Proposal

Earthquakes are natural disasters that can cause significant damage to life and property. Predicting earthquakes is a complex and challenging problem that requires the integration of multiple data sources and the use of advanced machine-learning techniques. The objective of this project is to develop a machine-learning model that can accurately predict the occurrence of earthquakes and their magnitude. To collect and analyze historical earthquake data from various sources.

To identify key factors that contribute to the occurrence of earthquakes and their magnitude. To develop a machine learning model that can accurately predict earthquakes and their magnitude. To compare the performance of the machine learning model with existing methods of earthquake prediction. To provide insights into the geological and seismological processes that lead to earthquakes.

Work Breakdown Structure:

The work breakdown structure for this project consists of four main phases: project initiation, data collection and preparation, machine learning model development, and reporting and dissemination, followed by project closure. In the first phase, the project's objectives are defined with scope and the project team and individual roles are decided.

The second phase involves identifying and collecting relevant earthquake data sources, cleaning and preprocessing the collected data, and conducting exploratory data analysis (EDA) and feature engineering.

The third phase focuses on selecting and implementing machine learning algorithms, training and optimizing the model, and evaluating its performance using appropriate metrics.

In the reporting and dissemination phase, reports and visualizations of our project findings are generated for presentation to technical audiences. A research paper can also be published at relevant academic conferences and journals to share findings with the wider scientific community.

Finally, the project closure phase involves conducting project evaluation and lessons learned analysis, archiving project documentation and deliverables, and presenting the project's challenge, approach, findings, and conclusions as the final delivery.

A Gantt chart and a PERT chart can be included with the work breakdown structure (WBS) to help plan and manage the project timeline. A Gantt chart is a type of bar chart that shows the project schedule, including the start and end dates of each task and their dependencies.

It provides a visual representation of the project timeline, allowing the project team to track progress, identify delays, and adjust the schedule accordingly.

A PERT chart, on the other hand, is a type of network diagram that shows the dependencies between tasks and their expected completion times. It helps the project team identify the critical path, which is the sequence of tasks that must be completed on time to ensure that the project is completed within the specified timeframe.

Data Management Plan

The data collection process will be carefully documented to ensure that all relevant data sources are identified and that data is collected in a consistent and standardized manner. All collected data will be securely stored in a centralized database. The database will be maintained by the project team and will be accessible to authorized personnel only. Data processing will be carried out using open-source tools, and all scripts used for data processing

will be documented and version controlled. Data processing steps will be recorded to ensure reproducibility.

The machine learning models used for the analysis will be trained on the processed data. The model evaluation will be conducted using statistical metrics, and the results will be recorded. Support Vector Machines, Random Forest, and Neural Network models will be used in the development of the earthquake prediction model. These models will be trained and optimized using the collected and processed data to achieve the best possible prediction accuracy for a timely response. The performance of each model will be evaluated using appropriate metrics, and the best-performing model will be selected as the final model.

The machine learning model developed will also be shared through an open-source repository such as GitHub. The project team will ensure that the collected data and associated documentation are preserved for long-term use. The data and documentation will be archived in a secure and durable format and will be accessible through appropriate repositories.

The project team will maintain ownership of the data collected during the project. The team will ensure that appropriate attribution is given to data sources, and data will be used only for the purposes of this project. The team will also ensure that any intellectual property arising from the project is managed appropriately.

The project team will ensure that all data collection and sharing activities are carried out in compliance with relevant laws and regulations. All data processing and sharing activities will be conducted in an ethical and responsible manner, ensuring the privacy and confidentiality of the individuals involved.

Data Collection and Preprocessing

During the data collection and preprocessing steps, the acquired data will be checked over for inconsistencies. This step involves deleting any errors, inconsistencies, or missing numbers from the acquired data. This will be accomplished with the help of data-cleansing technologies such as OpenRefine and pandas. Afterwards, the sanitized data will be combined from its various sources into a single dataset for analysis. The obtained data will be formatted appropriately for analysis. This requires transforming the data to a numerical format and normalizing the data. To reduce the dimensionality of the dataset and remove redundant features, data reduction techniques such as principal component analysis (PCA) may be employed. Finally, relevant characteristics will be extracted and engineered from the collected data. This involves finding the most influential features on earthquake occurrences and establishing new features based on existing ones.

The dataset will then be partitioned into training, validation, and testing sets. The training set will be used to train machine learning models, the validation set will be used to optimize the model's hyperparameters, and the testing set will be used to assess the final performance of the models.

Model Training and Prototype Development

The selected machine learning models for earthquake prediction will be SVM, Random Forest, and artificial neural networks. Each machine learning model will be trained using the training dataset, after which each of the models will discover independently the underlying patterns and associations between earthquake characteristics and the target variable. The performance of each machine learning model will then be evaluated using relevant evaluation metrics, including precision, and recall. This allows for the comparison and evaluation of the

model's ability to generalize using the validation dataset, as well as specify which parts of the task it may be better at or worse at.

The best-performed of the machine learning models will be selected for deployment in a production setting to forecast earthquakes. The model will be delivered as a REST API and hosted on a server in the cloud for scalability and accessibility.

System Architecture. The Data Ingestion component will be responsible for collecting and preprocessing the earthquake and seismic data from various sources. Feature Selection and Extraction components will be responsible for selecting and extracting the most informative features from the dataset. The machine learning models will include SVM, random forest, and neural network models that will be used to predict earthquake occurrence. The Prediction Engine will be responsible for predicting earthquake occurrence using machine learning models.

Prototype Development. The software components specified in the system architecture will be implemented using programming languages such as Python, Java, or R. The implemented components will be integrated to form a working prototype. The integration process will ensure that the components can communicate with each other and that the data flows correctly between them. The prototype will be tested to validate the functionality and reliability of the system. Testing will include unit testing, integration testing, and system testing.

Model Evaluation

For the testing dataset, performance of the selected machine learning model will be evaluated using appropriate evaluation measures such as accuracy, precision, and recall. Errors generated by the model will be investigated to determine the most challenging earthquake cases. This information can then be utilized to tune and further improve the models' performances, through methods such as fine-tuning or tweaking hyperparameters for the selected machine

learning model to improve its prediction accuracy. In the event that the source data's distribution changes drastically, such as through the introduction of a recent, high-magnitude earthquake, retraining the model with the new data may become necessary.

The goal of this model is to produce a speedy, quickly-retrainable set of models and related pipelines from the raw earthquake catalog to the final model. Therefore metrics related to speed are important for deciding whether the produced models are lightweight and fast enough to be deployed in a production environment, where the model may need to be retrained and updated with new data rapidly.

Final Report

The final report will include an overview of the project's goals, approach, solutions, and results. It will present a thorough assessment of the existing literature on earthquake prediction, including the latest research and advancements in the subject. Additionally, it will include the data gathering and preprocessing methods required to generate a high-quality earthquake prediction dataset.

Detailed information will be included, describing the machine learning models used to predict earthquakes and the training technique needed to optimize these models. Metrics will be presented following the model evaluation procedure used to evaluate the models' performance and the fine-tuning technique used to increase the models' precision. The project's outcomes will also be included, describing the performance of the machine learning models, the accuracy of the earthquake forecasts, and any insights gleaned from the analysis. Finally, it will conclude with a summary of the project's results and suggestions for future study in earthquake prediction.

Presentation

The final deliverable of the earthquake prediction project will be a presentation that provides a concise yet comprehensive overview of the project's main aspects. The presentation will cover the problem addressed, including the motivation for the project, the data collection and preprocessing steps used to create a high-quality dataset for earthquake prediction, the machine learning models used for earthquake prediction, and the training process used to optimize the models, the model evaluation process used to assess the models' performance, and the fine-tuning process used to improve the models' accuracy.

Additionally, the presentation will include a section on the results obtained, presenting the performance of the machine learning models and the accuracy of the earthquake predictions. Finally, the presentation will draw conclusions from the study, providing insights into the implications of the findings and recommendations for future research in earthquake prediction. The presentation will be designed to engage the audience, clearly convey the project's objectives and outcomes, and facilitate discussion and feedback.

A final table summarizing the deliverable due dates is provided below in Table 1. Each deliverable is spaced evenly across the span of roughly three months, with each deliverable having around one to two weeks between due dates. The process begins with the project proposal and ends with the final deliverables, the presentation and final report.

Table 1

Summary of Project Deliverables

Deliverable	Due Date
Project Proposal	February 24, 2023
Work Breakdown Structure	March 3, 2023
Data Management Plan	March 10, 2023
Data Preprocessing	Mar 24, 2023
Model Training and Prototype Development	April 9, 2023
Model Evaluation	April 23, 2023
Final Report	May 16, 2023
Presentation	May 12, 2023

Technology and Solution Survey

Earthquake forecasting has been attempted with statistical analyses, and in recent years have begun to integrate machine learning and automation into the process. This process involves the classification of incoming seismic waves into P-waves and S-waves to detect early warning signs on an earthquake. Models used to perform this clustering or classification of waves include convolutional neural networks (Pardo et al., 2021), fuzzy c-means clustering (Cano et al., 2021) or other clustering algorithms (Mendicelli et al., 2022), regression analysis using ground motion equations (Kwon & Elnashai, 2006) or support vector machines (Tezcan & Cheng, 2012), and random forests (Dumke & Berndt, 2019).

Velasco et al. (2022) chose to use a Bayesian block with a Fourier transform and modified a least-squares support vector machine (LS-SVM) to handle the non-linearity inherent to historical earthquake data. They noted that the usage of interpolation to fill in for a lack of

sufficient data in records of earthquake events tended to lead to an overestimation of smaller magnitudes and an underestimation of larger magnitudes. They modeled historical data surrounding major earthquakes in history and compared the results of their model's predictions with the predicted years of the historical earthquakes, and sought to explain the predicted time period for the given earthquake as produced by their model. Their reliance on Bayesian Machine Learning produces results with a measure of confidence, but did not lend itself to standard measures of accuracy like MAE. Instead, they performed a visual comparison of the earthquakes predicted by their model with the historical record of earthquakes produced for the given region and summarized their results for each region studied.

Console et al. (2002) used probability-based machine learning to predict the conditional probability of an earthquake happening given the historical data of earthquakes in the region. The probabilities of earthquake occurrences were simulated using Monte Carlo simulations, assuming the earthquake rate followed a Poisson process. This had the advantage of not requiring as much raw data, but had the downside of relying on a mathematical estimate of the regional earthquake activity rather than its real, historical record of activity. This approach applied several millennia of historical, paleontological data on historical earthquakes for each fault line surveyed in order to produce its probabilities, and produced predictions between a 90-99.5% level of confidence, with the level of confidence varying for each of the three regions under study. The study produced very low probabilities of future earthquakes, but involved statistical methods for validation and hypothesis testing, adding a level of rigor to their testing method.

Cano et al. (2021) used fuzzy c-means clustering to identify possible targets as part of their proposed multi-step process for automated wave picking, instead of manually labeling

waves for earthquake prediction. They found that such neural networks' performances are greatly dependent on having robust, good, and sufficiently large training data sets, and proposed a multi-step workflow involving separate fuzzy c-means (FCM) clustering and binary clustering steps to improve picking accuracy and reduce computing intensity. This was aimed at solving a fundamental issue in the lack of available training data for the earthquake prediction task, where the manual labeling of data served as a bottleneck to the amount of data available. Supervised machine learning methods involving classification or regression of waveforms previously required trained professionals to label a training set of data for the appropriate machine learning models. Semi-supervised or unsupervised models such as clustering algorithms or neural networks also needed labeled data for validation and accuracy calculations. They found their proposed model of clustering was able to accurately identify targets even at high noise-to-target ratios, common for earthquake data where the target earthquake reading is relatively rare compared to readings of regular, non-earthquake seismic activity. They compared against an existing STA/LTA model to measure the performance of their proposed model and compare the results of picks for both simulated and real data. They found that their model lost 221 picks, compared with STA/LTA which only lost 153 picks. They concluded this was due to sensitivity of their model to noise, which buried the picks and caused the model to overlook picks.

Pardo et al. (2021) used convolution neural networks (CNN) to select seismic phases, and they described two different approaches for selection. The first was to guess the middle of a phase given data, and then classify the type of wave of the identified phase. The second is to split the continuous feed of data into four-second chunks and use regression to guess what phase each chunk of data is. Of the two approaches, they took the first approach, which identifies locations of possible target waves and classifies them in a separate step, and combined both results with a

vote. They found that the combination of the two results resulted in a higher accuracy than taking the results individually. However, they found their process to be unwieldy as it involved pre-sorting through a large amount of noise in the data and filtering them for readings of one of the two target classes of waves This proved to be very time-consuming and relied upon pre-categorization of the waves either by hand or as provided in their data.

Dumke & Berndt (2019) tackled the issue of a lack of sufficient, appropriate data for making earthquake predictions and aimed to solve this using machine learning applied to global borehole data to predict for any given marine location, a model of P-wave velocity. The generated P-wave velocity could then be used in further machine learning or other mathematical models to calculate other earthquake predictors, resolving the issue of lack of suitable data. They accomplished this with a random forest of 100 trees for predicting P-wave velocity given global borehole data, noting that the performance improved as the number of trees increased past 50. They validated their trees with 10-fold cross validation and a leave-one-out approach, allowing them to train their data in partitions based on location. They found that using this approach, over 60% of their over 300 selected locations resulted with better model-predicted velocities over traditionally calculated velocities based on RMSE and MAE scores.

Mendicelli et al. (2022) chose to predict earthquakes through estimating peak ground velocity rather than through wave-picking and thus had access to instrument readings of their targets, peak ground acceleration and peak ground velocity, that they could compare their model's results with. They tackled the problem of increasing the accuracy of earthquake prediction when lacking a sufficient or complete history of earthquake events in the region, which could be partially solved by supplementing the missing data with maps of topological information like ground velocity. They followed the IGAG_20 methodology of clustering in

three distinct intervals and compared their results against existing site-specific Local Seismic Site Response (LSSR) results using MAE. They found their model tended to overestimate both peak ground acceleration and peak ground velocity, but was otherwise satisfactory in matching the results of the established LSSR method.

Mizrahi et al. (2021) tackle the problem of data incompleteness in historical earthquake records by proposing an alternative model to the popular epidemic-type aftershock sequence (ETAS) model for earthquake prediction. They proposed a method of accounting for earthquake completeness in models, and compared it against the ETAS null model for predicting earthquakes. They found that in earthquake catalogs with varying completeness over time, their decision tree was able to produce better prediction results and extract a better information gain from the inconsistent earthquake catalog than the ETAS null model. They additionally measured the amount of detected versus undetected events as a measurement of accuracy and concluded that their model had a lesser number of undetected events compared with the ETAS null model.

Xiong et al. (2021) supplemented the lack of seismic data with satellite data and employed the use of Inverse Boosting Pruning Trees (IBPT) in an AdaBoost ensemble to make short-term forecasts. They used the Dynamic Time-Warping (DTW) algorithm for clustering time-series data, using a Davies-Bouldin Index to select the ideal number of target clusters for each dataset. Their clustering resulted in four final datasets, which were normalized using z-score normalization and clustered using a sliding window to produce their final time series data. They compared the results of their convolutional neural network (CNN) with the results of eight other benchmarks using area under the curve (AUC) and receiver operating characteristic (ROC) curves, and were able to produce results better than the eight other models they selected as their benchmarks. However, they noted that their method involves significant computation

resources, while also lacking a guarantee of the optimal result, adding uncertainty to their predictions.

Debnath et al. (2021) compared the results of a Bayesian network, a random forest, a logistic regression, and a Logistic Model Tree (LMT) for the task of earthquake prediction using the Weka tool built on the Java programming language. They selected earthquakes between 4.5 and 5.5 magnitude as their targets for prediction and looked at the region of India for their studies, breaking up the country into geographical quadrants for further comparison. All models produced accuracies in the high 90s, with the simple logistic regression and Logistic Model Tree performing the best. However, when they compared the performance across F-1, recall, and precision scores together, they concluded the Bayesian network performed the best overall across all regions of India of the four models they implemented and tested, while the logistic regression and LMT performed better for specific regions of India.

Table 2

Summary of Models Used in Existing Research

Paper	Model	Validation and Methods
Console et al. (2012)	Bayesian machine learning with Monte Carlo simulations following a Poisson process	Confidence intervals, model comparison with benchmark classification accuracy
Cano & Peter (2021)	Fuzzy c-means clustering (FCM) of phase locations to determine targets for classification	S/N, mean and standard deviation of error (missed clusters) compared with performance of existing models
Debnath et al. (2021)	Bayesian network, random forest, logistic regression, logistic model tree (LMT)	Precision, recall, accuracy, F-1 score, MCC

Paper	Model	Validation and Methods
Dumke & Berndt (2019)	Random Forest of 100 trees using python scikit-learn RandomForestRegressor and recursive feature selection	10-fold cross validation with leave-one-out. Root mean squared error (RMSE) and mean absolute error (MAE)
Kwon & Elnashai (2006)	Regression analysis using ground motion equations	RMSE, MAE, R^2
Mendicelli et al. (2022)	Clustering algorithms, IGAG_20 methodology	MAE
Mizrahi et al. (2021)	Decision tree with variable catalog completeness algorithm	Information gain, count of detected versus undetected events compared with ETAS null model
Pardo et al. (2021)	Convolutional Neural Network focused on guessing phase locations using TensorFlow with cross-entropy loss & Adam optimizer	F1 score
Tezcan & Cheng (2021)	Regression analysis with support vector machines (SVM)	Comparison of generalization error and fitting error (validation vs training error)
Xiong et al. (2022)	Inverse Boosting Pruning Tree (IBPT) with AdaBoost	AUC, ROC
Velasco et al. (2022)	Bayesian non-linear LS-SVM	Visual comparison of locations and times of predicted model with historical results

Literature Survey of Existing Research

There is currently a focus on exploring new models and approaches to solving fundamental challenges to long-term and short-term earthquake forecasting. These challenges include a lack of complete earthquake catalogs for specific regions of high interest, a scarcity of target events for earthquakes of high magnitude, a lack of evenly-available global data especially in areas like oceanic trenches, and complex relationships between geological and seismic states

with differing methods of approximation or empirical calculations. Machine learning and advances in automation have improved catalog completeness, leading to new and adjusted approaches of analysis with the newly available data, but highlighting challenges with increases in noise, quality control, real-time processing, and fundamental imbalances in the data that an increased amount or quality of data cannot overcome alone. For instance, Rundle et al. (2021) note that earthquake forecasting using quantifiable observations of past earthquakes like a fault's rate of motion or amount of slip are only accurate for, at their best, a range of 100 years, and that other methods involving forecasting from various precursors have been tried, with varying success and none with consistent, global applicability to all earthquakes. To resolve these, a number of new approaches, models, and methods have been proposed for performing earthquake forecasting and prediction.

Gitis and Derendyaev (2019) proposed a new algorithm called the minimum area of alarm for machine learning models for automated, online forecasting of earthquakes of large magnitudes. In this method, they calculated for their target area and future time period, the minimum area expected to be affected by an earthquake of the predicted magnitude. This model was structured to make up-to-date predictions over time, and thus relied upon established features, data sources, and a self-made machine learning model optimizing for their given algorithm for calculating the minimum area of alarm. They found for their model that it predicted sufficiently accurately for deployment, and additionally solves the conundrum of handling single-class predictions in a method separate from other existing solutions. Their model additionally could account and adjust for a later decreased probability of rare, sufficiently anomalous events, where other models may perform worse or be less capable at picking up such events when their occurrences in the dataset drops. This provided a level of robustness to the

model as it handles changing data over time, allowing it to respond flexibly to changes in probabilities of the original dataset that may otherwise require a retraining of their model. However, their study estimated true magnitude from a given sample of expert estimates of the given earthquakes' magnitudes. Since the data involved estimates from experts, regions that did not participate were lacking in data, and the dependence on experts limited the quantity of data available, possibly affecting the reliability of their models.

Kubo et al. (2021) tackled the susceptibility of seismometers to noise, a major barrier to obtaining good training sets. This noise included noise from the natural motion of the earth, called ground motion, so mathematical models, called ground motion prediction equations (GMPEs), could be used to estimate nearby ground motion so that seismometers could then filter them out and avoid raising false alarms. However, training sets for modeling ground motion tended to be biased towards weaker motions due to the relative rarity of larger motions, so they proposed a method combining GMPEs with machine learning to overcome this limitation.

Hanks & Kanamori (1979) note that the measurement of larger motions is also bounded by the physical limitations of the machine doing the measuring. The traditional measurement for earthquake magnitude is more sensitive to lower intensity earthquakes and less sensitive to higher intensity earthquakes, which creates a clustering of wildly different high-intensity earthquakes in the upper measurements. Mendicelli et al. (2022) noted that clustering algorithms applied to local geology can be used to estimate ground motions and improve predictions about earthquakes and damages.

Gentili and Di Giovambattista (2022) proposed a new approach to machine learning called NESTORE, which aimed to solve issues with the inherent scarcity of target events in earthquake catalogs by instead estimating the probability of target events. This method focused

on analyzing aftershocks and foreshocks surrounding a target earthquake and using those readings to estimate the likelihood of a strong following earthquake. The reasoning behind this was the relative completeness of such events in earthquake and seismic catalogs, which could pick up local shocks relative to the larger earthquake given that they captured the larger earthquake to begin with. The locality of events both temporally and spatially ensured the completeness of the catalog, allowing them to predict with an 80% hit rate earthquakes of similar or greater magnitude to the target main earthquake happening in a time frame of up to forty days after the initial earthquake. This method focused on identifying earthquake clusters as they happened, and performed best for short-term forecasts on the order of 5-7 days. This result also formed the limitation of their study, where its accuracy dropped as the prediction time frame increased.

Mir et al. (2022) take the approach of using levels of radon gas to forecast earthquakes instead, based on the observation that higher radon concentrations in the soil precede earthquakes and can thus be used as a predictive measure for forecasting earthquakes. They propose predicting for elevated radon gas levels in soil as a proxy for future earthquakes..They concluded that radon gas could be added as a predictive feature for future earthquake prediction models to boost the accuracy of forecasts. However, the need to manually install radiation-measuring instruments in areas of seismic activity that currently lack them limits the ability to implement this finding, as otherwise there would be a lack of available data.

Chen et al. (2021) build on an approach to earthquake forecasting that trains models based on geoelectric signals. They apply machine learning in order to identify frequency bands of geoelectric signals that had the highest earthquake to non-earthquake ratio. Their algorithm improved on their original model (Chen & Chen, 2016) by applying a joint-stations approach and

the removal of a time parameter, and report observing statistical significance in the relationship between geoelectric signals and seismicity, suggesting that further research into applying machine learning to geoelectric signals as a method of forecasting earthquakes may help the field approach practical earthquake forecasting. Limitations of this study include that the algorithm needs further training on more datasets across different geographical regions and that further research needs to be done to replicate the observed relationship between geoelectric signals and seismicity.

Rouet-Leduc et al. (2017) demonstrate a different approach to earthquake forecasting that attempts to use live signals generated by faults, rather than historical data on when earthquakes have occurred, to predict how much time will pass before the earthquake happens. They tested a model in a laboratory setting, using a laboratory fault that emitted a sound signal prior to earthquakes to produce a training dataset for it. They found that their model identified a previously known signal, thought to only be low-amplitude noise, as a way to forecast earthquakes. However, the model's performance could benefit from being tested on real-world data as it is uncertain if it can generalize to practical settings.

Corbi et al. (2019) posit that the existing common approach of earthquake forecasting, which uses slip-deficit to estimate the next earthquake's size, is not a reliable approach based on the results from a model that generates simulated earthquake data through laboratory simulation. They suggest that a different approach utilizing loading history, which geodesists record, to forecast timing and size of earthquakes may yield higher-performing models and is a promising avenue to consider. Limitations to the study include that their claim that utilizing slip-deficit as an indicator for forecasting earthquakes could be further supported by models trained on real-life data and that their proposed novel approach could be tested further.

Mancini et al. (2022) explore the performance effects of adding enhanced seismic catalogs to short-term forecasting models. They used three catalogs from the 2016-2917 Central Italy sequence and applied them to inform both an existing physical model, Coulomb Rate-and-State, and an existing statistical model, Epidemic-Type Aftershock Sequence to forecast in a timeframe of 6 months. They report that while the introduction of enhanced catalogs improved performance, the improvement value was not significant compared to their near real-time counterparts. Limitations include that the data was generated based on just three catalogs in a single geographical region, meaning that the results could be different across other faults. Further research is needed to fully understand why these results occurred or if they are generalizable to other faults.

Mignan et al. (2021) review a set of papers published in *The European Physical Journal* for a special issue focusing on earthquake forecasting. They point out limitations and directions for future research after reviewing the papers individually. They acknowledge that difficulties of earthquake forecasting include overfitting due to lack of sufficient targets and biases inherent in the data models are being trained on, but hold that perhaps in part due to them, the papers published do not provide sufficient evidence to support their claims. They suggest that the observed issues could be mitigated by further incorporating multi-disciplinary expertise and focusing on upholding the usage of robust statistical methods when conducting research. However, this review has a limited scope of studies it reviewed, and did not apply robust meta-analysis procedures in their selection of papers to include or exclude. Their claims could be bolstered by the usage of a more holistic, statistically robust approach to papers in the field.

Data and Project Management Plan

Data Management Plans

Data Collection Approaches

The dataset used for this analysis was retrieved from the California Institute of Technology's (Caltech) Southern California Earthquake Data Center (SCEDC) Earthquake Catalogs of historical earthquake data recorded in California (SCEDC, 2013). Records span from 1932 to the present day, and began with the first installation of seismic stations in California in 1927 and now contains readings from over 160 stations across the state (Hutton, Woessner, & Hau, 2010). Hutton et al. advise that the readings in the catalog grow more accurate and more numerous over time, and currently total over 420,000 individual seismic events over the entire catalog. Seismic events are captured from Wood-Andersen seismometers at SCSN stations within the SCEDC monitoring region, and from surrounding stations outside the monitoring region whose data are processed at SCSN. The catalog is updated daily and is version controlled on GitHub. Our project takes a subset of this data for training, validation, and testing.

The SCEDC data is provided in either the Southern California Earthquake Center Data Catalog (SCEC_DC) format or in the Southern California Seismic Network (SCSN) format. The SCEC_DC data format is an extension of the SCSN format and includes all important SCSN data fields plus additional fields for the measurement of magnitude used, locality of the data to the SCEC data center, and classification of the type of event (earthquake, quarry blast, sonic boom, and so on). Ten different magnitude scales are listed in the catalog, from the standard magnitude reading from Wood-Andersen seismometers (local, or also called Wood-Andersen), to cases where standard magnitudes were unreadable or absent, usually prior to the digitization of the process (coda duration magnitude and coda decay).

Data from the Earthquake Catalog can be downloaded from the cloud, GitHub, or ASCII flat file archives or through streaming via live queries to their API. The Earthquake Catalog is publicly available and non-confidential. Access is only constrained by the internet and research is actively encouraged on this dataset. The dataset is provided as part of the AWS Public Dataset, and the SCEDC has granted rights to reproduce, use, and display the data contained in the SCEDC Earthquake Dataset. As described in the dataset's Citation Policy (SCEDC, 2013), the only requirement to use the data is to provide the listed citation as a reference back to the dataset, which we have included in the references of this paper.

Data obtained through the above methods are delivered in a series of flat files in the chosen format, either SCEC_DC or SCSN. The flat file archives provide files for every year of data available, while the online API allows live queries based on location and time period. The live queries can take a long time to return results, even for periods of time as short as one day. If the request is made for a time period that is too long, it can cause a slow down of the SCEDC servers or be rejected entirely, making data retrieval difficult or impossible through this method. As our project requires a large amount of historical data over several decades in order to perform predictions, we use the historical, flat file archive of earthquake data for the purposes of this project.

Data Management Methods

The project spans at most three months, until the submission deadline in the middle of May. Data quality will be preserved throughout the duration of the project by regularly reviewing the data through the steps of collection, processing, preservation, sharing, and analysis to ensure the quality of the data. Key input features such as date, location, and magnitude of the earthquake event will be checked between steps to ensure consistency and prevent loss of data.

For the duration of the project, data will be backed up using the collaborative environment provided by Google Drive. Google Drive maintains and preserves a change history of stored documents and similarly can be used to version control and act as a cloud backup of intermediary and processed data shared between project members for use in this project. The source for the models used in this project will be written in Python and provided in Jupyter Notebooks for collaboration and sharing of models and results.

After the conclusion of the project, the raw, intermediate, and processed data will be disposed of. This report and its data will not be published through an agency, nor is the project funded through an agency supporting a data catalog such as USGS, and as such, this data does not require an approval for public release. Additionally, the dataset analyzed in this project is from a published and publicly available dataset which has been cited in the references of this paper and fulfills criteria for exclusion from being re-published. Similarly, we have no plans to preserve the data in an open source format or to store it for future reference or use. The estimated size of the generated data will be less than 70 MB in size, as we will take the raw data and aggregate it across each time-space volume for prediction purposes. As the data will be disposed of after this project, there are no plans to maintain metadata for the processed dataset or to define restrictions for use.

Data Storage Methods

Uncompressed, the entire historical archive from 1932 until March 30, 2023 is estimated to total around 70 MB in size. This is a manageable size even for computers using inexpensive disks, and thus does not require any special technology for storing the data. The data is stored locally during the data collection and exploration phases, as well as for data processing and analysis. Utilizing local storage restricts the access of the data to those performing the analysis.

Raw, intermediate and processed data will be shared using the collaborative file-sharing platform Google Drive, and the access controls of the platform will be used to restrict access to only those participating in the project. The file size of 70 MB is several times less than the storage limit of 15 GB imposed by the most restrictive of limits from Google Drive, and therefore the provided storage space will be more than enough to hold all the data used in this project. Additionally, as the software is a cloud storage device used in this project as primarily for collaborative reasons and only secondarily as a data backup solution, should the storage limit be exceeded, it is possible to remove intermediary or raw data as they become unnecessary for analysis and archive them on local storage systems as opposed to storing them in the cloud.

Data Usage Mechanisms

The data were processed for analysis in several steps. As we were using data from an existing archive of data, we first selected a small sample containing one day's worth of data from the online API for exploration purposes. To look at larger files spanning a year of data, data was pulled from the archived data of earthquake events, which provides one flat file for each year of interest as a separate file in the archives. These archived files were used to provide a sample of the entire data set for initial data exploration and understanding.

To explore multiple years of data or different parts of the larger dataset, the individual files would first need to be downloaded for each year over the time span being looked at, and then combined into a single file before further processing is performed on the data, including filtering and other pre-processing steps. As these flat files were provided in a human-readable format, they can be easily appended to each other by a simple script for further analysis or exploration. Alternatively, each file can be imported one by one into a table of a relational database, then exported from the database as a final file for further processing in python using libraries such as pandas. However, this latter process involving a database includes greater

overhead and may not be necessary for this project considering the simplicity of the task and the consistency of the data catalog. The end result produces a single combined file, for which the overhead of importing into a database outweighs the benefit of strongly enforcing consistency, as the data to be combined is assumed to be consistent by way of the strict format requirements of the data catalog from which it came.

After completing this exploration of the data and selecting this dataset for use, the processing of the data will be performed using the Amazon Web Services architecture. The dataset is available through the ASW Public Datasets framework, allowing simple access to the dataset and bypassing the need to perform a manual combination of the entire dataset for processing and analysis.

The combined data must then be checked to ensure that its quality was maintained and that the data is ready for pre-processing. The entire desired data range should be included in the combined data, and important data features such as earthquake magnitude, location, and date should not be lost after the combination process.

These records will then be filtered specifically for seismic events caused by earthquakes, as opposed to human-related activities such as sonic booms and mining blasts. The data will also be further filtered to specifically only include readings from seismometers, by restricting the recorded magnitude scale to local or Wood-Andersen magnitudes. These are the measures of magnitudes recorded by Wood-Andersen seismometers installed at the various seismic stations, allowing the data to be filtered specifically for only raw measurements and to exclude processed magnitudes included in the dataset.

The data will then be labeled by region of interest and split into time-space volumes, where each volume covers a period of time over each area of interest. Geographical coordinates

will be converted to x, y, and z coordinates relative to each time-space volume. As seismic events are time-dependent variables, the dates of each seismic event will be converted into a time series using a backward shift. The data will be aggregated for the largest earthquake magnitude in each time-space volume in one-year, five-year, and ten-year spans, so that predictions can be run against it in later steps. The data will then be normalized for use in our machine learning models. Redundant features identified by a correlation analysis will be dropped from the data to be used as an input.

The three machine learning models chosen for this report use the same set of input features as each other: year, regional label, x-coordinate, y-coordinate, z-coordinate, magnitude of event, depth of event, recent 1-year aggregated recorded magnitude, recent 5-year aggregated recorded magnitude, and recent 10-year aggregated recorded magnitude. The expected output for each model will be the predicted, expected aggregated magnitudes of earthquakes for the next one year, five year, or ten year time span.

Project Development Methodology

Predictive analytics and data mining projects frequently employ the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology. It comprises six iterative steps that direct the project from comprehending the business objectives through implementing the predictive models when used for earthquake prediction. In the first step, the project's requirements and objectives are laid out. This entails being aware of the objectives and requirements of the stakeholders while predicting earthquakes. The specific issue that needs to be solved may be creating a model that can forecast the probability of an earthquake occurring within a given area and timeframe.

Data exploration and gathering are steps in the second phase, called Data Understanding. Seismic data from various sensors, historical earthquake records, geological data, and other pertinent data sources are all examples of relevant data sources in the context of earthquake prediction. Exploratory data analysis helps to uncover patterns, connections, and potential problems with the data. The third phase is Data Preparation, where the collected data is cleaned, transformed, and prepared for analysis. This includes removing duplicates, handling missing values, normalizing data, and selecting or creating meaningful features. In the case of earthquake prediction, features such as seismic activity patterns, geological characteristics, and historical earthquake data is derived or selected.

Predictive models are generated and assessed during the fourth phase, modeling. You can use a variety of machine learning algorithms, including deep learning methods, support vector machines, and decision trees. Using the proper performance criteria, the models are trained, adjusted, and reviewed. To make sure the selected model performs well on unobserved data, model selection and validation procedures are used.

The constructed models are evaluated in the fifth phase, Evaluation, based on how well they forecast earthquakes. They are measured against the first phase's outlined project objectives and requirements. To evaluate whether the models are suitable for deployment, metrics including accuracy, precision, recall, and other pertinent criteria are examined.

Deployment, the sixth and last phase, sees the integration of the chosen model into a real-world operational context. This includes things like stability, scalability, and long-term performance monitoring of the model. The model may need ongoing monitoring and regular updates after deployment in order to adjust to shifting earthquake patterns or fresh data.

Projects for earthquake prediction can methodically move through these phases by using a project development methodology like CRISP-DM technique. This guarantees a complete comprehension of the issue, makes use of relevant data sources, and creates strong predictive models that can help lessen the effects of earthquakes.

Project Organization Plan

The Cross Industry Standard Process for Data Mining (CRISP-DM) methodology will be used for this project. CRISP-DM is a widely recognized approach to data mining projects and consists of six phases: Business & Project Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment.

Business and Project Understanding:

The aim of this phase is to understand the project's objectives and requirements. The objective of this project is to predict earthquakes accurately and timely. To achieve this objective, machine learning techniques will be used. Before starting the project, a literature review of the topic will be conducted. This review will help identify the models used previously for predicting earthquakes and the challenges encountered during the project. Some of the anticipated challenges for this project include the lack of precise data and the complexity of seismic processes. These challenges will be addressed in the subsequent phases of the project.

Data Understanding

The purpose of this phase is to gain an understanding of the data that will be used for the project. The data requirements will be defined, and possible usable data sources will be identified. Relevant data will be checked for, and key features will be identified. It will be determined if the data is labeled or not. For this project, seismic data will be used. The data sources will be identified, and the relevant data will be checked for. The key features required for

this project include seismic waves, fault parameters, and earthquake location. It will also be determined if the data is labeled or not.

Data Preparation

This phase involves preparing the data for analysis. The data source(s) will be finalized, and features will be selected and unnecessary ones removed through data cleaning. The data will be normalized to ensure that all features are on the same scale. Missing features (or labels) will be identified and added. The data will be batched for training to ensure that the model is trained on representative data. The data preparation phase is essential to ensure that the data is ready for the modeling phase.

Modeling

The purpose of this phase is to develop a predictive model using the data prepared in the previous phase. Various machine learning techniques will be explored, and machine learning techniques for earthquake prediction will be finalized. The models to be used are Support Vector Machine (SVM), Random Forest Classifier, and Neural Networks. Test designs will be generated, and model performance will be analyzed using accuracy, among other metrics. The performance of each model will be evaluated, and the best model will be selected for deployment.

Evaluation

The purpose of this phase is to evaluate the performance of the selected model. Accuracy for detecting P-waves and differentiating P-waves from S-waves will be calculated and compared. Calculation times between models will be compared, and false positive and false negative rates will be checked. The model that gives the best prediction results will be finalized.

The model's performance will be evaluated using different metrics to ensure that it meets the project's objectives.

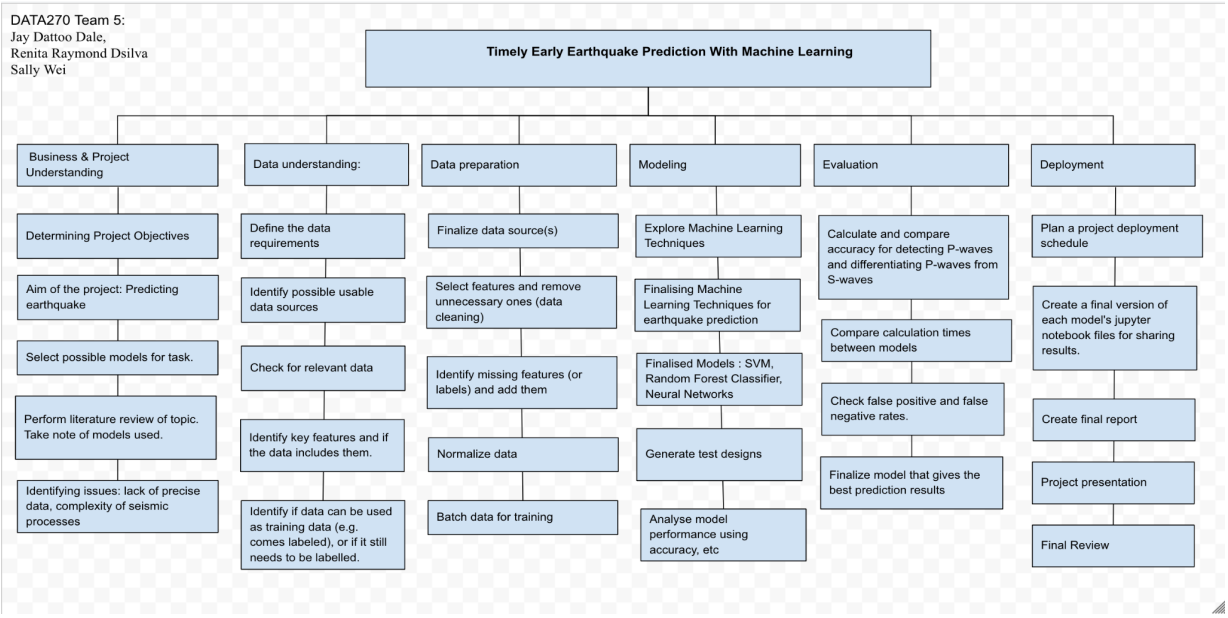
Deployment

The purpose of this phase is to deploy the predictive model. A project deployment schedule will be planned, and final versions of each model's Jupyter notebook files will be created for sharing results. A final report will be prepared, and the project presentation will be conducted. A final review will also be conducted. The predictive model will be used to predict earthquakes in real-time. The results obtained from the model will be made available to the stakeholders. The deployment phase is essential to ensure that the model is integrated into the project and used to achieve the project's objectives.

The work breakdown structure for this project is shown in Figure 2 and divides the project into six phases following the phases of the standard CRISP-DM structure from left to right. The tasks following each phase are listed below them.

Figure 2

Work Breakdown Structure



Project Resource Requirements and Plan

Table 3 presents a list of resources, including software tools, hardware, and licenses required for a data science project. The resources are categorized into different types, such as cloud computing platform, data storage, data preprocessing, machine learning framework and library, storing in repository, visualization, computation and execution, and libraries. The table also provides cost estimates and duration for each resource.

Hardware

The only device used to complete the project was a MacBook. To increase processing speed and effectiveness, the system had 8 cores. The setup also includes a 16 Gb RAM and a 10 core GPU.

Software

The model was run using the Jupyter notebook as the Integrated Development Environment (IDE) on a macOS Ventura operating system running version 13.3. A web-based computing platform called Jupyter notebooks supports Python libraries and makes it easier to develop the project's features while permitting code execution.

Table 3

Tools and Resources for the Project

Tools and applications	Resource type	Cost estimate	Duration	Objective
Amazon Web Services	Software	Free	3 months	Cloud computing platform
Snowflake	Software	Free	3 months	Data Storage
Jupyter Notebook	Software	Free	3 months	Data Preprocessing
Scikit Learn, Kera, Tensorflow	Software	Free	3 months	Machine Learning Framework and library
GitHub	Software	Free	3 months	Storing in repository
Tableau	Software	Free (Student license)	3 months	Visualization
Numpy, Pandas	Software	Free	3 months	Libraries
Local Machine(64 bit)	Hardware	Free (Lent from college)	3 months	Computation and Execution
macOS Ventura	Software	Inbuilt	3 months	Operating system
GPU	Hardware	Inbuilt	3 months	10 core for graphic related work
RAM	Hardware	Inbuilt	3 months	16GB of memory

Project Schedule

Gantt Chart

A Gantt chart is a type of bar chart that illustrates a project schedule. The tasks, their durations, and their interdependencies are all represented visually. Each task in a Gantt chart is

shown as a horizontal bar that spans the task's duration. The benefits of using a Gantt chart include better project planning where it enables project managers to schedule tasks in a way that eliminates scheduling conflicts and delays, better communication purpose that allows team members, stakeholders, and clients to all be informed of the project timetable which makes it easier to make sure that everyone understands what has to be done and is on the same page. It enhances project tracking for project managers to keep an eye on the status of each work, track the project's progress in real-time, and make adjustments as needed. Lastly, Gantt chart provides efficient resource allocation that aids in resource allocation for project managers so they may assign people and equipment to each activity in accordance with the project schedule.

The Gantt chart follows a CRISP-DM methodology which has 6 phases starting with business and project understanding which consists of tasks related to understanding the aim and objective of the project. The next phase is based on data understanding wherein the dataset is fetched from various sources based on the project and evaluated based on the key features that are required to build the project and continue to the next phase of the project. Once that dataset is finalized the next step would be moving on to the data preparation phase where the data is cleaned by removing duplicates or null values and normalized for further analysis. The next phase would be modeling wherein various models based on the project are taken under consideration and the data is trained inorder to get the best results out of the chosen models. Once the models are trained and tested the next phase would be evaluation which checks for false positives and negatives, calculate and compare the accuracy of the models and finalize which is the best model to be taken under consideration and the last phase would be deployment where a final version of the project is created and presented.

In the first sprint which includes business project and understanding phase along with data understanding phase the projects first task would be the main motivation towards the research on predicting the earthquake and finding related technical articles on the project topic and make a note of the paper reviews which would take a span of 2 days and help in notifying the gaps within the research done and learn about the about the models used which would assist in finalizing 3 models for the most accurate prediction results. Overall the business and project understanding phase corresponds to the first half of the first sprint that would take 4 days in total. Moving forward, the second half of the sprint would start off with reviewing the current method for classifying waves along with identifying 1-2 usable data sources. The further tasks would be identifying if data can be used as training data and identifying the target feature and input features in the data which would roughly take about 4 days. Once the tasks are completed the deliverable of finalizing the dataset will be accomplished in the span of next 5 days and would mark the end of the first sprint.

Furthermore, the second sprint would be the most crucial part of the project where analysis of the finalized dataset would be initiated with the data preparation phase wherein the foremost task would be integrating the data sources as needed along with feature selection for each of the three selected models consisting of subtasks which takes approximately 4-5 days. Next task would be performing data cleaning wherein duplicates from the dataset will be eliminated, filling or removal of missing or null data, checking for outliers and reformatting it for further processing. Once the data is cleaned it is normalized and it takes about 2-3 days to reach this step. Towards the end of the sprint training data will be prepared and labeled as P-waves vs S-waves and the data batches will be created.

The third sprint would be the next crucial part of this project which would be working on the three finalized models Neural Networks, Support Vector Machines and Random Forest Classifier based on the finalized dataset. Starting off with the first model i.e Neural Networks where the model is trained, calculating loss - sanity, testing model on a small subset, tuning the model size for number of neurons and layers and at the end checking for loss curves which would take 10 days of work followed by SVM where the model will be further optimized and validated and similarly working on the random forest classifier model with the next 11 days and in total the modeling, training and testing would take 21 days to complete.

Lastly, the fourth sprint comprises evaluation phase and deployment phase wherein the initial task would be checking for false positive and false negative rates, calculating and comparing accuracy for detecting P-waves and then finalizing the best model out of the three which would take 5-7 days of work. The deployment phase is the last part of the sprint that would consist of creating a final version of each model, creating a final report and presenting the project demonstration.

In addition, the final report will include a detailed analysis of the project, highlighting its significance and potential impact. Furthermore, the presentation of the project demonstration will provide an opportunity to showcase the model's capabilities and how it can be applied in real-world scenarios. Finally, the project team will conduct a post-implementation review to identify areas of improvement and provide recommendations for future enhancements. The team will also create documentation outlining the model's usage and maintenance guidelines for future reference.

Figure 3

Gantt Chart for Phase 1: Business and Project Understanding

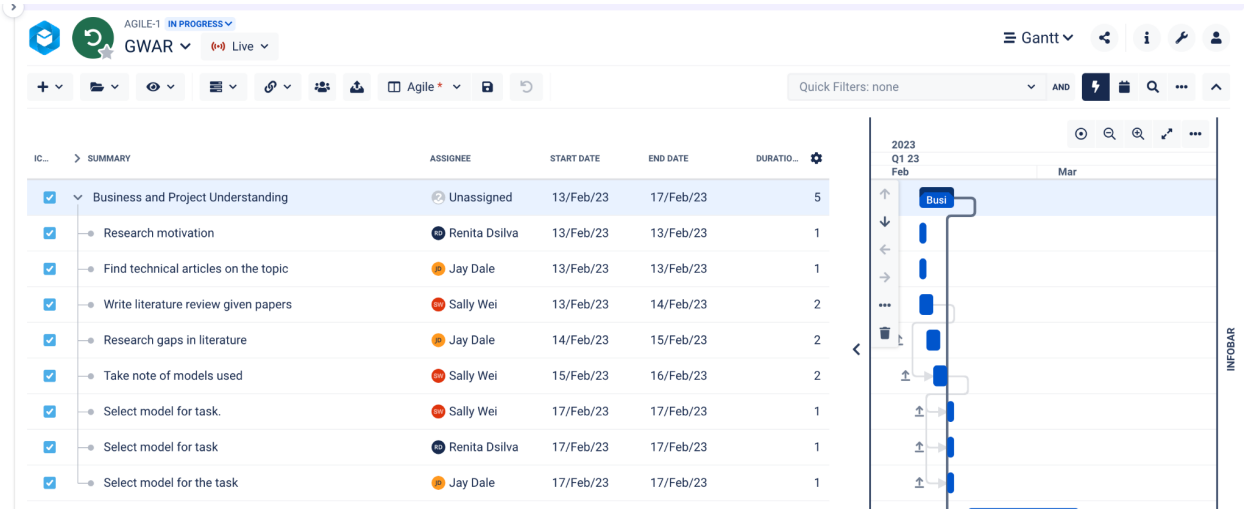


Figure 4

Gantt Chart for Phase 2: Data Understanding

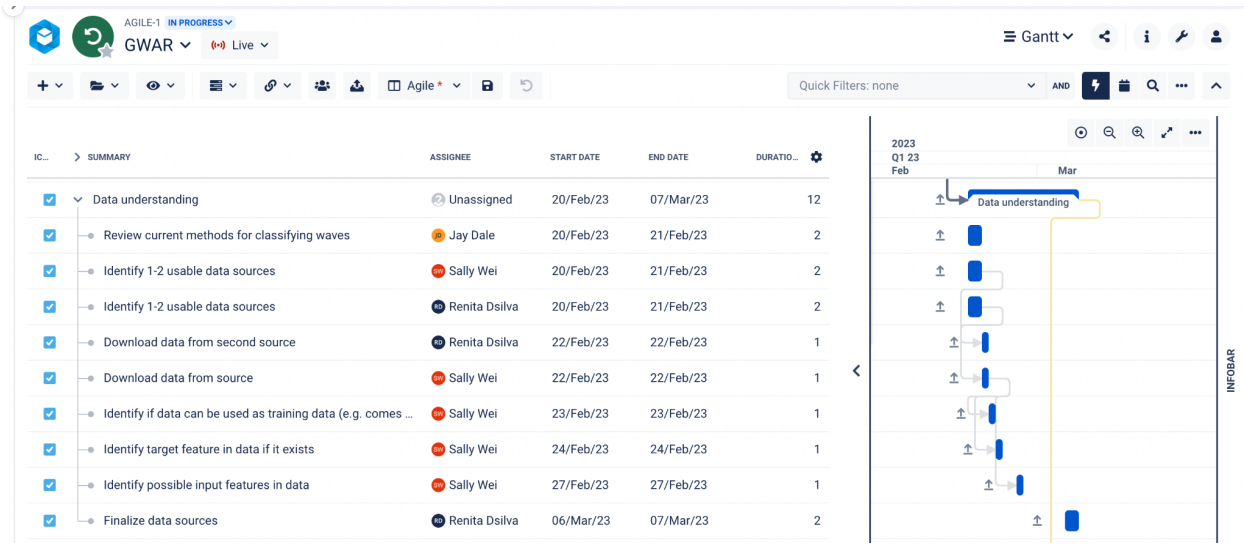


Figure 5

Gantt Chart for Phase 3: Data Preparation

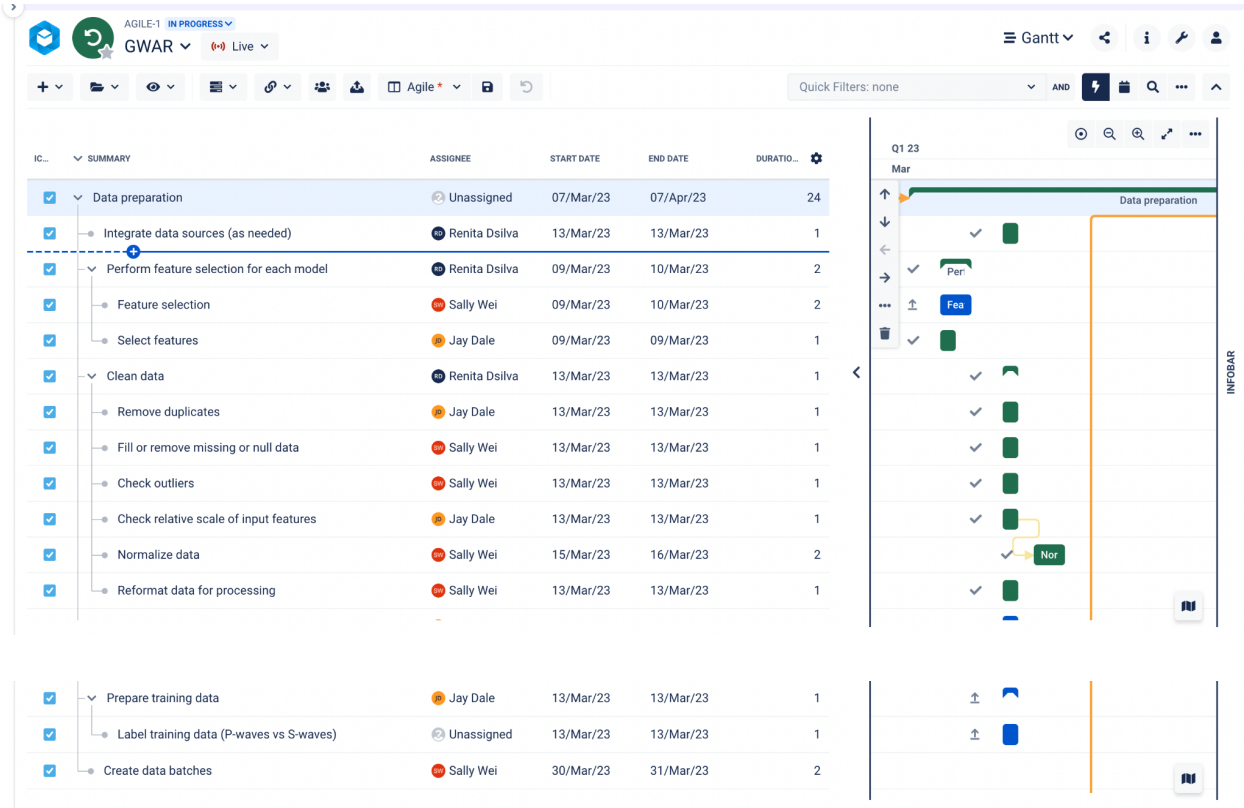


Figure 6

Gantt Chart for Phase 4: Modeling, Model 1: Neural Network

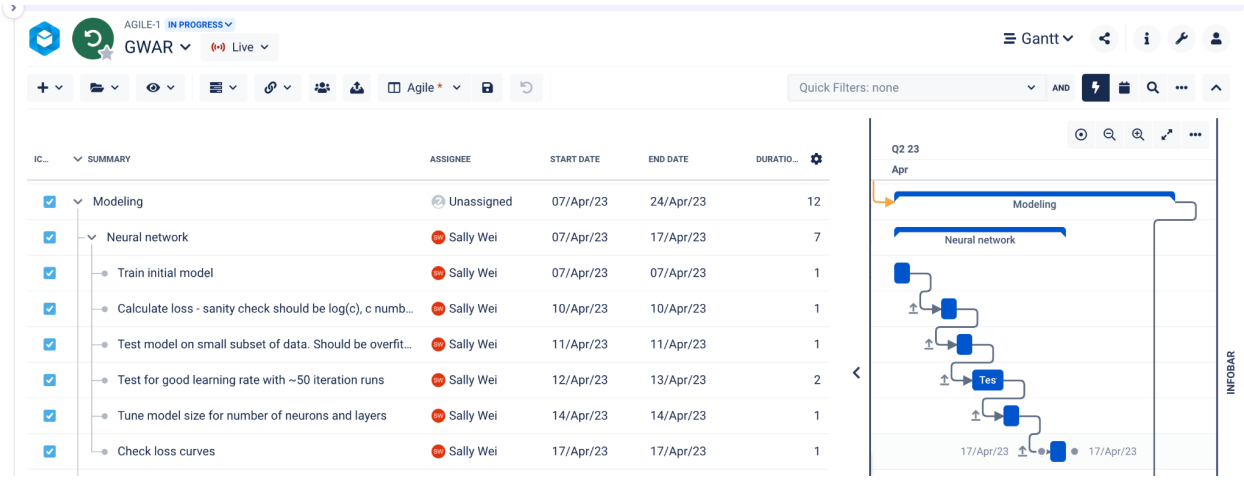


Figure 7

Gantt Chart for Phase 4: Modeling, Model 2: SVM

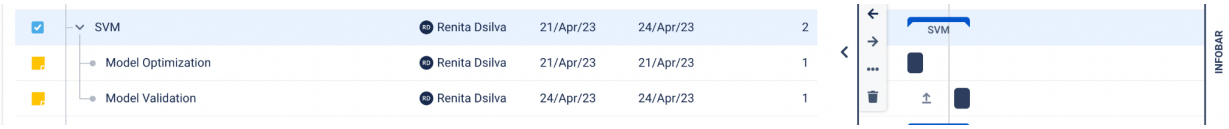


Figure 8

Gantt Chart for Phase 4: Modeling, Model 3: Random Forest Classifier

Figure 9

Gantt Chart for Phase 5: Evaluation

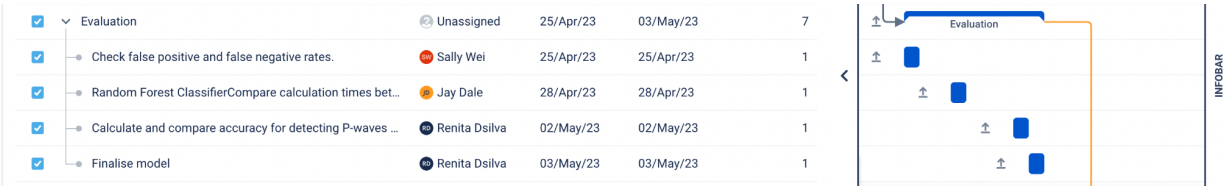
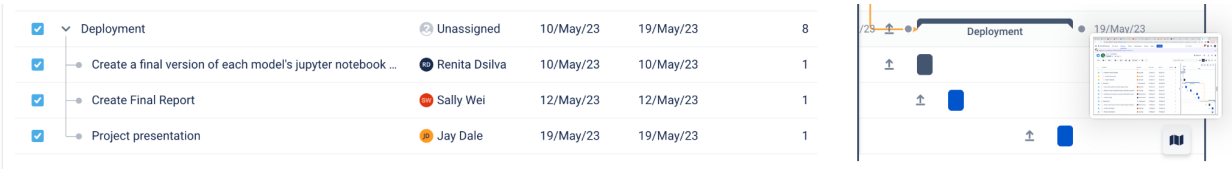


Figure 10

Gantt Chart for Phase 6: Deployment



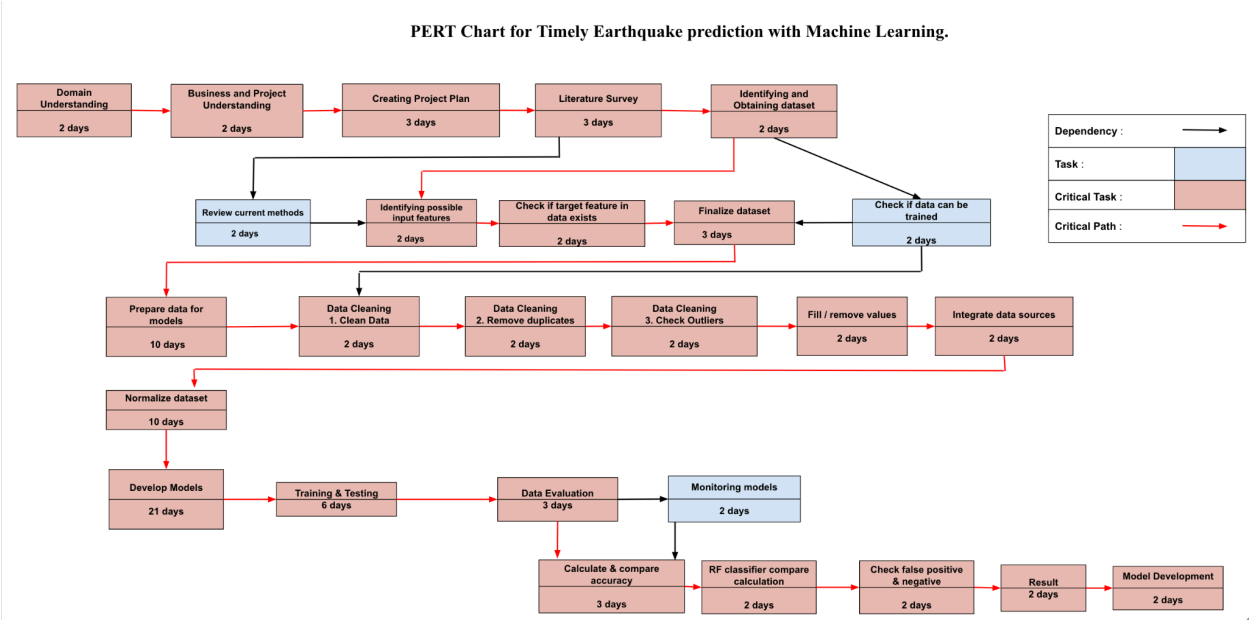
PERT Chart

A PERT chart is a project roadmap tracking dependencies between tasks and estimating the number of days particular tasks will take to complete in order to reach the end goal. It includes a critical path, which outlines the path through the network of dependent tasks that takes the longest amount of time to complete, and for which any delay along the path will delay the completion of the entire project.

The PERT chart for this project is provided in Figure 11 below. In it, the critical path is initiated by understanding the domain followed by business and project understanding based on which a project plan is created. Moreover, a literature survey is conducted which assists in identifying and obtaining the dataset and possible input features for performing analysis inorder to check if the target feature in the data exists that helps in finalizing the dataset. The next step would be preparing the data models which is done by performing data cleaning, removing any duplicates, checking for outliers, filling or removing values and integrating the data source and finally normalizing the dataset. Furthermore, the following machine learning models such as Support Vector Machine (SVM), Neural Networks (NN) and Random first classifier are used for predicting the earthquake and will be trained, tested and evaluated to get the best accuracy out of the three models used. Finally identifying the false positives and negatives and getting the best results to generate a model for development.

Figure 11

PERT Chart for Timely Earthquake Prediction with Machine Learning.



Data Engineering

Data Process

The initial stage of selecting data involves identifying the problem statement and researching to obtain the appropriate dataset. The traditional methods of earthquake forecasting, which rely on calculating statistics or monitoring known active regions, can only predict the likelihood of a single high-magnitude earthquake within the next thirty years. This makes it difficult for organizations and individuals to plan long-term disaster preparation. This project aims to develop a more practical approach by exploring three models: artificial neural network, random forest, and support vector machine models to predict earthquake likelihoods for a shorter time frame of one year, five years, or one decade. After a survey of existing available data sources for use in earthquake prediction training, a dataset of seismic activity across California collected by the Southern California Earthquake Data Center between 1982 to the present year was selected. The dataset includes the depth, magnitude, and coordinates of historic earthquake activity across California.

After identifying the website to obtain the necessary data for the earthquake prediction project, the Southern California Earthquake Data Center website was accessed. The SCSN catalog search 1982 present searchable form link was selected from the Access methods for SCSN section. After selecting the desired start and end dates for the data analysis, relevant attributes such as minimum and maximum magnitude, depth, latitude, longitude, and event type were chosen. The csv file containing the data was then downloaded, with a file size of approximately 68.7 MB. Notably, this dataset was obtained for free, and there were no licensing or access restrictions imposed.

Once the dataset was selected for the earthquake prediction project, the data was prepped for use in training, validation, and testing. This process begins with data cleaning, which involves handling missing values, duplicates, noisy data, and inconsistent data to ensure accuracy and reliability. In this project, missing values were handled by checking if the columns have some null values and replacing them with values based on statistical methods or by removing the columns. Noisy data was eliminated by dropping unwanted columns like *event id*, *geographical type*, *number of picked phases*, and *number of grams*. Inconsistent data was handled in a way where the date column is converted from date to datetime object, applying min-max normalization on certain columns due to varying ranges. Removing superfluous columns, adjusting for missing values, and converting data types in the dataset leads to a more compact and user-friendly dataset, which, in turn enhances the efficiency of data analysis and modeling

The next step in the data processing pipeline is data pre-processing, which involves transforming the raw data into a format that is suitable for analysis. This typically includes several steps such as normalizing the data, feature selection, and data regularization. During data regularization, columns with categorical values such as *event type*, *magnitude*, and *quality* are encoded using techniques like one-hot encoding or label encoding. To prevent overfitting of the model, the dataset is further regularized using techniques like Lasso Regression (L1 Regularization) . The next steps in pre-processing involve data reduction and normalization to ensure consistency in the dataset. These steps are crucial for ensuring that the data is appropriately processed and ready for analysis and modeling.

After completing the pre-processing stage, the dataset is divided into three subsets: training, validation, and test data. The training data is utilized to train the machine learning

model, while the validation data helps to fine-tune the model's hyperparameters. Finally, the test data is used to assess the performance of the trained model.

Data Collection

The dataset is collected from Southern California Earthquake Data Center which updates the report on the seismic events occurring on an everyday basis. The data is combined from Seismographic networks of US Geological Survey, UC Berkeley, Menlo Park & USGS, The Seismological Laboratory, University of Nevada, Reno. The data is collected by a real-time computer that monitors the Earth for any earthquake occurrences and seismic signals are monitored using radio and landlines from 600 remote locations. If any earthquake is detected the data is sent to the web pages using real-time computers and therefore the data sources have data starting from 1980 to present day.

When collected manually and firsthand, seismographs first need to be purchased and installed across locations in the area to be monitored. They must be distributed evenly enough to cover the region in its entirety while being close enough to be able to compare readings of the same event from multiple seismographs. The geographical coordinates of each installed seismograph needs to be tracked for each seismograph so that their readings can be used to identify the location of origin of seismic events that are picked up by the machines.

The seismographs need to be labeled for identification so they can be distinguished from one another. The naming convention can also be designed to include important information like the type of galvanometer used., so that each seismograph’s name also makes this information easily accessible when looking over the raw data being generated.

After the raw data is generated, it needs to be named and labeled accordingly for storage and analysis. The Wood-Anderson seismographs may also not include a method of automatically

digitizing their measurements, and thus the data needs to be digitized and saved as well. This data will be used for analysis and should include consistent naming conventions, storage conventions, and type of data included.

The dataset collection plan is included in Table 4.

Table 4

Data Collection Plan

Goal	Metric	Data Collection Mechanism	Analysis	Sampling Plan	Sampling Questions
Capture record of earthquake activity	Seismic wave readings	Wood-Anderson seismographs	Phase-picking	Set up seismographs in area of interest	Need measures to control for quality and digitization.
Calculate earthquake magnitude and coordinates	Wood-Anderson magnitude	Calculate value from readings	Perform calculations of magnitude	Digitize seismometer readings	Is the digitized data easy to analyze?
Track the location of events	Geographical coordinates	GPS	Used to calculate earthquake sources	Record seismograph coordinates upon installation	What degree of errors is there in the reading? How much will it affect calculations?
Identify seismometers and their data	Name	Individually assigned	Used to match readings with area of origin	Not applicable	Can this name identify each seismometer satisfactorily?

Parameters within the earthquake catalog dataset consist of 13 unique columns: date, time, event type, geographical type, magnitude, magnitude type, latitude, longitude, depth,

quality, event ID, number of picked phases and number of grams. The description of the parameters are as follows.

The date is a datetime data type and follows a YYYY/MM/DD format. Time is in a HH:mm:SS:ss format. The event type column in earthquake data contains values related to earthquakes, quarry blasts, sonic booms, nuclear blasts, and some unknown events. The main focus is on earthquakes as that occurs very often and is periodically present in the dataset.

The geographical type column labels the earthquake as one of three types of earthquakes based on the distance from the observation point to the point of the earthquake's epicenter. The distance from the observation point to the earthquake's epicenter is an important parameter in earthquake prediction because it affects the strength and intensity of the earthquake as well as the time it takes for seismic waves to reach the observation point. Local earthquakes (l) are typically the most destructive because they occur closer to populated areas and their seismic waves can travel through the Earth's crust more efficiently. Teleseismic earthquakes (t), on the other hand, are less destructive but can still be significant because they occur farther away from populated areas and their seismic waves can travel across the entire globe. Regional earthquakes (r) fall somewhere in between local and teleseismic earthquakes in terms of their destructiveness and impact.

The type of magnitude column in earthquake data provides information about the magnitude scale used to measure the size and strength of an earthquake. Each magnitude scale has its advantages and limitations, and the appropriate scale to use depends on the characteristics of the earthquake being measured. The columns have values related to energy magnitude, moment magnitude, body-wave magnitude, surface-wave magnitude, local

(WOOD-ANDERSON) magnitude, revised local magnitude, coda amplitude, helicorder magnitude (short-period Benioff), and coda duration magnitude.

Magnitude is an important parameter in earthquake data because it can provide insights into the potential impact and damage of an earthquake, as well as its likelihood of triggering other seismic activity. It is also used to help understand and predict seismic activity in a region. It is a numerical feature in the dataset.

Latitude refers to the geographic coordinate that represents the north-south position of an earthquake's epicenter on the Earth's surface. The latitude value can be used in combination with the longitude value to pinpoint the exact location of the earthquake on a map. Similarly, the longitude value in earthquake data is used in conjunction with other parameters, such as the latitude, depth, and magnitude, to provide a comprehensive understanding of the earthquake event.

The depth of an earthquake is an important parameter as it provides information on the location and depth of the seismic activity. It is a numerical feature. The quality parameter in earthquake data is used to indicate the reliability of the reported earthquake data and its associated parameters, such as location, magnitude, and depth. The values in this parameter are the following: *A*, *B*, *C*, *D* and *Z*.

Event ID is a unique identifier assigned to each earthquake event recorded in the earthquake prediction dataset. Number of picked phases - The number of picked phases column is a numerical column in earthquake prediction data that records the number of seismic phases picked during an earthquake event. The grams column in earthquake prediction dataset represents numerical data. The values in the grams column are measured in grams, which is a

metric unit of mass. This column provides information on the mass or weight of a specific earthquake event.

See Figures 12 through 17 for samples of the raw catalog file retrieved from the SCEDC dataset are shown below. The file is prepared in a human-readable format using ASCII, saved with the .catalog ending. A text header is included at the top of each file with a description of the file and its format. The zip file had raw data for every year starting from 1980 to present data.

Figure 12

A zip file containing raw data from 1980 to present date in human readable format

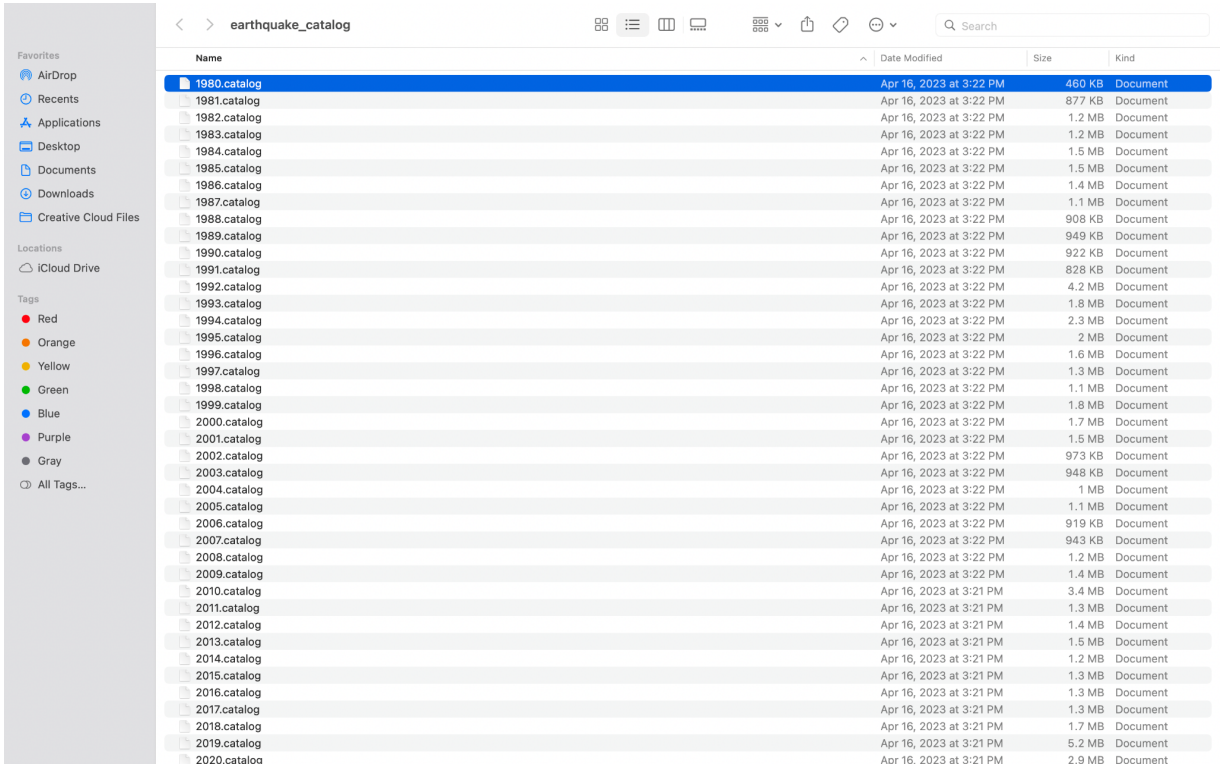


Figure 13

Samples of raw data for year 1980

1980.catalog

```
################################################################################
# The following is an SCEDC-format listing of catalog data from the Southern  #
# California Earthquake Data Center (SCEDC) holdings for 1980. For a format   #
# description, see http://www.data.scec.org/ftp/catalogs/SCEC_DC. The data  #
# below include local, regional, and quarry-blast events with epicenters be-  #
# tween latitudes 32.0S and 37.0N and longitudes between -122.0W and -114.0E. #
# For other event types or locations, see the SCEDC catalog-search page at    #
# http://www.data.scec.org/catalog_search.                                    #
################################################################################
#YYY/MM/DD HH:mm:SS.ss ET GT MAG  M    LAT     LON     DEPTH Q  EVID     NPH NGRM
1980/01/01 00:05:01.21 eq  l 1.80 h    33.723 -118.854   6.0 C 12277543   12  164
1980/01/01 00:05:54.16 eq  l 2.40 h    33.727 -118.811   0.2 A  3301488   27  164
1980/01/01 01:53:06.21 eq  l 1.60 h    33.093 -116.077   6.0 C  3301492   22  164
1980/01/01 02:09:20.62 eq  r 3.10 h    36.522 -121.143   6.0 D  3325141    8   97
1980/01/01 02:29:13.71 eq  l 1.80 h    36.455 -117.934   6.4 D  3301493    7  164
1980/01/01 04:13:56.08 eq  l 2.32 h    33.248 -115.994   6.8 C 12319359    7    0
1980/01/01 04:29:43.26 eq  l 3.12 h    32.795 -115.457  10.0 C  3325143   45   99
1980/01/01 04:34:47.57 eq  l 1.40 h    32.949 -115.525  10.0 C  3301496   11  164
1980/01/01 05:00:13.12 eq  l 2.00 h    32.419 -115.205   6.0 C  3301497   10  164
1980/01/01 06:17:45.25 eq  l 1.50 h    33.492 -116.786   5.5 B  3301498   35  164
1980/01/01 06:36:39.08 eq  l 1.40 h    34.036 -117.259  16.8 A  3325145   32   37
1980/01/01 09:05:23.54 eq  l 1.70 h    34.361 -118.284   4.6 A  3301500   24  164
1980/01/01 09:35:02.54 eq  l 1.80 h    34.410 -118.433  10.1 A  3301501   21  164
1980/01/01 09:39:31.96 eq  l 1.80 h    33.487 -116.804   5.8 C  3301502   28  164
1980/01/01 10:00:20.02 eq  l 1.70 h    34.038 -117.129   7.1 A  3325146   35   56
1980/01/01 10:37:32.49 eq  l 1.60 h    34.014 -116.399   4.1 A  3325151   35   40
1980/01/01 11:58:21.49 eq  l 2.30 h    33.437 -116.406   8.7 A  3301506   52  164
1980/01/01 13:41:59.23 eq  l 1.50 h    32.439 -115.324   6.0 C  3301511   13  164
1980/01/01 14:47:52.85 eq  l 2.00 h    34.916 -119.070  12.3 A  3301514   25  164
1980/01/01 21:14:05.85 eq  l 1.60 h    34.019 -116.724  18.5 A  3325150   32   56
```

Figure 14

Samples of raw data for year 1990

1990.catalog

```
################################################################################
# The following is an SCEDC-format listing of catalog data from the Southern  #
# California Earthquake Data Center (SCEDC) holdings for 1990. For a format   #
# description, see http://www.data.scec.org/ftp/catalogs/SCEC_DC. The data  #
# below include local, regional, and quarry-blast events with epicenters be-  #
# tween latitudes 32.0S and 37.0N and longitudes between -122.0W and -114.0E. #
# For other event types or locations, see the SCEDC catalog-search page at    #
# http://www.data.scec.org/catalog_search.                                    #
################################################################################
#YYY/MM/DD HH:mm:SS.ss ET GT MAG  M    LAT     LON     DEPTH Q  EVID     NPH NGRM
1990/01/01 01:03:44.49 eq  l 2.95 l    34.546 -118.934  16.1 A  1049041   49  192
1990/01/01 01:04:13.76 eq  l 2.77 c    34.543 -118.923  16.3 A   140495   14  192
1990/01/01 01:51:37.89 eq  l 1.73 c    34.545 -118.937  15.4 A  1049043   14   30
1990/01/01 02:59:55.54 eq  l 1.40 n    33.809 -115.932   2.5 A  1049044   10   17
1990/01/01 03:40:45.39 eq  l 1.40 n    35.290 -117.485  19.7 B  1049046    8   10
1990/01/01 04:02:39.51 eq  l 1.50 n    34.383 -116.474   4.8 C  1049047   10   16
1990/01/01 04:09:24.99 eq  l 1.74 c    33.774 -115.991   2.6 A  1049049   16   44
1990/01/01 04:14:14.57 eq  l 2.03 c    33.735 -116.022   8.2 A  1049050   18   52
1990/01/01 04:42:54.11 eq  l 1.10 n    35.534 -118.433   3.4 A  1049051   10   16
1990/01/01 05:17:29.46 eq  l 2.18 c    34.413 -120.093   2.6 A  1049052   18   28
1990/01/01 05:35:30.74 eq  l 0.67 c    33.305 -116.313   8.9 A  1049053    6   15
1990/01/01 06:47:05.93 eq  l 1.68 c    35.365 -117.874   2.5 A  1049055   24   52
1990/01/01 06:55:41.26 eq  l 0.90 n    35.462 -118.808   8.3 A  1049057    8   16
1990/01/01 07:28:39.34 eq  l 1.66 c    36.419 -117.843   4.4 C  1049058   13   17
1990/01/01 07:42:47.12 eq  l 1.50 n    34.550 -118.932  16.1 A  1049059   15   23
1990/01/01 08:07:13.98 eq  l 1.09 c    35.970 -117.592   5.1 A  1049060    9   11
1990/01/01 08:08:02.50 eq  l 2.61 l    35.983 -118.357   4.2 C  1049061   32  119
1990/01/01 08:46:19.25 eq  l 1.67 c    35.286 -118.900  19.2 A  1049063   17   41
1990/01/01 09:05:19.27 eq  l 0.82 c    33.151 -116.499   7.6 A  1049064   10   15
1990/01/01 09:34:48.67 eq  l 1.10 n    33.518 -116.448   8.3 A  1049066   14   24
```

Figure 15

Samples of raw data for year 2010

2010.catalog

```
##################################################################################
# The following is an SCEDC-format listing of catalog data from the Southern  #
# California Earthquake Data Center (SCEDC) holdings for 2010. For a format   #
# description, see http://www.data.scec.org/ftp/catalogs/SCEC_DC. The data  #
# below include local, regional, and quarry-blast events with epicenters be-  #
# tween latitudes 32.0S and 37.0N and longitudes between -122.0W and -114.0E. #
# For other event types or locations, see the SCEDC catalog-search page at    #
# http://www.data.scec.org/catalog_search.                                    #
##################################################################################
#YYY/MM/DD HH:mm:SS.ss ET GT MAG  M    LAT     LON     DEPTH Q  EVID     NPH NGRM
2010/01/01 00:16:54.48 eq  l 1.10 l    36.032 -117.782   1.1 A 14566836   26  402
2010/01/01 00:30:16.72 eq  l 0.75 l    33.475 -116.427   4.4 C 14566844   23  515
2010/01/01 00:34:29.70 eq  l 0.46 l    36.004 -117.789   2.0 A 14566852   12   69
2010/01/01 01:16:54.15 eq  l 0.50 h    33.203 -115.573   0.2 C 14566860   13  727
2010/01/01 02:27:45.96 eq  l 2.18 l    32.471 -115.214   6.0 C 14566868   24  131
2010/01/01 02:33:42.82 eq  l 3.24 l    32.454 -115.203   6.0 C 14566876   38 1770
2010/01/01 02:55:04.37 eq  l 2.79 l    35.985 -117.298   0.7 A 14566884   59 1951
2010/01/01 03:01:18.47 eq  l 1.44 l    34.146 -117.628  11.7 A 14566900   29 1533
2010/01/01 03:25:30.18 eq  l 2.86 l    36.032 -117.783   0.7 A 14566908   57 1950
2010/01/01 03:39:21.99 eq  l 0.16 l    36.029 -117.778   1.8 A 14566916   12   55
2010/01/01 03:43:42.89 eq  l 0.82 l    33.158 -115.648   4.9 A 14566940   17  118
2010/01/01 03:44:34.53 eq  l 1.30 l    36.033 -117.781   1.0 A 14566932   37 1239
2010/01/01 03:46:22.63 eq  l 0.82 l    36.034 -117.775  -1.2 C 14566948   14   52
2010/01/01 03:50:24.10 eq  l 0.75 l    36.031 -117.781   1.4 A 14566956   18   52
2010/01/01 03:51:07.24 eq  l 0.63 l    36.026 -117.778   2.3 A 14566964   14   40
2010/01/01 03:57:40.71 eq  l 0.71 l    36.034 -117.777  -1.2 C 14566972   14   55
2010/01/01 03:59:33.93 eq  l 1.76 l    32.495 -115.214   6.0 C 14566980   16  109
2010/01/01 04:07:36.32 eq  l 0.69 l    36.002 -117.788   1.7 A 14566988   18   87
2010/01/01 04:31:52.21 eq  l 0.75 l    33.393 -116.989  16.8 A 14566996   35 1134
2010/01/01 04:49:24.99 eq  l 0.62 l    33.516 -116.449   7.2 A 14567004   40  642
```

Figure 16

A sample of the combined raw data from year 1980 to present day

earthquake_catalog_1980_2023

	#YYY/MM/DD	HH:mm:SS.ss	ET	GT	MAG	M	LAT	LON	DEPTH	Q	EVID	NPH	NGRM
0	1980/01/01	00:05:01.21	eq	l	1.80	h	33.723	-118.854	6.0	C	12277543.0	12.0	164.0
1	1980/01/01	00:05:54.16	eq	l	2.40	h	33.727	-118.811	0.2	A	3301488.0	27.0	164.0
2	1980/01/01	01:53:06.21	eq	l	1.60	h	33.093	-116.077	6.0	C	3301492.0	22.0	164.0
3	1980/01/01	02:09:20.62	eq	r	3.10	h	36.522	-121.143	6.0	D	3325141.0	8.0	97.0
4	1980/01/01	02:29:13.71	eq	l	1.80	h	36.455	-117.934	6.4	D	3301493.0	7.0	164.0
5	1980/01/01	04:13:56.08	eq	l	2.32	h	33.248	-115.994	6.8	C	12319359.0	7.0	0.0
6	1980/01/01	04:29:43.26	eq	l	3.12	h	32.795	-115.457	10.0	C	3325143.0	45.0	99.0
7	1980/01/01	04:34:47.57	eq	l	1.40	h	32.949	-115.525	10.0	C	3301496.0	11.0	164.0
8	1980/01/01	05:00:13.12	eq	l	2.00	h	32.419	-115.205	6.0	C	3301497.0	10.0	164.0
9	1980/01/01	06:17:45.25	eq	l	1.50	h	33.492	-116.786	5.5	B	3301498.0	35.0	164.0
10	1980/01/01	06:36:39.08	eq	l	1.40	h	34.036	-117.259	16.8	A	3325145.0	32.0	37.0
11	1980/01/01	09:05:23.54	eq	l	1.70	h	34.361	-118.284	4.6	A	3301500.0	24.0	164.0
12	1980/01/01	09:35:02.54	eq	l	1.80	h	34.41	-118.433	10.1	A	3301501.0	21.0	164.0
13	1980/01/01	09:39:31.96	eq	l	1.80	h	33.487	-116.804	5.8	C	3301502.0	28.0	164.0
14	1980/01/01	10:00:20.02	eq	l	1.70	h	34.038	-117.129	7.1	A	3325146.0	35.0	56.0
15	1980/01/01	10:37:32.49	eq	l	1.60	h	34.014	-116.399	4.1	A	3325151.0	35.0	40.0
16	1980/01/01	11:58:21.49	eq	l	2.30	h	33.437	-116.406	8.7	A	3301506.0	52.0	164.0
17	1980/01/01	13:41:59.23	eq	l	1.50	h	32.439	-115.324	6.0	C	3301511.0	13.0	164.0
18	1980/01/01	14:47:52.85	eq	l	2.00	h	34.916	-119.07	12.3	A	3301514.0	25.0	164.0
19	1980/01/01	21:14:05.85	eq	l	1.60	h	34.019	-116.724	18.5	A	3325150.0	32.0	56.0
20	1980/01/02	03:41:47.52	eq	l	1.20	h	34.213	-117.02	4.4	D	10068506.0	26.0	0.0

Data Pre-processing

The data was provided in a series of SCEDC-format ASCII files, one file for each year. The dataset implicitly missed earthquake readings from locations and times where seismometers were not set up to capture the earthquakes as they occurred. The number of readings per year also started with very few recordings when the data first started to be collected in the 1930s due to the lesser number of seismic stations installed in those times. The amount of data then grew over time as more stations were installed, leading to an inconsistent volume of data for each year over time. There were also missing records that could not be digitized from their physical forms during the SCEDC's transition to digital seismic readings in the late twentieth century.

To avoid inconsistencies in the analysis arising from a significantly decreased volume of data in years prior to 1980, as well as inconsistencies from poorly digitized data or data missing that could not be digitized, we opted to only include events from 1980 and onwards. The individual files for each year of interest were then downloaded so they could be combined using the pandas library for python.

Before the files containing each individual year of data could be combined, they needed to be cleaned up into a format recognizable by the pandas to_csv() function. Each file contains an 8-line explanatory header, describing the data format, which was cut off of each file using a python script. This script opened files in the folder, looped through each file, and cut out the first eight lines before saving each file again to the filesystem.

Additionally, extra spaces between columns were removed to have only a single space between columns, so that a single space could be used as a delimiter when reading the file into a pandas DataFrame with the to_csv() method. The metadata describing the format of the SCEDC format confirms that there are no spaces used within a field, therefore it is safe to use the space

character as the sole delimiter between fields. Then the to_csv() function's delimiter could be set to a single space between each entry, allowing the file to be read into a pandas DataFrame. The script then looped over each file to read them into a single DataFrame, and then exported the final DataFrame to file.

See Figure 17 below for an example of a cleaned catalog file with comments removed and single-spaced padding between columns.

Figure 17

Cleaned Catalog File with Comments Removed and Single-spaced Padding Between Columns

```
#YYY/MM/DD HH:mm:SS.ss ET GT MAG M LAT LON DEPTH Q EVID NPH NGRM
1980/01/01 00:05:01.21 eq l 1.80 h 33.723 -118.854 6.0 C 12277543 12 164
1980/01/01 00:05:54.16 eq l 2.40 h 33.727 -118.811 0.2 A 3301488 27 164
1980/01/01 01:53:06.21 eq l 1.60 h 33.093 -116.077 6.0 C 3301492 22 164
1980/01/01 02:09:20.62 eq r 3.10 h 36.522 -121.143 6.0 D 3325141 8 97
1980/01/01 02:29:13.71 eq l 1.80 h 36.455 -117.934 6.4 D 3301493 7 164
1980/01/01 04:13:56.08 eq l 2.32 h 33.248 -115.994 6.8 C 12319359 7 0
1980/01/01 04:29:43.26 eq l 3.12 h 32.795 -115.457 10.0 C 3325143 45 99
1980/01/01 04:34:47.57 eq l 1.40 h 32.949 -115.525 10.0 C 3301496 11 164
1980/01/01 05:00:13.12 eq l 2.00 h 32.419 -115.205 6.0 C 3301497 10 164
1980/01/01 06:17:45.25 eq l 1.50 h 33.492 -116.786 5.5 B 3301498 35 164
1980/01/01 06:36:39.08 eq l 1.40 h 34.036 -117.259 16.8 A 3325145 32 37
1980/01/01 09:05:23.54 eq l 1.70 h 34.361 -118.284 4.6 A 3301500 24 164
1980/01/01 09:35:02.54 eq l 1.80 h 34.410 -118.433 10.1 A 3301501 21 164
1980/01/01 09:39:31.96 eq l 1.80 h 33.487 -116.804 5.8 C 3301502 28 164
1980/01/01 10:00:20.02 eq l 1.70 h 34.038 -117.129 7.1 A 3325146 35 56
1980/01/01 10:37:32.49 eq l 1.60 h 34.014 -116.399 4.1 A 3325151 35 40
1980/01/01 11:58:21.49 eq l 2.30 h 33.437 -116.406 8.7 A 3301506 52 164
```

The final row of each individual catalog file is a line that lists the counts of the total rows of data included in the file. Since the catalog files were combined in a single DataFrame, they can be filtered for and removed directly from the DataFrame. Figure 18 shows the invalid rows, identified by filtering for invalid text entries in the date-time column #YYYY/MM/DD. The invalid rows were then dropped from the dataset.

The separate date and time columns were then combined and converted to a pandas datetime object in order to perform a time-series analysis on the data in the following manner. First, the date column *#YYYY/MM/DD* and the time column *HH:mm:SS:ss* were concatenated

into a new column called Date. This Date column then needed to be converted to a pandas datetime object using a method call.

Figure 18

Artifacts of Data Combination

	Unnamed: 0	#YYY/MM/DD	HH:mm:SS.ss	ET	GT	MAG	M	LAT	LON	DEPTH	Q	EVID	NPH	NGRI
5598	5598	###	Number	of	rows	returned:5598	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
16283	10684	###	Number	of	rows	returned:10684	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
30311	14027	###	Number	of	rows	returned:14027	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
44776	14464	###	Number	of	rows	returned:14464	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
62666	17889	###	Number	of	rows	returned:17889	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

However, the converted Date column included columns with an invalid number of seconds, 60, when the upper limit is 59 (Figure 19). Since the data being analyzed included data spanning forty years, the exact down-to-the-second timing of each reading was unnecessary. It was determined that only the day of the earthquake was required, and not at what time of day it occurred, so time was removed from consideration and the *#YYYY/MM/DD* column was converted to a pandas datetime type for further use.

Figure 19

Rows with Invalid Timestamps

```
Number of columns with an invalid number of seconds: 30
```

	#YYY/MM/DD	HH:mm:SS.ss	ET	GT	MAG	M	LAT	LON	DEPTH	Q	EVID	NPH	NGRM	Date
17096	1982-02-04	20:12:60.00	qb	l	1.30	n	34.330	-116.950	-1.6	C	105447.0	7.0	10.0	1982/02/04 20:12:60.00
20969	1982-05-09	10:21:60.00	eq	l	1.91	c	35.795	-117.747	8.0	C	530414.0	14.0	14.0	1982/05/09 10:21:60.00
21043	1982-05-11	00:33:60.00	eq	l	0.00	n	36.088	-117.720	7.8	D	530520.0	8.0	9.0	1982/05/11 00:33:60.00
37421	1983-07-20	01:39:60.00	eq	r	1.80	n	36.170	-121.530	5.4	D	10844.0	6.0	12.0	1983/07/20 01:39:60.00
38635	1983-08-18	05:47:60.00	eq	l	1.10	n	33.019	-116.346	2.3	B	20080.0	6.0	9.0	1983/08/18 05:47:60.00

Inconsistent data is a common issue that can arise in datasets, where some features may have absent or incomplete values, or a different value range than the other features. This inconsistency can hinder the performance of machine learning algorithms and lead to erroneous conclusions.

Aside from locations with less readings over time, there were also coordinate-blocks that had fewer or no readings over a given time span due to a lower frequency of earthquakes in the region. Such locations were filled with a magnitude of 0, to represent the area's lack of earthquakes (Figure 20).

Figure 20

Dataset After Filling Missing Magnitudes

```
Total missing values in the MAG column: 0
  #YYY/MM/DD  HH:mm:SS.ss  ET GT  MAG  M     LAT      LON  DEPTH  Q  \
0 1980-01-01  00:05:01.21  eq  l  1.8  h  33.723 -118.854    6.0  C
1 1980-01-01  00:05:54.16  eq  l  2.4  h  33.727 -118.811    0.2  A
2 1980-01-01  01:53:06.21  eq  l  1.6  h  33.093 -116.077    6.0  C
3 1980-01-01  02:09:20.62  eq  r  3.1  h  36.522 -121.143    6.0  D
4 1980-01-01  02:29:13.71  eq  l  1.8  h  36.455 -117.934    6.4  D

          EVID   NPH   NGRM                     Date
0  12277543.0  12.0  164.0  1980/01/01 00:05:01.21
1   3301488.0  27.0  164.0  1980/01/01 00:05:54.16
2   3301492.0  22.0  164.0  1980/01/01 01:53:06.21
3   3325141.0   8.0   97.0  1980/01/01 02:09:20.62
4   3301493.0   7.0  164.0  1980/01/01 02:29:13.71
```

Noisy data, such as columns with irrelevant or inconsistent information, can significantly affect the accuracy of the analysis. Therefore, it is crucial to identify and remove such data to ensure that the analysis is based on accurate and relevant information.

The GT column for geographical type was dropped as unnecessary for the analysis. Geographical type tracks the source of the data, as in whether it comes from a seismometer belonging to the SCEDC itself, or was instead pulled from a seismometer from a different group, whose data was only routed through the SCEDC's network for analysis or storage. The SCEDC

does not place restrictions on the usage of the dataset or its subsets and makes no note of restrictions based on the GT column, therefore it was deemed noise and dropped (Figure 21). By eliminating this column, the dataset was simplified, and the focus could be directed towards the more relevant attributes that would contribute to better results.

Figure 21

Dataset with Geographical Type (GT) Removed

	#YYY/MM/DD	HH:mm:SS.ss	ET	MAG	M	LAT	LON	DEPTH	Q	EVID	NPH	NGRM	Date
0	1980-01-01	00:05:01.21	eq	1.8	h	33.723	-118.854	6.0	C	12277543.0	12.0	164.0	1980/01/01 00:05:01.21
1	1980-01-01	00:05:54.16	eq	2.4	h	33.727	-118.811	0.2	A	3301488.0	27.0	164.0	1980/01/01 00:05:54.16
2	1980-01-01	01:53:06.21	eq	1.6	h	33.093	-116.077	6.0	C	3301492.0	22.0	164.0	1980/01/01 01:53:06.21
3	1980-01-01	02:09:20.62	eq	3.1	h	36.522	-121.143	6.0	D	3325141.0	8.0	97.0	1980/01/01 02:09:20.62
4	1980-01-01	02:29:13.71	eq	1.8	h	36.455	-117.934	6.4	D	3301493.0	7.0	164.0	1980/01/01 02:29:13.71

In Figure 22 below, the columns for Event ID (EVID), number of picked phases (NPH), and number of grams (NGRM) have been dropped as they were deemed unnecessary and thus noise. Event ID, as a numerical identifier, had no relation to earthquake magnitude and was thus excluded. The number of picked phases and number of grams related to the expected quality of the extracted magnitude, location, and depth of the raw seismic readings, and while it represented a measure of confidence in the dataset, could not be used as an input to the final prediction task and were also removed.

Figure 22

Dataset with EVID, NPH, NGRM Removed

	#YYY/MM/DD	HH:mm:SS.ss	ET	MAG	M	LAT	LON	DEPTH	Q	Date
0	1980-01-01	00:05:01.21	eq	1.8	h	33.723	-118.854	6.0	C	1980/01/01 00:05:01.21
1	1980-01-01	00:05:54.16	eq	2.4	h	33.727	-118.811	0.2	A	1980/01/01 00:05:54.16
2	1980-01-01	01:53:06.21	eq	1.6	h	33.093	-116.077	6.0	C	1980/01/01 01:53:06.21
3	1980-01-01	02:09:20.62	eq	3.1	h	36.522	-121.143	6.0	D	1980/01/01 02:09:20.62
4	1980-01-01	02:29:13.71	eq	1.8	h	36.455	-117.934	6.4	D	1980/01/01 02:29:13.71

The focus of this project was specifically on what the dataset describes as local or Wood-Anderson magnitudes, denoted with a single lowercase L, which consists of magnitude

readings derived from the SCEDC's own Wood-Anderson seismometers. However, the dataset includes magnitudes with ten other types of magnitude, which we then eliminated to remove the possibility of inconsistencies between the different types of magnitudes. The eliminated magnitudes included energy magnitude, moment magnitude, body-wave magnitude, surface-wave magnitude, revised local magnitude, coda amplitude, helicorder magnitude (short-period Benioff), and coda duration magnitude. Figure 23 shows the count of rows in the dataset before and after rows were filtered on magnitude.

Figure 23

Dataset Statistics Before And After Filtering On Magnitude

```
Before filter:
count     812749
unique        11
top            l
freq      428801
Name: M, dtype: object
After filter

count     428801
unique         1
top            l
freq      428801
Name: M, dtype: object
```

The data also includes seismic events triggered by sources other than earthquakes, including nearby quarry blasts or sonic booms from overhead airplanes, denoted in a column called ET, for event type. As this project is only interested in earthquake readings, any non-earthquake events were excluded before further analysis. Figure 24 shows that before filtering the data, there were two event types in the data, earthquake (eq) and quarry blast (qb). After filtering the data for earthquakes only, only one event type remained.

Figure 24

Dataset Statistics Before And After Filtering On Event Type

```
Before filter:
count     415206
unique         1
top           eq
freq      415206
Name: ET, dtype: object

After filter
count     415206
unique         1
top           eq
freq      415206
Name: ET, dtype: object
```

To prepare the data for analysis, we drop the extra Date column created during the cleaning process, the HH:mm:SS.ss column that won't be used for future analysis, and the ET and M columns that have been filtered on for a single value and are no longer needed (Figure 25).

Figure 25

Dataset After Dropping HH:mm:SS.ss, ET, M, Date Columns

	#YYY/MM/DD	MAG	LAT	LON	DEPTH	Q
21	1980-01-02	2.35	32.445	-115.162	4.8	C
30	1980-01-02	3.15	34.449	-119.680	15.6	D
33	1980-01-02	2.83	33.040	-115.499	5.1	A
38	1980-01-03	2.49	32.967	-115.542	14.5	A
48	1980-01-03	2.52	33.943	-116.304	0.7	A

Data Transformation

Data Regularization

Lasso Regression or L1 regularization was performed on the dataset to identify feature significance and importance for predicting earthquakes and earthquake magnitudes. The target feature was selected to be earthquake magnitude, the MAG column. Magnitude is a continuous feature, so it was first converted to a categorical feature by taking the floor of each magnitude to serve as the target class.

A breakdown of the counts of rows in each class is shown in Figure 26. Notably, the class with a magnitude of 6 only has one sample in it, whereas L1 regularization performed through logistic regression requires each target class to contain at least two samples. Therefore, the one row with an earthquake magnitude of 6 was dropped before the regression was run.

Figure 26

Count of Rows For Classified Magnitudes

MAG	Count of Rows	#YYY/MM/DD	LAT	LON	DEPTH	Q
0	164905	164905	164905	164905	164905	164905
1	187056	187056	187056	187056	187056	187056
2	52225	52225	52225	52225	52225	52225
3	10119	10119	10119	10119	10119	10119
4	845	845	845	845	845	845
5	55	55	55	55	55	55
6	1	1	1	1	1	1

The Q column contains a letter grade representing the quality of the dataset, which were one-hot encoded for regularization. The Q column was expanded through the encoding into one

column each for the five letter grades A, B, C, D, and Z. A sample of the dataset with binary-encoded columns included is provided in Figure 27.

Figure 27

Regularization Dataset With Column Q One-Hot Encoded

	Unnamed: 0	#YYY/MM/DD	MAG	LAT	LON	DEPTH	Q_A	Q_B	Q_C	Q_D	Q_Z
0	21	1980-01-02	2	32.445	-115.162	4.8	0.0	0.0	1.0	0.0	0.0
1	30	1980-01-02	3	34.449	-119.680	15.6	0.0	0.0	0.0	1.0	0.0
2	33	1980-01-02	2	33.040	-115.499	5.1	1.0	0.0	0.0	0.0	0.0
3	38	1980-01-03	2	32.967	-115.542	14.5	1.0	0.0	0.0	0.0	0.0
4	48	1980-01-03	2	33.943	-116.304	0.7	1.0	0.0	0.0	0.0	0.0

After preparing the cleaned dataset for the logistic regression, the model was trained using MAG, LAT, LON, DEPTH, and the quality columns Q_A through Q_Z as input data, with MAG as the target class. The resulting coefficients are provided in Figure 28. Higher qualities of data related positively with higher magnitudes, while latitude, longitude, depth, and Q_Z or data of unknown or extremely poor quality were of lesser or little consequence.

Figure 28

L1 Regularization Coefficients

```
-0.2941458385007907      LAT
-0.016513208018584684    LON
-0.07348082207674606     DEPTH
3.1993663741376857       Q_A
3.6580773757278635       Q_B
3.786477004503291        Q_C
4.324821932018484        Q_D
0.0      Q_Z
```

Data Aggregation

In Figure 29, the data is aggregated by latitude, longitude, and depth, and the mean values of the other fields are calculated for each group. Using this method can find useful information in big datasets. When putting all the data together, we can see patterns and trends that might not be clear when looking at the raw data. Grouping earthquake data by location and depth can help find places where earthquakes are more likely to happen. By figuring out the average size and other features of earthquakes in each group, one can learn more about how often and how strong they are in different parts of the world.

Figure 29

Averages Aggregated By Longitude, Latitude, and Depth

	LAT	LON	DEPTH	MAG	EVID	NPH	NGRM
0	0.0	0.621625	-0.155254	0.796433	3.240154e+06	12.000000	58.000000
1	0.0	0.687500	-0.196641	-1.217921	3.121953e+06	9.500000	50.000000
2	0.0	0.700250	-0.196641	0.629204	9.082726e+06	9.000000	28.000000
3	0.0	0.711875	-0.196641	1.328527	7.143630e+05	17.000000	12.000000
4	0.0	0.720375	-0.155254	2.210282	3.798400e+04	13.000000	76.000000
...	...	...	...	...	...	...	...
775167	1.0	0.750000	-1.521047	-2.107277	5.269840e+05	4.000000	17.000000
775168	1.0	0.750000	-1.334803	-2.107277	3.341153e+06	0.666667	122.666667
775169	1.0	0.750000	-0.610518	1.685790	9.368856e+06	6.000000	68.500000
775170	1.0	0.750000	-0.589824	1.054879	5.166770e+05	7.000000	15.000000
775171	1.0	0.750000	-0.093172	2.552342	6.259564e+06	15.000000	498.000000

Additionally, because it was considered that the number or frequency of lesser-magnitude earthquakes versus those of higher-magnitude earthquakes may be useful as a feature for prediction, we aggregated the count of earthquakes based on their magnitude. If the magnitude was greater than a value of 5.5, then it was marked L for large, and if less than 5.5, marked S for small. Two new columns were created to hold the aggregated counts of these two earthquake types, named Count_L and Count_S, and appended to the DataFrame.

Average magnitudes were then calculated based on rolling 1-year, 5-year, and 10-year windows, and named MAG for the rolling 1-year average magnitude, MAG5yr for the rolling

5-year average magnitude, and MAG10yr for the rolling 10-year average magnitude. The results of these additional aggregations can be seen in Figure 30 below.

Figure 30

Average Rolling WindowMagnitudes and Annual Earthquake Counts By Year

Year	Count_S	Count_L	MAG	MAG5yr	MAG10yr
1980	626.0	0.0	5.34	5.34	5.34
1981	892.0	1.0	5.75	5.75	5.75
1982	822.0	2.0	5.75	5.75	5.75
1983	965.0	4.0	6.07	6.07	6.07
1984	983.0	4.0	6.07	6.07	6.07

Data Normalization

The numerical ranges of latitude, longitude, and depth were fairly different, so Min-max normalization was applied to the latitude (LAT), longitude (LON), and depth (DEPTH) columns of the DataFrame to align them all to the same scale. By applying normalization to the data, issues of inconsistency that could have resulted from differences in the extent and range of the features were eliminated. This allowed the machine learning algorithm to be trained more efficiently and generate more precise predictions by preventing a skew during gradient descent.

This is especially true in the case of seismic data, where regularizing the data in this manner eliminates any errors that could have been introduced by differences in scale among the columns. By putting all of the columns on the same scale, this method not only makes it simpler to compare and analyze the data, but it also makes it more accurate. The normalized data that are contained within the resultant DataFrame can be utilized in subsequent studies or visualizations. Z-score normalization was then applied to make the \magnitude and depth equivalent (Figure 31).

Figure 31

Sample of the Standardized Magnitude and Depth Columns

```
Standardized data:
              MAG       DEPTH
0        0.629205  -0.093170
1        1.541365  -1.293412
2        0.325151  -0.093170
3        2.605553  -0.093170
4        0.629205  -0.010395
...           ...         ...
812787   1.419744  10.688310
812788  -0.784645  -1.086473
```

First, we checked to see if there were any inconsistent numbers in the magnitudes listed in the dataset by listing all unique values and converting them to numeric data type from string (Figure 32). Then invalid numbers in the magnitude were replaced with NaN values. Finally, the columns were normalized to ensure that the values in each column are comparable by rescaling them to a mean of zero and a standard deviation of one, to produce better results in neural networks with multiple layers.

Figure 32

Unique Earthquake Magnitudes Present in the Dataset

```
Unique values in the MAG column: [1.8  2.4  1.6  3.1  2.32 3.12 1.4  2.   1.5  1.7  2.3  1.2  2.35 1.99
 1.68 0.5  2.5  3.15 1.48 2.83 0.8  2.49 1.1  1.9  2.06 2.52 1.78 1.98
 2.21 2.76 2.22 1.89 2.1  2.2  3.   1.3  0.6  1.59 1.82 2.13 1.67 2.14
 1.95 2.12 2.53 2.62 1.79 3.08 2.09 0.9  2.31 0.7  3.18 2.45 2.02 2.89
 2.7  2.15 2.29 2.47 2.11 2.66 2.05 3.99 2.17 3.05 2.07 2.64 3.53 2.43
 1.83 2.03 1.73 2.33 2.6  1.   1.97 1.92 2.73 3.55 2.88 1.87 1.66 2.25
 1.74 2.92 1.63 1.57 2.27 1.72 2.91 1.88 2.86 1.28 2.41 2.59 1.86 3.13
 2.9  2.24 1.45 2.67 2.81 1.85 2.56 2.37 2.04 2.84 2.8  2.28 1.76 2.19
 1.91 1.93 2.68 3.03 1.39 1.55 3.8  2.69 1.65 1.71 2.01 1.52 1.84 3.02
 2.42 2.34 1.62 2.98 2.38 1.27 3.09 3.73 1.81 2.08 2.16 1.96 3.48 3.88
 2.72 1.61 2.23 5.34 3.27 2.26 2.36 3.61 3.26 2.39 3.17 2.57 1.94 3.82
 2.51 1.24 1.42 1.77 1.56 1.75 1.49 2.93 2.18 3.52 3.06 2.78 2.54 3.11
```

Data Reduction

Single value decomposition using a TruncatedSVD model was applied to the magnitude, latitude, longitude, and depth columns to reduce the dimensionality to two variables with the greatest spread of data. The resulting derived features are shown in Figure 33. This method can

increase the computational efficiency of further models by reducing the dimensionality of a dataset containing earthquake data.

Figure 33

Derived Features From SVD

	SVD1	SVD2
0	0.525582	0.313450
1	2.019721	0.135118
2	0.310678	0.092988
3	1.930130	1.661112
4	0.491639	0.291027

The use of these derived fields has the additional benefit of reducing the number of columns needed to aggregate along for the future time-series analysis, reducing the features being used to just the two resulting from SVD and the time axis. The reduced dataset may be helpful in data visualization and the identification of patterns and correlations between variables. The results of SVD can be used for further analysis, such as aggregating or classifying earthquakes according to their magnitude and geographic location.

Data Preparation

Data preparation played a crucial role in ensuring the accuracy and reliability of the subsequent analysis. The initial data importation, type checking, and summary statistics provided a solid foundation for the subsequent data cleansing and conversion processes. The removal of unnecessary columns like the date and time data from the data frame helped simplify the dataset and save computational resources. Conversion functions like pd.to_numeric and astype allowed for uniformity in the data types of the columns. The LabelEncoder function was then used to encode the categorical features, which were identified after checking for absent value

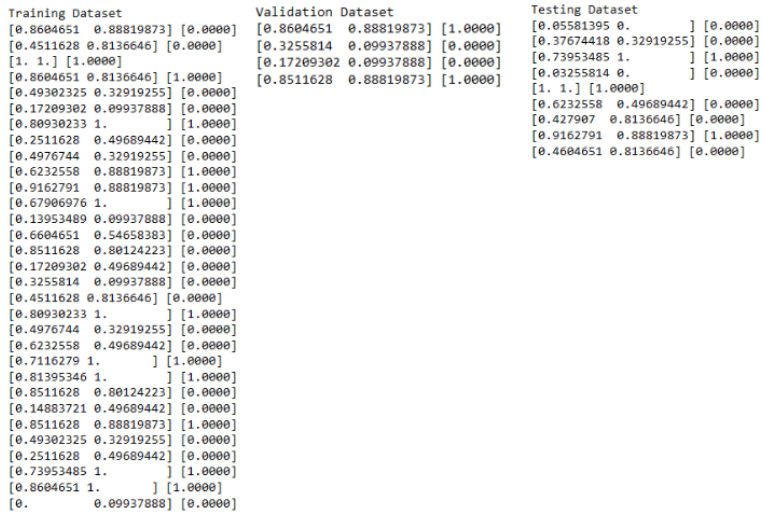
To prepare the data as input, the data needed to be split into training data, testing data, and validation data. The validation dataset was to be compared against the training dataset to

monitor whether the model was underfitting or overfitting, while the testing dataset was used to gauge the model's final performance. The input variables were the rolling one-year maximum magnitude (MAG) and the rolling five-year maximum magnitude (MAG5yr). The target variable was the rolling 10-year expected maximum magnitude (MAG10yr).

The data was split using scikit-learn's train_test_split module in ratios of 10% for test and 90% for training, and the training data was further split into training and validation, with 10% of the data going to the validation dataset. Using this distribution of data, the final training dataset included 31 data points, the validation dataset had 4 data points, and the testing dataset had 9 data points. Samples of the tensor datasets after being split are provided in Figure 34 below. The split data was converted into a torch TensorDataset for usage with a torch DataLoader so the data could be batched during training.

Figure 34

Training, Validation, and Testing Dataset Data Samples



Data Statistics

The earthquake data was collected from Southern California Earthquake Data Center website and after pre-processing steps, the data had 812,749 instances and 8 features. The

features included the magnitude of earthquake, latitude, longitude, depth, earthquake ID, number of phases, and the ground motion readings. During the pre-processing steps, the irrelevant features were dropped, and missing values were removed

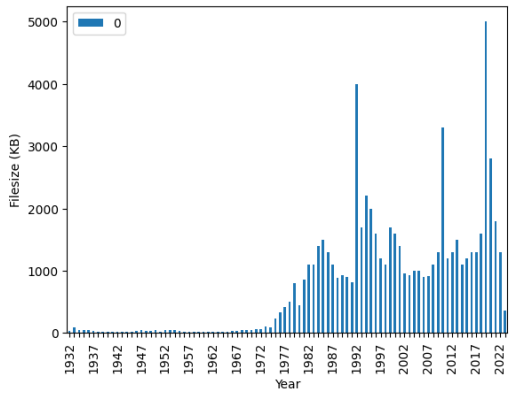
To better analyze the categorical features in the dataset, one encoding was applied, increasing the number of features to 10. In terms of visualizations, various graphs were created to better understand the relationship between different features in the dataset

The pre-processing steps ensured that the dataset was clean and ready for further analysis, while the visualizations helped to better understand the relationships between different features.

The entire available SCEDC data catalog includes data spanning from 1932 to the present day, split into 93 files, one for each year. Each file ranges between 30 KB and 5 MB in size, with the majority. In total, the files are 67.8 MB in size. When compressed into a tar file, the data is 18 MB total. The distribution of the file sizes in kilobytes is shown in Figure 35.

Figure 35

Southern California Earthquake Data Catalog File Sizes in Kilobytes



After selecting our subset of data to only include data from 1980 onwards, the data was reduced to 43 files totalling 63.9 MB before compression and 17 MB total after compression. By selecting only years between 1980 onwards, although 50 years of data between 1932 and 1979

were removed, the data was only reduced by 4 MB. This subset of data from between January 1980 and March 2023 included 812,783 rows and 13 columns after being combined into a single dataset. This combined data formed the raw dataset that all of the further preparation, transformation, modeling, and analysis was based on.

After the data was cleaned, it had 415,206 rows and was 18.6 MB in size uncompressed. After transforming and aggregating the data, it was left with 44 rows and was only 4KB in size uncompressed. A graph of the dataset sizes throughout the analytical process when stored on disk is provided in Figure 36, and a table summarizing the results in Table 5.

Figure 36

File Sizes Throughout the Analytical Process In Kilobytes

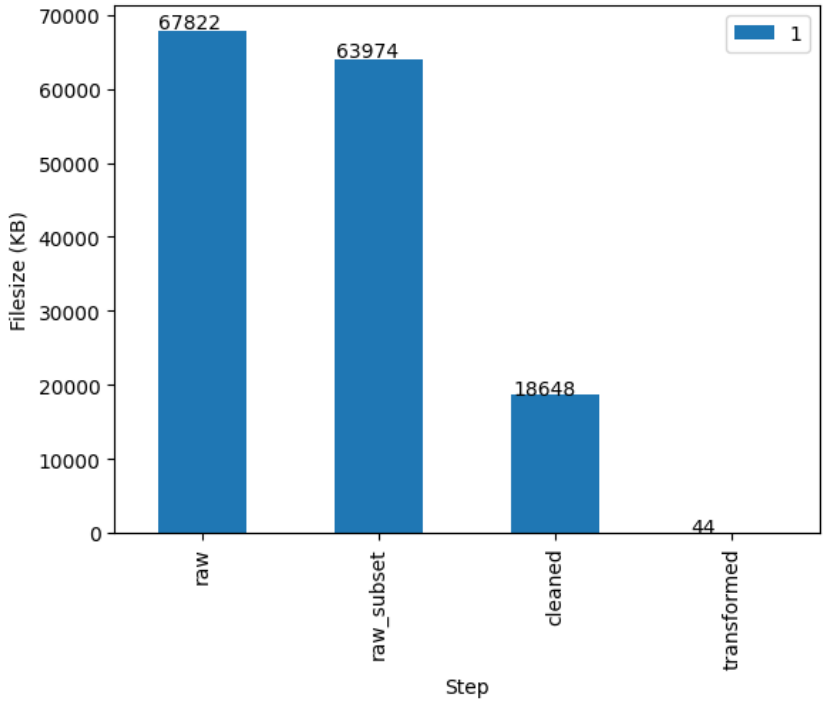


Table 5

Summary of Dataset Statistics Throughout the Analytical Process In Kilobytes

Dataset	Filesize (Uncompressed)	Rows × Columns
SCEDC 1932 - 2023	67.8 MB	-
Raw Data (SCEDC 1/1/1980 - 3/30/2023)	63.9 MB	812,783 × 13
Cleaned Data	18.6 MB	415,206 × 13
Transformed Data	4 KB	44 × 6

Figure 36 shows a heatmap of the correlation matrix, which provides information about the correlation between different variables in the dataset. The correlation coefficient ranges from -1 to 1, with a value of 1 indicating a perfect positive correlation, a value of -1 indicating a perfect negative correlation, and a value of 0 indicating no correlation. From the heatmap, we can observe that there is no strong correlation between any of the descriptive features in the dataset. This indicates that the variables are independent of each other and that there is no multicollinearity present in the data. Therefore, this data can be used to build models without having to worry about the impact of multicollinearity on the model's performance.

Figure 36

Correlation Matrix for EDA of the Prepared Dataset

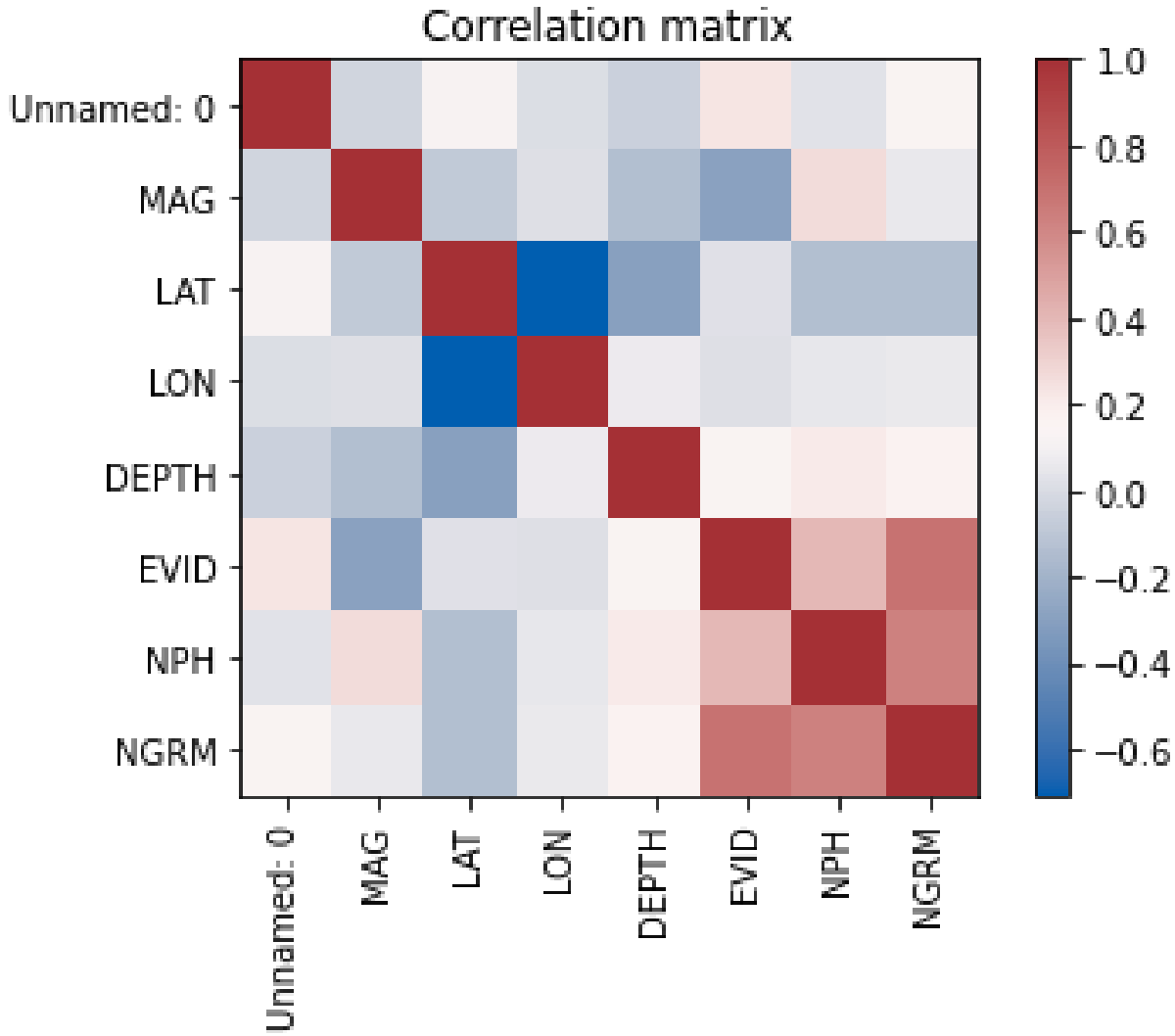
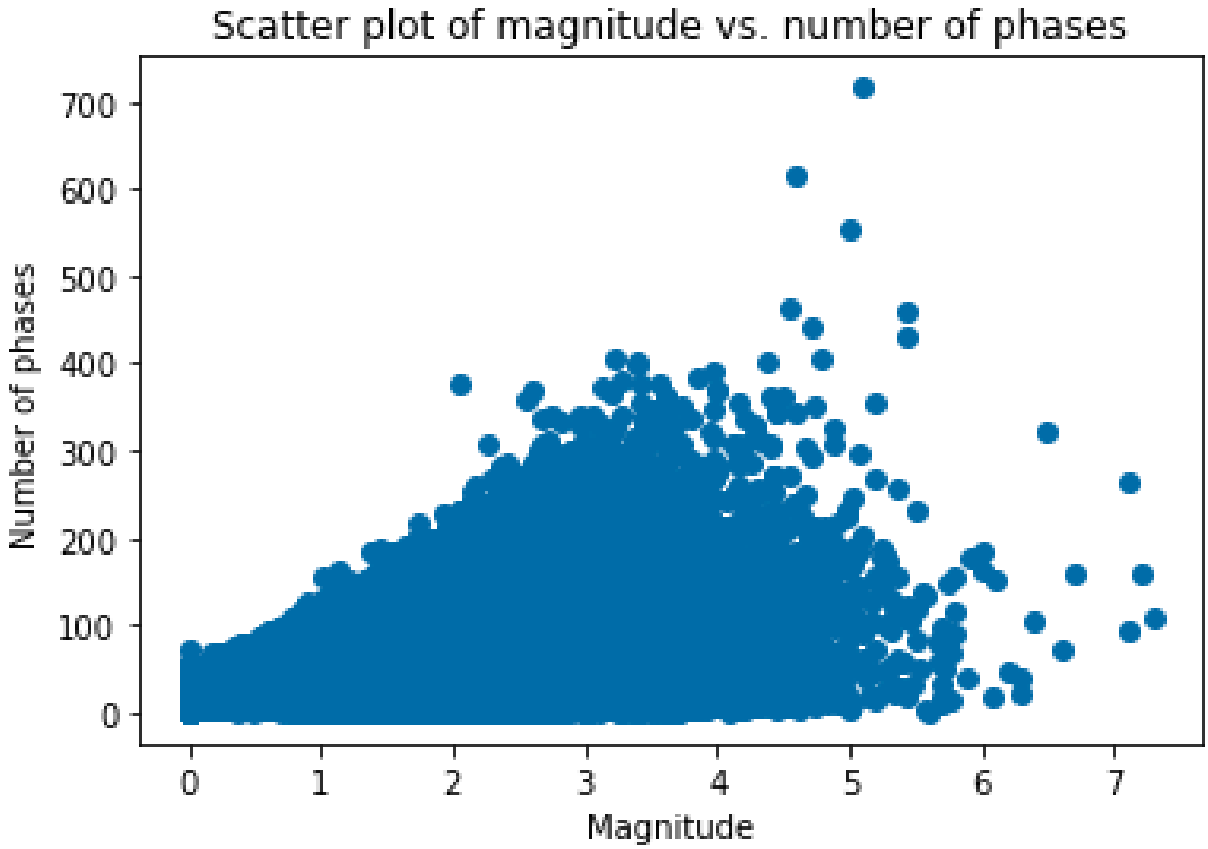


Figure 37 shows a scatter plot of the earthquake's calculated magnitude versus the number of picked phases used to identify its magnitude. It plots the magnitude on the x-axis and the number of phases on the y-axis. The scatter plot shows how the number of phases relates to the magnitude of the earthquake. The plot helps to identify the range of magnitudes and the corresponding number of phases.

It can be seen that there is a positive relationship between the magnitude and the number of phases. As the magnitude increases, the number of phases detected also tends to increase. This could be due to a greater need for precision as the strength of the detected earthquake increases.

Figure 37

Scatterplot for EDA of the Prepared Dataset

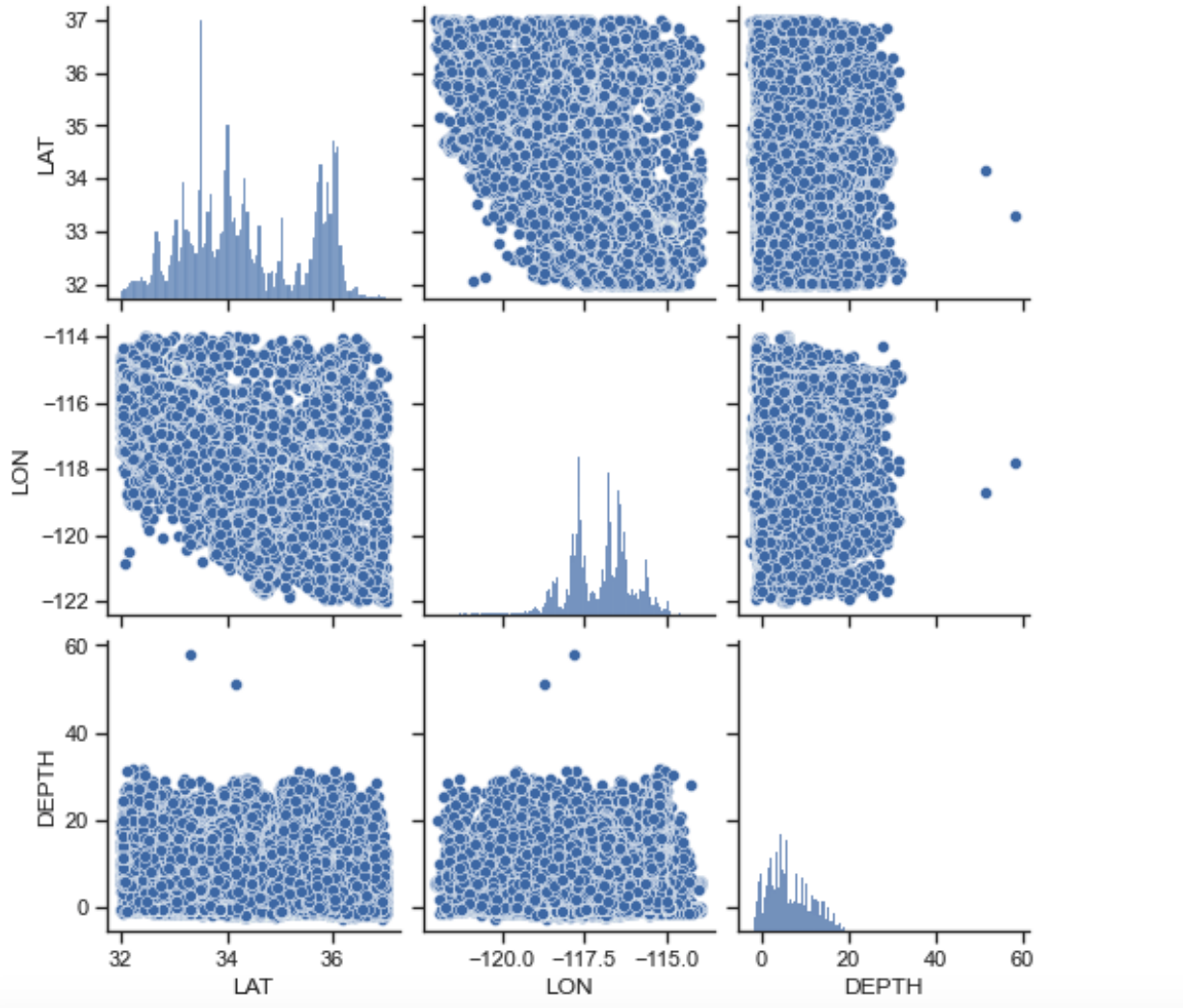


Data Analytics Results

Figure 38 shows a set of scatter plots created using a pair plot to visualize the relationship between various descriptive features. The scatter plot was created using a new dataframe that contained only numeric features. The pair plot revealed no significant correlation between the descriptive features, indicating that there was no multicollinearity in the data.

Figure 38

Pair Plot of Descriptive Features



To further corroborate this finding, a heat map was plotted to display the correlation coefficients between various descriptive features (Figure 39). The heat map showed that the correlation coefficient was quite low among all the descriptive features, which further confirmed the absence of multicollinearity in the data. These findings suggest that the descriptive features used in the analysis are independent and can be used effectively in the machine learning models for timely earthquake prediction.

Figure 39

Correlation Heat Map of Features

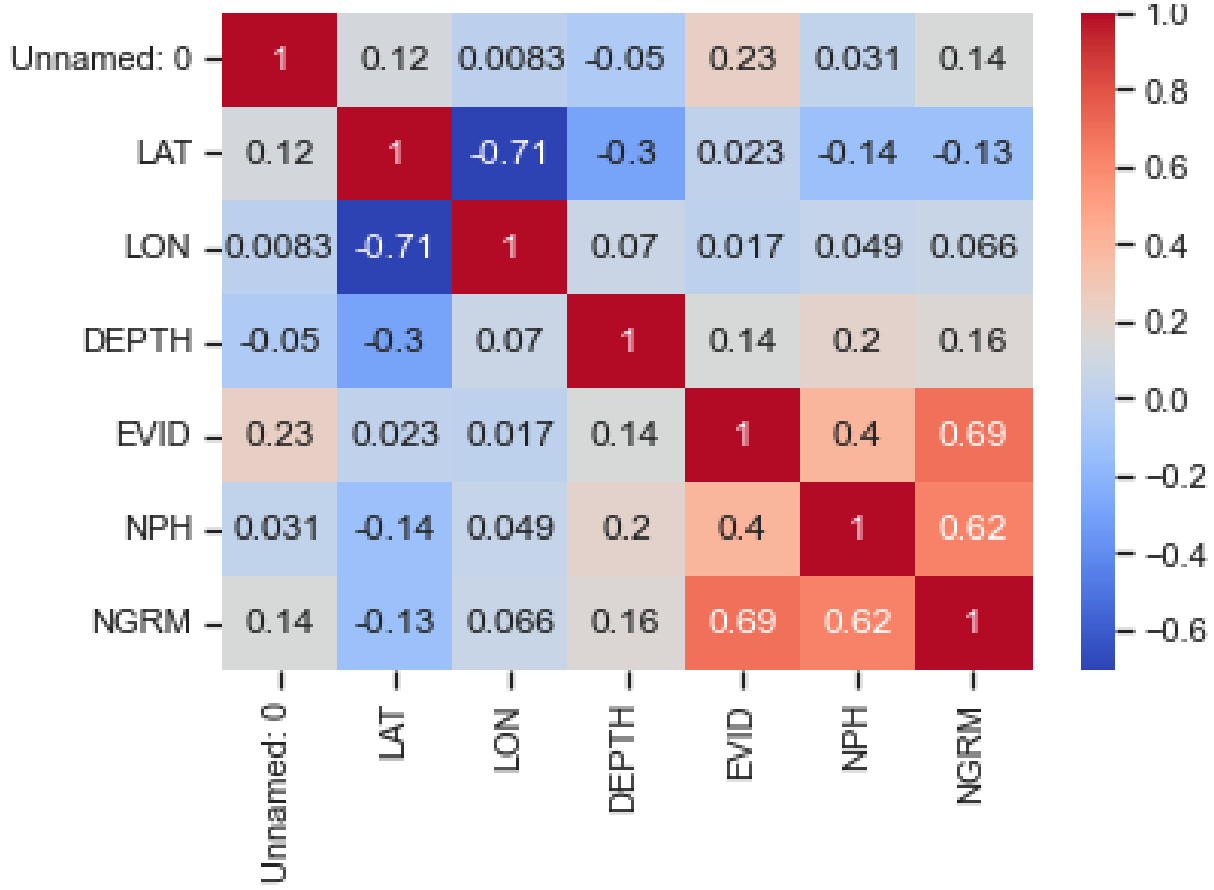


Figure 40 shows a heatmap layer using earthquake data on a map centered on California using the Python package Folium. The heatmap displays the intensity of earthquake occurrences in different areas based on the density of data points. The heatmap shows the areas where the earthquakes were more frequent and the intensity of each occurrence. This visualization technique helps in quickly identifying the hotspots and the intensity of seismic activity in the region of interest. The Folium map can be easily saved as an HTML file for further use.

Figure 40

Geographical Heatmap

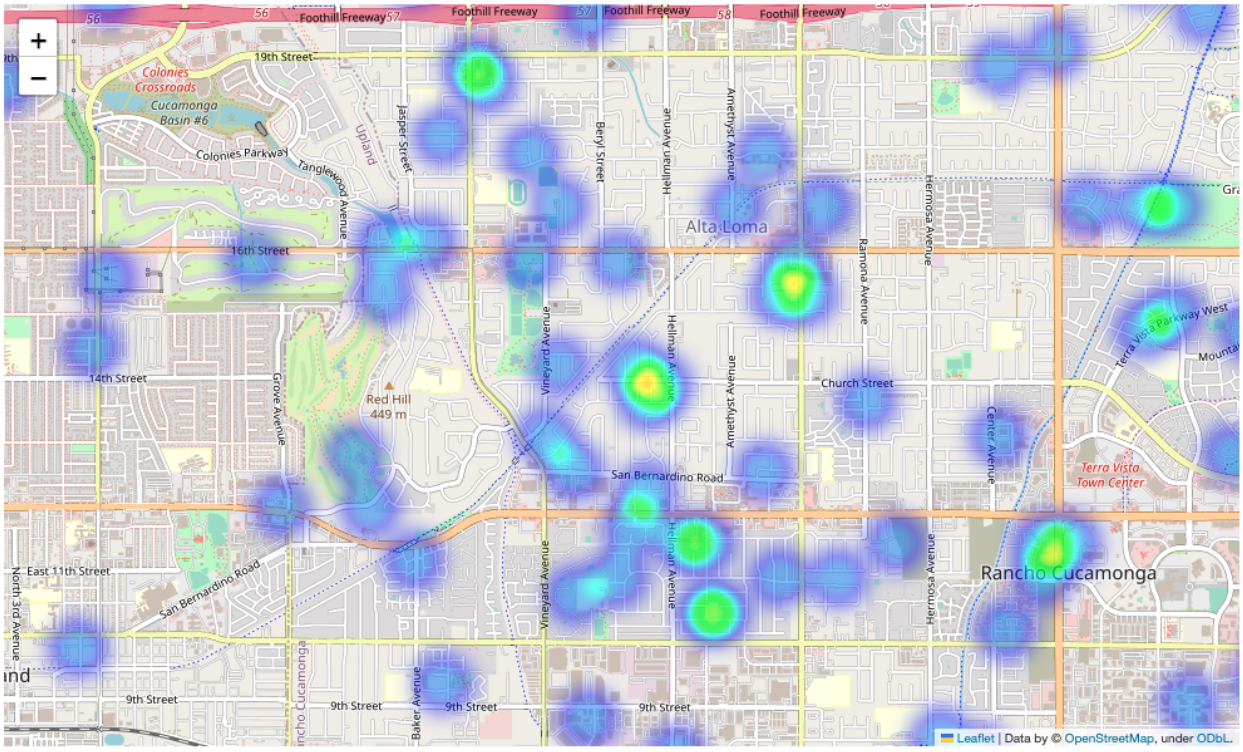


Figure 41 shows a histogram of the distribution of earthquake magnitudes in the dataset. The x-axis represents the magnitude values and the y-axis represents the frequency of earthquakes with that magnitude. The bins parameter determines the number of bars or columns in the histogram. The output of this code is a histogram plot that shows the distribution of earthquake magnitudes in the dataset. From the histogram, we can observe the frequency of earthquakes with different magnitudes. We can see that the majority of earthquakes have a magnitude between 2.5 and 5.5. We can also observe that the number of earthquakes decreases as the magnitude increases, indicating that more powerful earthquakes are less common.

Figure 41

Histogram of Frequency of Earthquake Magnitudes

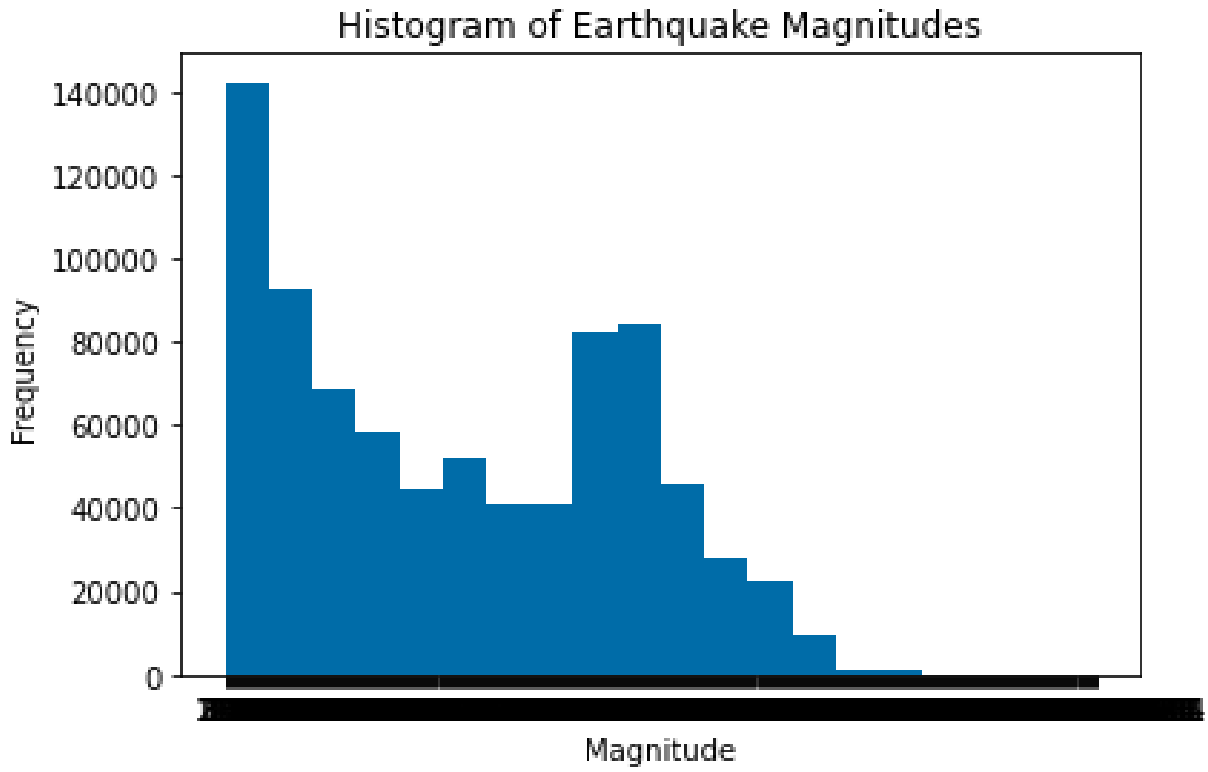
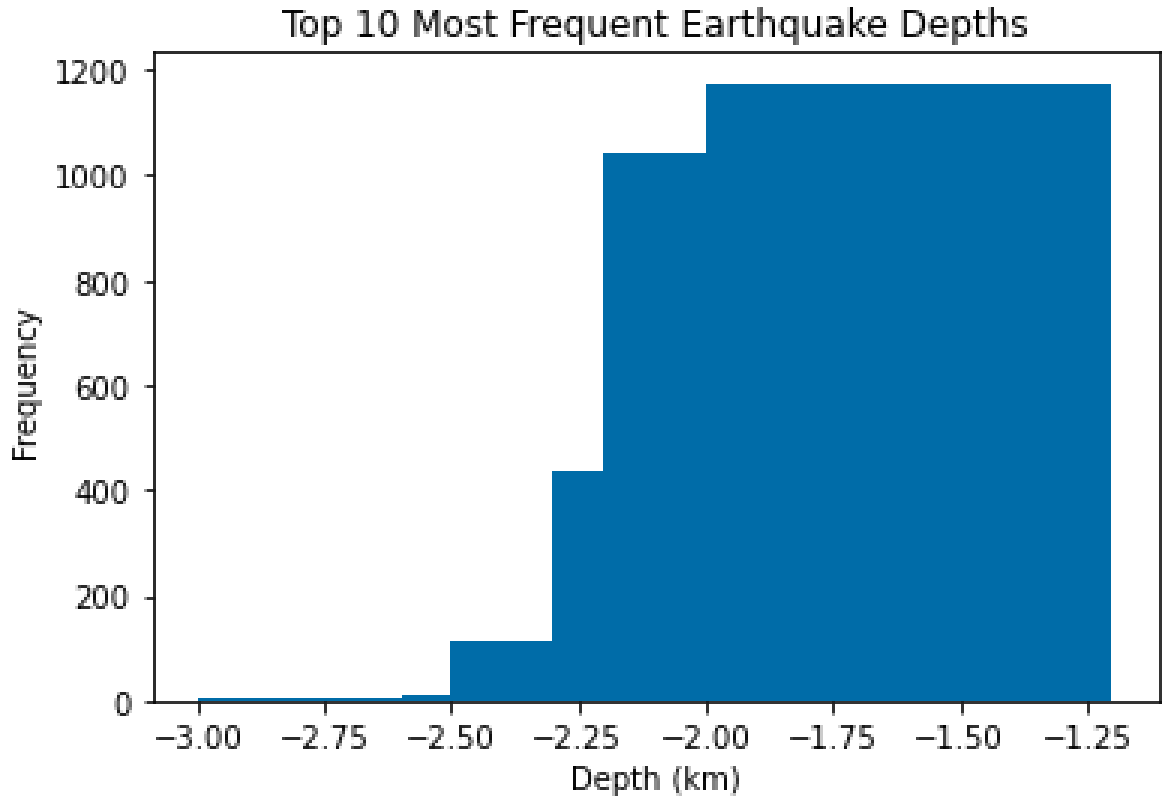


Figure 42 shows a bar chart, from which we get the top 10 most frequent earthquake depths in the dataset. First, the frequency of each depth value is obtained and sorted. Then, a bar chart is created using the top 10 depth values and their corresponding frequencies. The chart is labeled with a title and axis labels. The output is a bar chart that displays the top 10 most frequent earthquake depths in the dataset. We can conclude that the most frequent earthquake depths are concentrated between 0-40 km with the highest frequency at around 10 km.

Figure 42

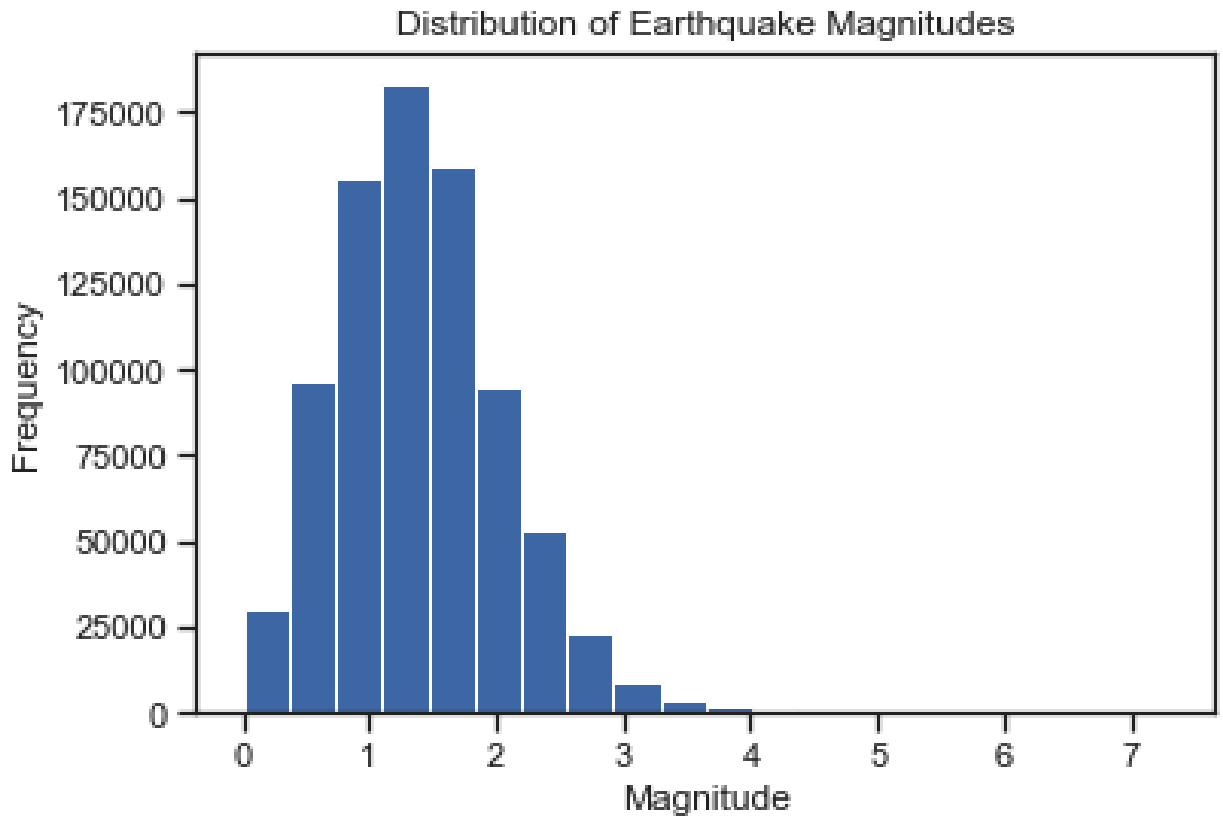
Earthquake Depths



From Figure 43, the histogram, it can be seen that the distribution of earthquake magnitudes. The "plt.hist" function takes two arguments - the feature to be plotted (in this case, "MAG") and the number of bins to be used in the histogram. The "plt.title", "plt.xlabel", and "plt.ylabel" functions are used to set the chart title, x-axis label, and y-axis label respectively. The output is a histogram that shows the frequency of earthquake magnitudes in the dataset. The x-axis represents the magnitude range and the y-axis represents the frequency of earthquakes in each bin. From this plot, we can conclude that the majority of the earthquakes in the dataset have magnitudes between 5.5 and 6.5, with a decreasing frequency as magnitude increases or decreases.

Figure 43

Earthquake Magnitude Distribution



Model Development

Model Proposals

Random Forest

Random forest is one of the popular machine learning algorithms that can work on categorical data and also numerical data. Classification methods include splitting the data into separate categories or even classes, such as if the earthquake will occur or not, while regression is a method where the model's target is to predict some numerical value, such as the magnitude or intensity of an earthquake. Breiman and Adele Cutler introduced this algorithm in 2001. The random forest also does not require any linearity between the independent and the dependent variable which makes it a perfect fit for the complex dataset. This model is trained on input features and their corresponding target values which can be continuous.

The target variable to be predicted is the magnitude, while time and several other important features will be used as predictors. In regression problems, the random forest model works by creating many decision trees and then taking an average of the prediction to make the final prediction. Each tree in the decision forest is trained with data and features drawn at random. This improves the model's accuracy while decreasing the chance of overfitting.

The Random Forest model makes a prediction for a new sample by having each tree in the forest make its own prediction, and then taking the average of these predictions. Aggregation is a procedure that helps models have lower variance and better generalization performance. Using an ensemble method, the Random Forest model can easily deal with high-dimensional data while maintaining its renowned accuracy and reduced overfitting.

The machine learning algorithm Random Forest is based on decision trees. A splitting function is a mathematical function used to divide data into two or more branches in a decision tree depending on a certain attribute and threshold value. The Gini impurity or information gain

is the most widely used function for data splitting since it extensively describes the degree to which each branch is similar to or distinct from the original set. Using the entropy of the data as a metric, the splitting function seeks to identify the feature and threshold value that best distinguishes across classes. Common criteria for splitting in decision trees and Random Forest are Gini impurity and information gain.

The Gini impurity index measures how much noise or irrelevant data is included in a dataset. Its value can be between 0 and 1, where 0 denotes that all data points belong to the same category and 1 denotes that there is an equal number of data points in each category. Equation 5 shows the formula to calculate Gini impureness.

$$\text{Gini impurity} = 1 - \sum(p\_i)\text{^}2 \quad (1)$$

Where p_i is the probability of an element belonging to class i. The sum is taken over all classes.

Information Gain indicates the improvement in knowledge gained from dividing a dataset along certain criteria. It estimates the degree to which uncertainty or randomness in the data is reduced after being partitioned. Better splits are associated with larger values of Information Gain because they produce more reliable predictions. Equation 6 shows the formula to calculate the Information Gain.

$$\text{Information Gain} = H(\text{parent}) - [\sum(p\_i * H(\text{child\_i}))] \quad (2)$$

The entropy of the parent node is denoted by H(parent), the percentage of elements in the i-th child node is denoted by p_i, and the entropy of the i-th child node is denoted by H(child_i).

Bagging (bootstrap aggregating), a method of building numerous models using independent samples of the training data, is the foundation of Random Forest. In classification,

Random Forest uses the median of each tree prediction, while in regression, the mean of individual tree predictions is used.

The CART (Classification and Regression Tree) technique is the basis for the decision tree algorithm utilized in the Random Forest. Using the value of a predefined feature, the CART algorithm repeatedly splits the training data into smaller and smaller subsets in an attempt to maximize the information gained at each split.

Researchers have investigated the feasibility of using the random forest algorithm to anticipate earthquakes and evaluate the risk of landslides caused by them. The random forest has been shown to be useful in a global scale assessment of landslide vulnerability. (He et al. 2021). In addition, Essam et al. (2021) examined the accuracy of several AI methods for predicting earthquakes in Malaysia and concluded that random forest performed best. Using random forest models, Jain et al. (2021) performed an in-depth analysis and prediction of earthquake magnitude. Using machine learning techniques like the random forest, Asim et al. (2016) predicted earthquake magnitudes in the Hindukush region. Finally, in their discussion of the benefits and drawbacks of employing the random forest method for earthquake prediction, Budiman and Ifriza (2021) provide a concluding analysis.

Jain et al. (2021) demonstrates the use of multiple models and emphasizes the ongoing need for improvement in machine learning models . The initial research paper utilized the USGS dataset to construct prediction models using Random Forest Regression, Multi-Layer Perceptron Regression, and Support Vector Regression. The findings of the study highlight the superior performance of the MLPRegressor on the utilized datasets, indicating the need for further investigations to enhance the efficiency of the models. This can be achieved by incorporating additional data mining techniques or by combining algorithms that yield the lowest errors.

Collectively, these works show that the random forest algorithm is effective in determining how likely a landslide will be caused by an earthquake, locating seismic events, estimating earthquake magnitudes, and providing earthquake forecasts. Examining these papers helped shed light on random forest's effective use in earthquake research, which ultimately led to the model's selection.

Several strategies can be taken into consideration in order to improve an existing model for predicting earthquake magnitude. Feature selection and preprocessing can be carried out to identify the most important characteristics and to preprocess the data through methods like normalization and scaling to guarantee that the features are all of similar sizes and have similar weights. Additional informative features can be extracted from the data using feature engineering approaches. Investigating various modifications, interactions, or combinations of already-existing characteristics may be necessary for this.

In conclusion, a random forest was selected as a model for this project because of the many positive examples of its use in earthquake research found in the literature.

Aside from feature selection and other steps, there is also hyperparameter tuning, which involves experimenting with different values for the model's hyperparameters. For random forests, this included values such as the maximum depth of each tree, the number of trees in the forest or the, and the minimum data points needed to split at each node. Optimization can be done on the number of decision trees used in the model. More trees make the model more accurate, but they also make it harder to train and interpret. Thus, the number of trees and model performance must be balanced. Optimization techniques such as grid search or randomized search, include different combinations of hyperparameters to find the solution. Different sets of hyperparameters are used in optimization methods like grid search and randomized search to find

the best answer. Cross-validation can be done where the model's accuracy is checked by applying it to varying subsets of the data in order to prevent overfitting and underfitting.

Support Vector Machine (SVM)

Support Vector Machines (SVM) have emerged as an effective solution for handling such high-dimensional and complex data. SVMs, a set of supervised learning methods utilized for classification, regression, and outlier detection, work effectively when the number of dimensions surpasses the number of samples, hence making it an optimal choice for earthquake prediction.

In the field of earthquake prediction, SVM can be applied as a Support Vector Regression (SVR) model, aiming to identify a function that deviates from actual training instances by no more than a specified threshold. This concept, known as the "insensitive loss function," allows for the balance between accuracy and reliability in earthquake magnitude predictions, a balance that's vital for efficient disaster management.The input features for such models can include seismic wave measurements, ground movements, earthquake duration, depth, type of magnitude, seismic activity quality, geological characteristics, and historical earthquake data. The output would be the predicted earthquake magnitude, with primary features including historical seismic activity and depth of the earthquake, among others.

SVMs operate in a high-dimensional feature space, defined by the features of the data which in this project includes input features like *year, Count_L and Count_S*. The goal of SVM is to find an optimal hyperplane in the feature space that separates the data into different classes. In the model for a regression task, the hyperplane is chosen to minimize the sum of the distances from it to the data points.The SVM algorithm determines the location of the optimal hyperplane by using support vectors. Support vectors are the data points that are closest to the hyperplane and have a direct bearing on the optimum location of the hyperplane. The optimal hyperplane is the one that maximizes the margin, which is the distance between the hyperplane and the nearest support vectors from each class. By maximizing the margin, the SVM aims to increase the

model's robustness and generalization performance. Since data is not linearly separable in the original feature space, the SVM employs a technique called the kernel trick which involves transforming the data into a higher-dimensional space where it becomes linearly separable. In this architecture the Radial Basis Function Kernel is chosen as it transforms the input data into a higher-dimensional space where a linear separation is possible. Since the interactions between features in a problem like earthquake prediction can be highly complex, this enables it to capture complex correlations between the features in the data. It is possible to tailor the kernel function used for this translation to the particular requirements of the data. The regularization parameter C in the SVM sets the balance between maximizing the margin and minimizing the error on the training data. Smaller values of C focus maximizing the margin whereas bigger values of C prioritize minimizing categorization errors. Cross-validation can be used to get the ideal value of C.

Kernel selection in SVR plays a crucial role and heavily depends on the data characteristics. While several kernels, such as linear, polynomial, gaussian radial basis function (RBF), exponential RBF, and sigmoid, can be applied, each has its advantages and disadvantages. For earthquake magnitude prediction, given the complex and non-linear relationships between attributes, the RBF kernel could be a suitable choice due to its ability to handle non-linear and high-dimensional data. This kernel type transforms the input data into a higher-dimensional space, making linear separation possible and capturing the complex relationships between the data features. Furthermore, the RBF kernel is easier to work with due to fewer hyperparameters and its local behavior, considering only close examples for predictions rather than all instances in the training set.

The RBF kernel is popular in machine learning, particularly for tasks involving non-linear regression or classification, such as predicting earthquake magnitudes. Its non-parametric nature allows it to calculate similarity between two data points in

high-dimensional space based on their separation. The versatility, simplicity, and local behavior of the RBF kernel make it a robust tool for tasks like earthquake magnitude prediction, where the data can present complex and localized structures. Nevertheless, like all machine learning models, it's critical to perform proper hyperparameter tuning and cross-validation to ensure the model's performance and suitability for the task at hand.

Kelleher et al. (2020) combine time-scale wavelet decomposition methods with Support Vector Machines (SVM). Their method improves the estimation of earthquake magnitude based on threshold wavelet coefficients by utilizing SVMs' capacity to handle high-dimensional data and classify seismogram data into various magnitude events. This methodology emphasizes the value of data collection and quality. The research emphasizes the need for many seismograph stations since depending on a single station estimate, while speedy, may not be accurate enough. The anticipated magnitude error can be decreased to less than 0.4 by using three or more stations, improving the dependability and accuracy of the suggested approach.

In conclusion, SVM, particularly with the RBF kernel, presents a promising avenue for earthquake magnitude prediction. SVM was selected as a model that excels at identifying anomalous data in datasets, which in the problem of earthquake forecasting refers to predicting rare instances of high magnitude earthquakes. By leveraging this approach, analysts can improve their predictions accuracy and reliability, thereby contributing significantly to mitigating the impact of these natural disasters.

For an SVM, hyperparameters that can be tuned include regularization parameter, kernel parameters, and kernel coefficients, using methods like grid search or randomized search. Additionally, ensemble methods, such as merging numerous SVM models or combining SVM with other algorithms, can be used to capitalize on the advantages of various models and boost overall prediction accuracy. Validating models frequently against fresh data can help pinpoint areas for improvement and direct additional optimization efforts.

Support Vector Machine as Regressor (SVR). The objective of SVR is to find a function that approximates the actual outputs of the training data with some allowance for errors and keeping the model as simple as possible.

The function we want to find is of the form $f(x) = \langle w, \Phi(x) \rangle + b$, where $f(x)$ is the prediction for input x, $\langle w, \Phi(x) \rangle$ is the dot product of the weight vector w and the mapped input $\Phi(x)$ and b is a bias term

Here the algorithm will find w and b such that the function $f(x)$ deviates from the actual output y by no more than ε for all training data, and at the same time, $||w||^2$ (the squared norm of w) is minimized. Mathematically, it is formulated as given in Equation 3, which is subject to $y_i - \langle w, \Phi(x_i) \rangle - b \leq \varepsilon + \xi_i$, $\langle w, \Phi(x_i) \rangle + b - y_i \leq \varepsilon + \xi_i^*$ and $\xi_i, \xi_i^* \geq 0$.

$$1/2 * ||w||^2 + C \Sigma(\xi_i + \xi_i^*) \quad (3)$$

Here, $C > 0$ is a regularization parameter that controls the trade-off between the flatness of $f(x)$ and the amount of errors that are tolerated. ξ_i and ξ_i^* are slack variables that measure the deviation of predictions outside the ε-insensitive tube. The following section will discuss the RBF kernel. (Jakkula, V. 2006)

Radial Basis Function (RBF) Kernel. The Radial Basis Function (RBF) kernel works by computing the distance between the data points in a transformed space. The function used for this is a Gaussian function, which is a bell-shaped curve often used in statistics. The RBF kernel is defined in Equation 4.

$$K(x, y) = \exp(-\gamma||x - y||^2) \quad (4)$$

Here, x and y are two data points, and $||x - y||^2$ is the squared Euclidean distance between them. The γ (gamma) parameter determines how fast the similarity metric decreases as the data points get further apart. The larger the γ, the faster the similarity score drops and the tighter the decision boundary in the case of Support Vector Machines.

By using this kernel function, the RBF kernel effectively measures the 'similarity' between new data points and the data points in the training set. This allows the model to classify or predict the new data point based on its 'similarity' to the known data points.

In essence, the RBF kernel computes the 'similarity' between a new data point and the data points in the training set. This similarity measure is then used by the model to make predictions. The RBF kernel has the effect of creating 'landmarks' at the location of each training point in the higher-dimensional space and measuring the distance of a new data point to these landmarks. (Jakkula, V. 2006)

Artificial Neural Networks (ANN)

Artificial neural networks (ANN) are machine learning models borrowing from the biological neurons in the human brain to create a network of artificial neurons, consisting of an input, a body containing a transformative function, the activation function, and an output. Artificial neurons are structured in layers, with one layer consisting of multiple neurons laid out with each receiving one variable as input, then passing the result of its activation function as an output to the neurons of the next layer. The final layer of neurons produces the output, with the output size equivalent to the number of neurons present in the final layer.

Neural networks learn through a process of trial and error, where it produces a single output in a forward pass. The output, a, of any given neuron in a layer is given in Equation 5. It is the result of the sum of each vector input, X, multiplied by the transposed vector of weights, W, passed through an activation function, g_z, which transforms the data to provide nonlinearity.

$$a = g_z(\sum W^T X) \tag{5}$$

After performing the forward pass, the algorithm calculates the loss between the predicted output of the model and the actual expected output, and sums the loss across the model

to produce a cost, and passes that cost back through the model to update the internal weights. The equation to update weights is given in Equation 6 as described by Ahuja & Pasari (2022). The updated weight is the sum of the previous weight less the result of the learning rate, l, multiplied by the slope of the loss function, given as the derivative of the cost with respect to its weight.

$$\omega_{i+1} = \omega_i - l_r \times \Delta\omega \tag{6}$$

Ahuja and Pasari (2022) note that the strength of artificial neural networks for the task of forecasting earthquake magnitudes is its ability to capture the non-linear relationship between the given predictors of the earthquake and its resulting magnitude. Velasco Herrera et al. (2022) selected a least-squares support vector machine to model the nonlinearity of the problem, but noted an inherent uncertainty in their model, where there existed a tradeoff between spatial and temporal accuracy of their forecast. Gitis and Derendyaev (2019) propose a new regression machine learning model called the minimum area of alarm, and they select specifically a non-linear model for their study, suggesting again that nonlinearity is an integral part of earthquake forecasting.

One possible drawback of using neural networks to forecast earthquakes is handling possible periodicity in the data. Although some studies do explicitly consider periodicity in their models (Field et al, 2015), periodicity of earthquakes is not considered in all studies attempting earthquake forecasting, whether involving machine learning (Ahuja & Pasari, 2022) or not (Ellsworth, 1995; Kagan & Jackson, 2010). A periodicity analysis was performed on the data used in this project and concluded that there was no periodicity.

Bao et al. (2021) emphasize the correlation between earthquake magnitude and intensity, and how it affects human society. The authors introduce a novel approach to classify earthquake

magnitudes using a Convolutional Neural Network (CNN) model. This model incorporates a 3D feature-map that combines shallow features and high-dimensional information. To tackle the issue of imbalanced seismic data, the authors employ noise simulation technology and Synthetic Minority Over-sampling Technique (SMOTE). The techniques employed in this study deviate significantly from the conventional methods used in machine learning models. Remarkably, the model achieves an impressive accuracy of 97.88%, establishing itself as a powerful tool for enhancing magnitude classification performance.

Mousavi & Beroza (2020) introduces MagNet, an earthquake magnitude estimation model that combines convolutional and recurrent neural networks. This study was considered due to the strong performance of hybrid models and their unique advantages. The MagNet model directly learns distance-dependent and site-dependent functions from the raw waveforms in the training data. Notably, it achieves accurate predictions without requiring instrument response correction. The paper highlights the potential applications of this method, ranging from routine earthquake monitoring to early warning systems. However, it acknowledges the necessity for future enhancements to effectively handle larger magnitudes.

Neural networks were chosen as a model with great flexibility in handling different types of data and especially for its ability to handle nonlinearity in data, which is an important trait of historical earthquake records and derived measurements commonly used in forecasting earthquakes.

Neural networks can be optimized for a number of hyperparameters such as the number of neurons in each layer, the number of hidden layers, the learning optimizer, and the activation functions for each layer present. The number of neurons in each layer and the number of hidden layers affects the overall complexity of the model. A more complex model is more prone to

overfitting and will have a longer training time and prediction time, but will be more capable of handling large dimensions of data and making complex calculations and predictions with higher accuracy. Simpler models with less hidden layers or neurons are more prone to underfitting, or not learning patterns in the data sufficiently, but are faster to train and make predictions, while taking less space and memory. For building models, it is preferred to start with simpler models and gradually tune them to be more complex as needed until the model is no longer underfitting, so that the final model will be as simple as it can be while still hitting the sweet spot where it learning the patterns in the data sufficiently, yet is still able to generalize to unknown data, where it is neither underfitting nor overfitting.

Selecting an appropriate activation function can make the training process more efficient, both by speeding up the training time of the model, while also improving its minimization of loss and possibly reducing the computational power required for training. This is especially important for complex models involving many layers, or a large number of dimensions, or for problems like earthquake forecasting that are highly time-sensitive, subject to frequent retraining, and require quick training times.

Different activation functions face different challenges that can affect the neural network differently, including vanishing gradients, non-zero-centered outputs, and expensive computation from involving exponents. Activation functions suffering from vanishing gradients halt the learning of the model and are more prone to happen with deeper, multi-layered networks. Non-zero-centered outputs create an inefficiency in the entire learning process when present in even a single layer, and can severely extend the amount of training time to reach an acceptably low level of loss. In the worst case, non-zero-centered outputs can even prevent the loss from converging, preventing the model from learning sufficiently enough at all. Finally, computations

involving exponents produce a significant increase in training time, which can prevent the model from being usable on weaker machines, and especially pose a problem for extremely large datasets and complex models.

Model Supports

Environment, Platform, Tools

The random forest was executed on an Apple M2 processor, an 8-core CPU, a 10-core GPU, 16GB of integrated memory, and 1TB of storage. For the Support Vector Machine as a Regressor, the model was implemented on a MacBook with a 8-core CPU, 10-core GPU and 16 GBs of RAM running the macOS Ventura operating system version 13.3. The neural networks were trained on the same machine running a Windows 11 operating system using a 12th generation Intel Core i5-12400F processor, 12 GB RAM, and an NVIDIA GeForce GTX 1060 6GB GPU. The neural network models were trained using CPU and not GPU.

Python was selected as the programming language for its broad support of machine learning and data analytics via packages such as Pandas and Numpy. Jupyter is a web-based computing platform that supports Python libraries and facilitates the implementation of the project's functionality while enabling the execution of code. Jupyter notebooks were used as the development platform for fast prototyping, immediate feedback, built-in checkpoint features, and its support for collaboration.

Jupyter notebook and Visual Studio Code IDEs were used throughout development for all three models. Several Python libraries were used to develop and evaluate the models, including Numpy for handling numerical computing tasks and arrays; Pandas for large-scale data manipulation and analysis of both structured and unstructured data. Scikit-learn was the primary library for machine learning and for metrics of model performance, like mean squared error which is used in regression problems, and for classification problems , metrics like the F1-score,

recall, precision, and accuracy. The train_test_split function from Scikit-learn was also used to split the data into training and testing sets. The python library PyTorch was used to implement the neural networks, and was selected for its pythonic structure, lending it familiarity and an ease of transferring knowledge from python to pytorch. Table 6 includes a summary of the various software tools used.

Table 6

Summary of Software Tools Used

Library	**Module**	**Method**	**Usage**
Matplotlib	pyplot	plt	Visualizations
Numpy	Random	float, integer	To perform arithmetic operations on the dataset.
Seaborn	seaborn	sns	Visualizations
Pandas	DataFrame	describe, shape, drop, isna, astype	To clean and convert the data for transforming the data in desired format.
Scikit-Learn	svm	Support Vector Regressor	Used to implement Support Vector Machines (SVM) as a Regressor
	linear_model	LogisticRegression	For feature selection
	model_selection	train_test_split	The splitting of the dataset is executed using this method as it splits it into train and test datasets.
	metrics	mean_squared_error r2_score	Used to determine the MSE and R^2 between the earthquake's actual and projected magnitude readings.
	preprocessing	OneHotEncoder	One-hot encoding categorical features for feature selection

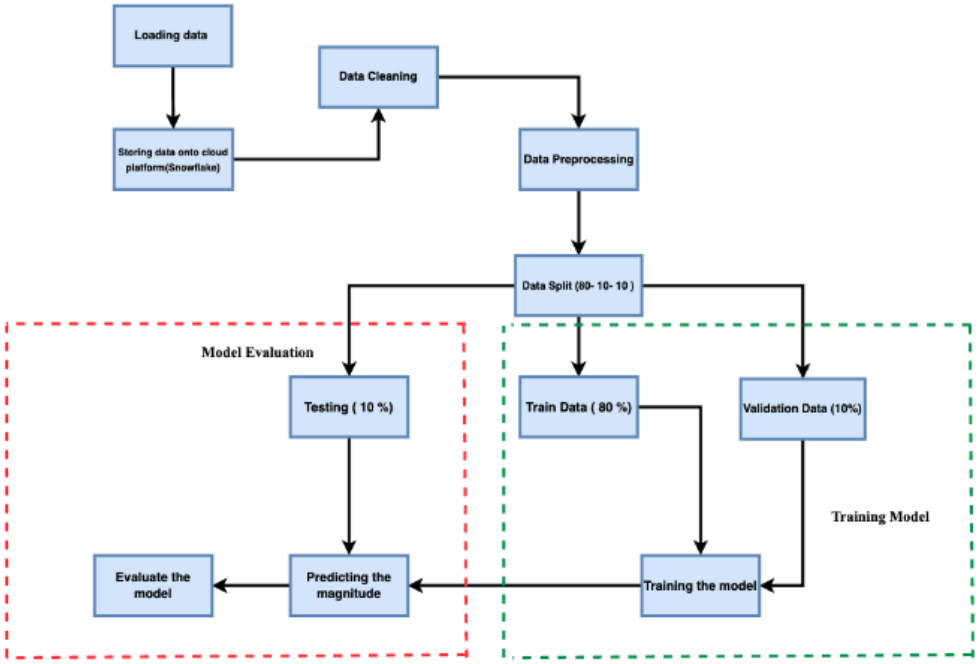
Library	Module	Method	Usage
PyTorch	utils.data	random_split DataLoader TensorDataset	Split the dataset and handle data batching
	optim	Adam	Adam optimizer for ANN
	nn	Neural Network	To implement ANNs

Model Architecture and Data Flow

The generated features Year, Count_L, and Count_S are used as input into the models. Each model takes the set of input features and uses that information to make a prediction about the corresponding magnitude value for a time frame of 1 year, 5 years, and 10 years. The expected magnitude within a certain time frame is the target variable that the model is estimating.

Figure 44

General Data Flow Diagram



The data flow to all three models begin the same, with transferring the earthquake data into a cloud-based data warehouse (such as Snowflake). The information is then prepped and prepared so that it may be used by the model. Eighty percent of the cleaned and prepared data is utilized for training, ten percent for validation, and ten percent for testing. The data is then sent to each of the individual models being trained. A generic diagram describing this overall data flow is provided in Figure 44.

The Random Forest model combines a set of decision trees to produce a more precise model. In a decision tree, the criteria for data splitting at each node is the feature that provides the greatest information gain. The procedure is repeated until the tree reaches a predetermined depth or all the information in a given branch is of the same type. In addition to a single decision tree, the trees in a Random Forest model are all trained with their own distinct subset of data and features. Bagging (bootstrap aggregating) is a method for minimizing model overfitting and increasing precision.

Additionally, the Random Forest model improves the model's diversity by introducing unpredictability through the selection of a random subset of features at each node of each tree. Figure 45 shows a diagram of the data flow for the random forests. The architecture of the random forest model can be seen in Figure 45 as a set of decision trees that are individually trained on a random subset of the data and features and then aggregated to create predictions

Figure 45

Data Flow for Random Forests

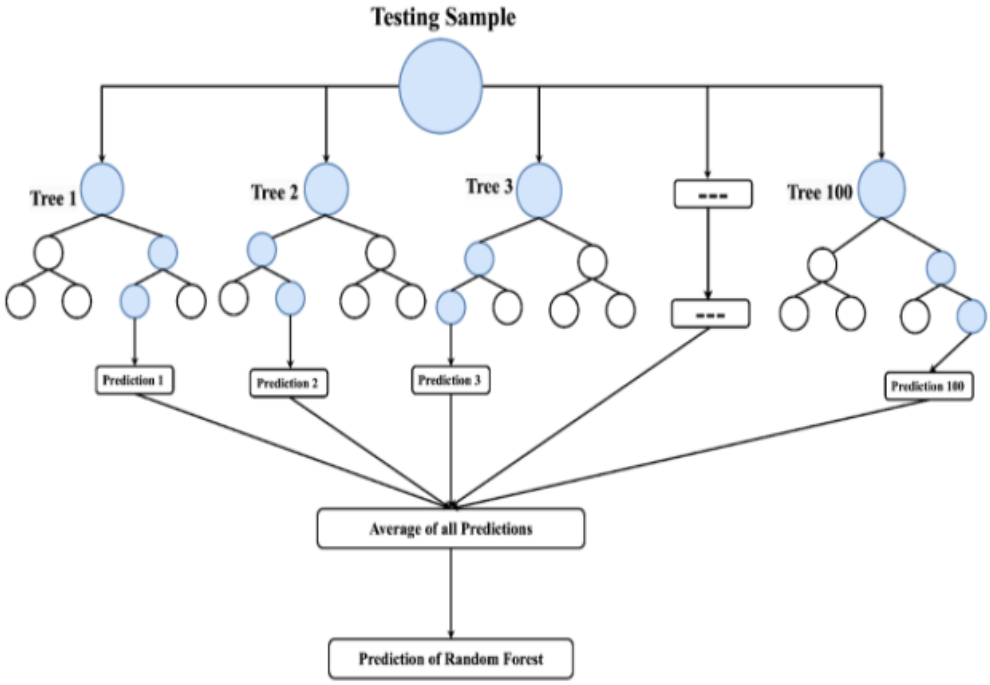
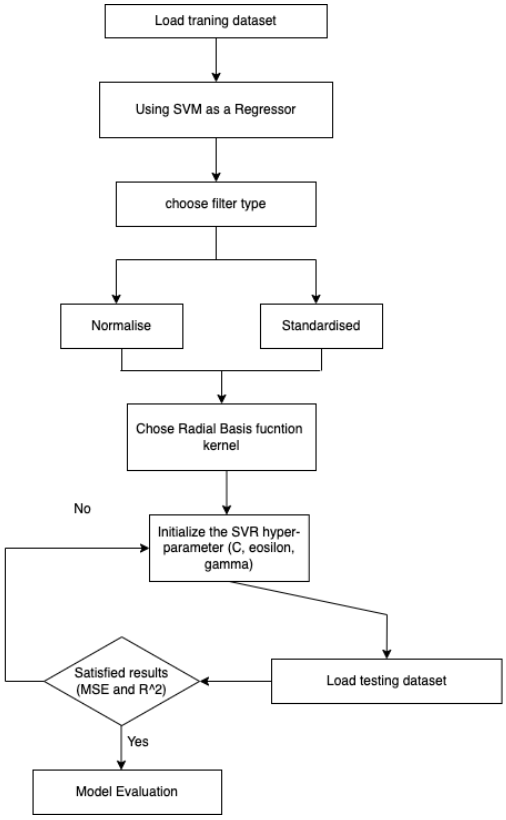


Figure 46 below shows the data flow for the SVR model for predicting magnitude of an earthquake for a rolling period of 1 year, 5 year and 10 years. In this data flow diagram the preprocessed data is split as training data to the SVR model that uses it to train the model. The next step would be selecting a filter type; either normalized or standardized filter is applied to the training data. The SVM model choses one of the kernels out of the many such as polynomial kernel, linear kernel, radial basis function kernel, etc. For this project a radial basis function kernel would leverage in producing accurate results. After choosing a kernel the SVR parameters are initialized (C, epsilon and gamma) using which SVM creates a hyperplane that best distinguishes the classes from input labeled training data. Then, using this hyperplane, predictions are made using fresh, unlabeled data. The kernel function transforms the initial input space into a higher-dimensional space where the classes can be linearly segregated as the data

passes through it. The margin, or the separation between the hyperplane and the nearest points of each class, is then used to determine the decision boundary that will produce the maximum margin.

Figure 46

Dataflow of Support Vector Machine as a Regressor (SVR)



The three neural networks were built using the exact same architecture but taking different input features and a different target time period for prediction: one year, five years, or ten years. Each model included three layers, consisting of an input layer of four neurons, one for each of the input variables, a second layer, which was also the first hidden layer, with five neurons, and a final output layer with two neurons, one for each of the two classes of the output. The two hidden layers used ReLU as their activation function, and the final layer used sigmoid to

produce the desired binary output. The equation for ReLU is given in Equation 7 (Goodfellow et al, 2016).

$$\max(0, x) \tag{7}$$

The expected target output was a binary output of class 0, where only earthquakes of magnitude less than 5.5 were expected in the time period, and class 1, which represented that the predicted maximum magnitude for the time period was over 5.5. For all three models, the training, validation, and testing data were batched into groups of 35, and trained over 1000 epochs of data. All three neural networks use an Adam optimizer, which adds both speed and momentum to the learning process of the neural network model, to improve its overall training time. Cross-entropy loss was used as the loss function for learning for all three networks.

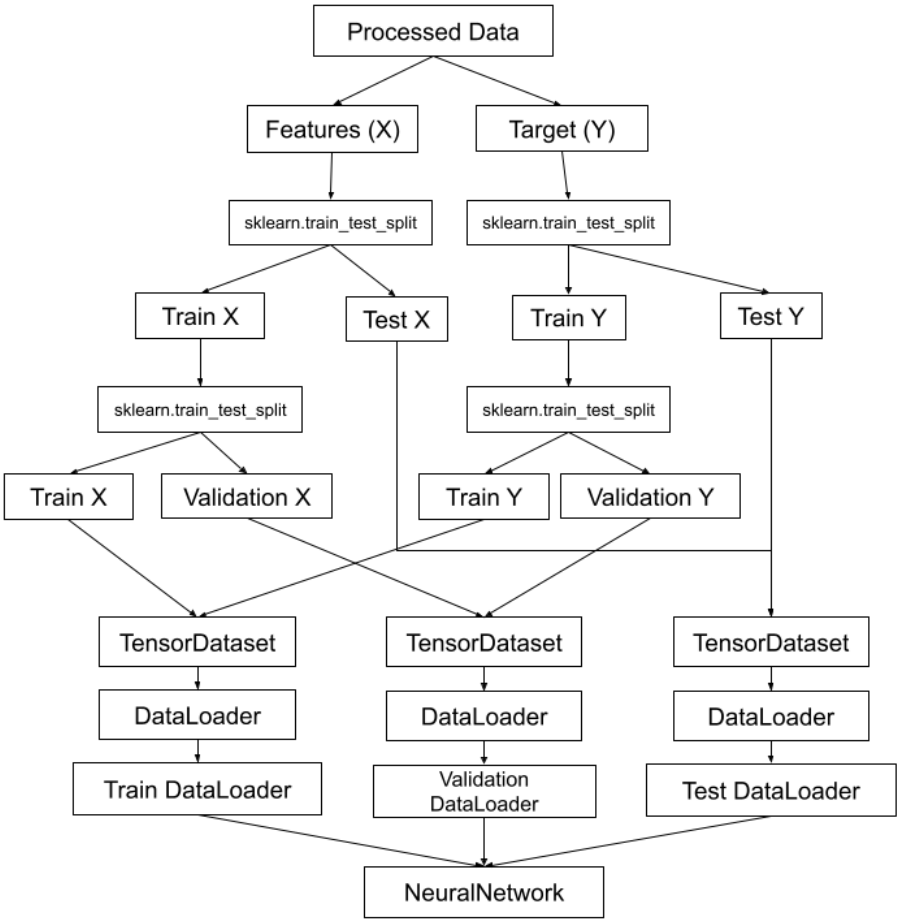
The pytorch.nn module is a framework for neural network architectures, which was extended to create a single class defining the shared, three-layer architecture for the three neural networks. This ensured all three models used the same architecture. The data was split into training, testing, and validation datasets using scikit-learn's train-test-split module, then transformed from the output numpy arrays into pandas DataFrames to create a pytorch TensorDataset object, which is arranged internally as a tensor of tensors. Finally the TensorDataset objects for each of the test, validation, and training datasets were used to create a pytorch DataLoader, which handled batching each of the dataset for the neural networks.

The torch.nn class was extended to create a custom NeuralNetwork class holding the logic for the neural networks as well as methods for tracking the training, validation accuracy and handling the training of the network over the specified number of epochs. The method would train the model for one epoch of data, then halt the backpropagation process temporarily to run the model against the validation dataset and then the testing dataset, saving the calculated loss

and accuracy for later analysis. Then the model would continue on to train for the next epoch of data. In this way, the gradual performance of the network over the training process was preserved. The NeuralNetwork class also tracked runtime from the beginning of the training cycle to its end, to give an overall estimate of expected training time.

Figure 47

Data Flow for the Artificial Neural Networks



Three NeuralNetwork classes were instantiated and given DataLoaders with different input and target variables. Their differing variables were taken from the shared processed data as subsets and passed through the data process in Figure 47, marked in the figure as Features (X)

and Target (Y), to create train, validation, and test DataLoaders, which were then passed to each of the neural networks.

The generic data flow from processed data to the DataLoader object used as input to the NeuralNetwork class is presented in Figure 47. This data flow was applied once for each of the three neural networks built, with different input features and target.

Model Comparison and Justification

The neural networks implemented predicted the expected maximum magnitude of earthquake in its time frame, either the next one year, five years, or ten years, as a binary class of either 0 for smaller than 5.5, or 1 for larger than 5.5. This was due to the limitations of predicting the probability of the exact occurrence of an earthquake along a continuous time scale, which for any given time would be close to or effectively zero due to the nature of continuous probabilities.

The predictions' target time periods of one year, five years, and ten years were selected so that the model would predict a sufficient amount of positive responses. Other time ranges considered included that of one month, or thirty days, as performed by Ahuja and Pasari (2022) for the Himalayas region, or of roughly three months, or one hundred days, as performed by Gitis and Derendyaev (2019) for the Mediterranean region and the whole of California. However, the earthquakes of note in the Southern California region for which the data for this project was pulled, happen along a time scale of several decades (Field et al, 2015), making month-by-month predictions too granular to produce meaningful, positive predictions of earthquakes over the coming decades of interest.

A magnitude of 5.5 was selected to be the threshold for larger versus smaller earthquakes, where earthquakes with a larger magnitude are considered the target for prediction. Baranov et al. (2019) selected three earthquakes to study of magnitudes 8.1, 8.6, and 9.1, while Ahuja and

Pasari (2022) considered earthquakes of magnitudes 4.0 and above. To find a middle ground and consider only those moderate earthquakes of a higher risk, but not those moderate ones with lesser risk, 5.5 was selected as the magnitude threshold for earthquakes of interest.

Regardless of model, the earthquake catalog presented for training needed to have a relatively complete record of the history of earthquake events in the forecasting region and time period. Compared to random forests and Bayesian statistical inference, neural networks require a greater amount of data in order to learn and create accurate predictions. Neural networks are particularly prone to overfitting when there are insufficient amounts of data, causing the accuracy of predictions to suffer. Additionally, the confidence of the prediction may suffer if the outputs of each successive layer are not roughly normally distributed.

The lack of data can be resolved with a simulation of earthquakes using various probability distributions like Poisson distribution, exponential distribution, or other models. However, these have the potential to overestimate weaker earthquakes due to their higher frequency of occurrence. With earthquake forecasting, the absence of a sufficient number of larger-magnitude earthquakes can cause the model to overestimate earthquakes of weaker magnitudes, skewing the overall predictions.

In exchange for training speed, lightweight calculations, and a low amount of necessary storage space, the implemented artificial neural networks are restricted to serialized, one-dimensional sets of data. They lack the ability to maintain temporal relationships in sequential data or to learn patterns of periodicity in the data, the former of which was the reason for performing a time-independent forecast of earthquake magnitudes for this study. Additionally, the earthquake data catalog used in this paper did not exhibit periodicity, but for an analysis of a time period greater than 40 years, or of different data catalogs covering larger

geographical areas, periodicity of earthquakes may come into play and require consideration. In such cases, it could be suggested to transition to a more complex architecture, such as an LSTM, that is proficient at sequence processing and at handling periodicity, in exchange for heavier computational and memory requirements.

Further, two well-known machine learning models that were included in the project are Random Forest (RF) and Support Vector Machines (SVM) as a Regressor that have different basic architectures. SVM is a linear model that extracts a hyperplane from input training data that best separates the classes. It can handle a variety of data kinds, including tabular, time-series, image, audio, and text. SVM may experience overfitting or underfitting problems, and it performs better with sparse data and small data sizes. Preprocessing is necessary due to the model's complexity, and the size of the input affects the training time, space complexity, and computing complexity. The RF ensemble approach uses parallel processing to build decision trees on subsets of data. It is effective on tabular data and large datasets, where it can capture complex relationships between variables. RF can also handle missing data and non-linear relationships. However, it may be prone to overfitting and requires more computational resources than SVM. In summary, SVM is a simple model that works well with small data sizes and various data types, while RF is more complex and suited for larger tabular datasets with non-linear relationships.

Table 7

Comparison of Architectures Between SVM, Random Forest and ANN

Characteristics	Support Vector Machine (SVM)	Random Forest	ANN
Basic Architecture	Linear model	Parallel processing	Multilayer feedforward neural network

Characteristics	Support Vector Machine (SVM)	Random Forest	ANN
Datatypes the model can process	The model can handle different types of data tabular, audio, text, images, time-series	Effective on tabular and large datasets.	Tabular, numerical, categorical, image, text
Size of data the model can work on	Works well on sparse and small datasets	Works well on large and complex datasets	Works well on large datasets
Known Issues	It faces overfitting or underfitting	Maybe prone to overfitting	Requires careful tuning of hyperparameters
Amount of processing	Take a high amount of time for processing	Takes moderate time for processing	Depends on the complexity of the network and the data
Training time	It can take hours to train	Faster training time compared to SVM	Depends on the size and complexity of the network
Computational Complexity	High computational complexity	Moderate computational complexity	Depends on the size and complexity of the network
Space Complexity	$O(N^2)$	O(k'*depth of tree)	O(n)
Non-linearity	Can handle non-linear data with kernel trick	Handles non-linearity with decision trees	Handles non-linearity through activation functions
Robustness to noise/outliers	Sensitive to noise/outliers	Robust to noise/outliers	Sensitive to noise, specially with deeper networks
Handling of high dimensional data	Can handle high-dimensional data efficiently	Well-suited for handling high-dimensional data	Flexible for handling high dimensions of data, but may get expensive
Strengths	Effective on small, well-separated datasets and non-linearly separable data with	Effective on large and complex datasets, handles non-linearity with decision trees	Excels at providing non-linearity with a flexible structure

Characteristics	Support Vector Machine (SVM)	Random Forest	ANN
	kernel trick		
Limitations	Requires careful tuning of hyperparameters, may not work well with noisy data	May be prone to overfitting, can be slow on real-time predictions.	Data-hungry, many hyperparameters, requires more computational power

Model Evaluation Methods

Evaluation metrics are crucial for the process of developing a machine learning model. They measure how well the model can make predictions about previously unseen data. The appropriate statistics are utilized and it also depends on the criteria used to evaluate the data at access. These evaluation methods are used to compare the predicted values from a regression problem to the actual values. As a regression algorithm, Random Forest Regression is also measured by a number of different measures.

Mean Absolute Error

The mean absolute error (MAE) is derived by adding up all the times the projected values were off from the actual values and averaging the results. The MAE can be calculated using Equation 8.

$$MAE = (1/n) * \sum|y\_pred - y\_actual| \quad (8)$$

Where y_pred is predicted, y_actual is what was observed, and n is the number of observations. The mean absolute error (MAE) is useful when the prediction error magnitude is more important than the direction. It's simple because it's in the same units as the dependent variable. A better performing model will have a smaller MAE, indicating that its predictions are, on average, more accurate.

Mean Squared Error

The Mean Squared Error (MSE) is another popular metric for judging the success of a regression solution. It's quite similar to the MAE but uses the squared difference between the expected and observed values. Larger errors, when squared, carry more weight in this metric, making it more error-sensitive overall. MSE can be calculated using Equation 9.

$$MSE = (1/n) * \Sigma(i = 1\ to\ n)\ (yi - \hat{y}i)\text{^}2 \quad (9)$$

The total number of observations is denoted by n above, where yi represents the actual value of the target variable at the ith observation and i represents the predicted value at the ith observation. Lower MSE values indicate higher model performance and MSE values are always positive. The problem with comparing models across datasets or target variables is that the MSE value depends on the magnitude of the target variable.

Root Mean Squared Error

Regression models are also evaluated using the Root Mean Squared Error (RMSE). The average squared deviation from the projected value is what this metric measures. It's easy to understand because it's expressed using the same units as the dependent variable. RMSE can be calculated using Equation 10.

$$RMSE = sqrt(mean((y\_true - y\_pred)\text{^}2)) \quad (10)$$

Where y_true are actual values of the target variable and y_pred are predicted values of the target variable. Similar to MSE, RMSE is more sensitive to outliers because it penalizes greater errors more severely than smaller ones. If the RMSE is smaller, then the performance is better.

R-squared (R^2)

R-squared reflects how much of the observed variation is in the dependent variable which can be accounted for by the model's independent variables. It is a measure of how effectively one variable can predict another; it is also known as the coefficient of determination. Equation 11, can help identify the value of R2.

$$\text{R-squared} = 1 - (\text{RSS} / \text{TSS}) \quad (11)$$

Where the RSS statistic estimates the difference between the expected and observed values of the dependent variable. It stands for the variance that cannot be accounted for by the independent variables. Total variation in the dependent variable is quantified by the Total Sum of Squares (TSS).

Cross-Entropy Loss

Cross-entropy loss is used to calculate the loss of the model against training, validation, and testing data sets. The cross-entropy loss compares the ground truth against the predicted class output, where only the correctly matched class is preserved.

The calculated loss during the training period is compared against the loss obtained during the validation period to determine how well the model is performing at the prediction task. The overall change in loss throughout the training period can then be plotted in a chart to determine how fast the model is learning over time, and then be used to tune the model's learning rate to produce a model that will reach a lower loss faster. whether the model is overfitting or underfitting the data.

Accuracy

In the case of earthquake forecasting, positively identifying and forecasting an upcoming large earthquake has the highest value. Therefore, accuracy was selected to measure the performance of the model. Accuracy tracks the ability of the model to identify correctly. The

formula for calculating accuracy is provided in Equation 12, where TP is the count of true positives, TN is the count of true negatives, and N is the total count of data under consideration.

$$\frac{TP+TN}{N} \quad (12)$$

Confusion Matrix

A confusion matrix is a way of visually representing the classification and misclassification results of a prediction. The expected results are plotted along one axis, while the predicted results are plotted on the other, producing a two-dimensional table tracking the number of overall matching and missing predictions for each category present.

Model Validation and Evaluation Results

Random forest Regressor

It is important to use accurate measures to measure the precision and efficiency of the Random Forest and Support Vector Regression (SVR) models before evaluating their performance for earthquake prediction. Mean Squared Error (MSE) is used here; it measures the typical squared deviation between predicted and observed earthquake intensities. More precise earthquake predictions correspond to smaller MSE values, suggesting a better match between the model and the data. Mean squared error (MSE) values for the Random Forest model indicate its effectiveness over various time spans for making predictions. The 1-year MSE of 0.0171 suggests the model successfully captures variations in earthquake activity over very short time periods. The 5-year prediction has a slightly higher MSE, of 0.0438. This indicates that the model's accuracy could be compromised when trying to foretell earthquakes in the distant future. The MSE drops to 0.0186 for the 10-year prediction, indicating that the model can still offer an accurate prediction of earthquake frequency and magnitude over the long term. Overall, the

Random Forest model delivers reliable results with low MSE values, proving its efficacy in earthquake prediction tasks.

When comparing the SVR model to the Random Forest model, the MSE values show how well each works. In comparison to Random Forest, this model has a larger MSE (0.03397) for a prediction made 1 year into the future. This means that the SVR model may have marginally larger prediction errors when making short-term predictions about earthquakes. The mean squared error (MSE) for the prediction over the next five years is also moderate, at 0.03507. For a 10-year prediction, however, the SVR model shows substantial improvement, with an MSE of just 0.0124. That the SVR model can produce more reliable earthquake prediction results over longer time periods is supported by these results. In most cases, the MSE is smaller for the Random Forest model, showing that the model accurately reflects short-term seismic trends. However, the SVR model maintains its performance overall prediction horizons, although with somewhat higher MSE values than the Random Forest model. Increasing R2 suggests a more precise model. With a one-year prediction horizon, the Random Forest Regression model surpasses the SVR model in R2 scores. SVR's R2 values are much higher than Random Forest's for both 5- and 10-year prediction timeframes. The RMSE of the random forest model is 0.1306 over a horizon of 1 year. This indicates that the model has an average inaccuracy of about 0.1306 units when making predictions compared to the actual values for this time period. The RMSE increases to 0.2092 when considering the 5 year timeframe. As a result, the average inaccuracy of the model’s prediction. In the SVR model, the RMSE values for each prediction horizon are 0.1843, 0.1873, and 0.1117. Average mistakes in predictions made by the SVR model can be analyzed using these RMSE values. In most circumstances, the random forest model outperforms the SVR model, as measured by their respective RMSE values. The RMSE

values for the random forest model were lower across all instances for the three different time periods being predicted. For MAE for the random forest, the model is 0.1004 for one year, for a five-year timeframe MAE is 0.1606, and at last for 10 years is 0.0816. The average difference between the model predictions could be seen here. A smaller MAE is indicative of improved performance since it shows that the model is making predictions that are on average closer to true values. The MAE for the SVR model is 0.1485 for the first year, for five years it is 0.1618 and for 10 years it is 0.1045. With lower MAE values across all three timeframes, the random forest model looks to be preferable as compared to the SVR model.

Support Vector Machine

The algorithms SVM and RF are implemented to solve a regression problem. It would help in establishing an accuracy model to predict the earthquake magnitude for a period of 1, 5 and 10 years. Both the models were evaluated against the same metric, i.e. for MSE, RMSE, MAE and R^2. Table 8 shows the comparison using evaluation methods.

Table 8

Model Comparison using Evaluation Methods

Metric	Random Forest (1 year)	Random Forest (5 years)	Random Forest (10 years)	SVR (1 year)	SVR (5 years)	SVR (10 years)
R2 Score	0.8529	0.7429	0.718	0.7069	0.7939	0.8112
RMSE	0.1306	0.2092	0.1364	0.1843	0.1873	0.1117
MAE	0.1004	0.1606	0.0816	0.1487	0.1618	0.1045
MSE	0.0171	0.0438	0.0186	0.03397	0.03507	0.0124

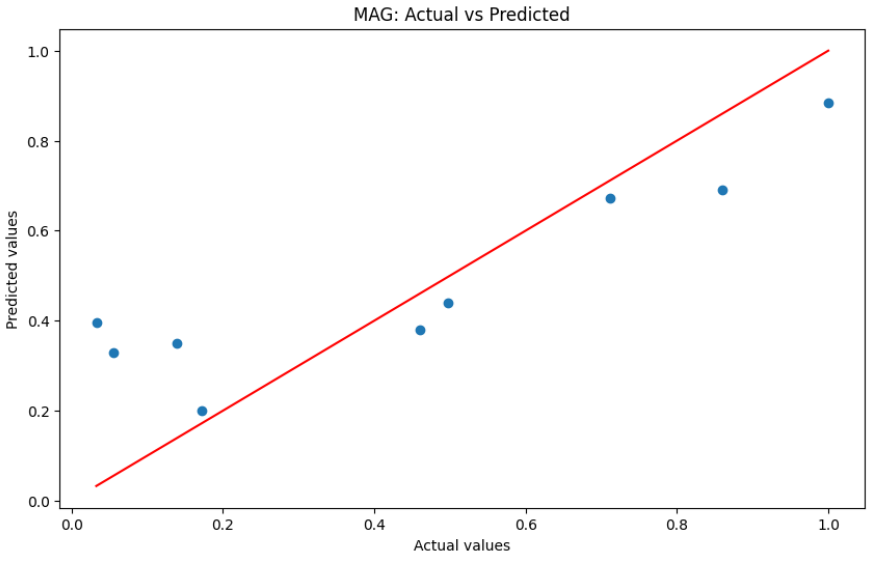
Table 8 presents a comparison of the two models to determine the best option for accurate predictions. The Mean Squared Error (MSE) values are analyzed for both Random Forest (RF)

and Support Vector Machine (SVM) models over different rolling periods. It is observed that the MSE for RF is lower than SVM for a 1-year rolling period, but for rolling periods of 5-years and 10-years, SVM has lower MSE values. Since a lower MSE indicates a better model, SVM is considered superior to RF based on this criterion.

Similarly, the R^2 scores are evaluated for both models. RF has a higher R^2 score for the 1-year rolling period, while SVM shows higher R^2 scores for the 5-year and 10-year rolling periods compared to RF. This indicates that SVM is a better fit for longer-term predictions. Overall, considering the MSE values and R^2 scores, SVM is favored over RF in terms of accuracy and performance.

Figure 48

Scatter plot for 1 - year rolling period

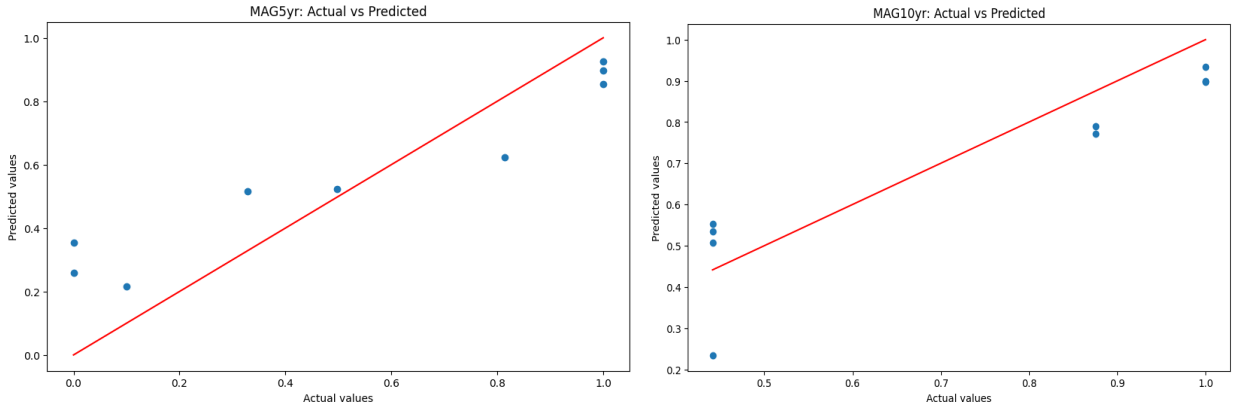


In the above figure, the scatter plot illustrates the relationship between the actual and predicted magnitudes of earthquakes over a one-year rolling period. The accurate predictions are

indicated by the red line, while the points surrounding the line reflect the level of accuracy in the predictions.

Figure 49

Scatter plot for 5 - year and 10-year rolling period



Similarly, in Figure 49 above, the scatter plot depicts the relationship between the actual and predicted magnitudes of earthquakes over a 5 - year rolling period and a 10 - year rolling period. The red line represents the precise forecasts, while the spots around the line represent the precision of the predictions.

Neural Network

An analysis of the confusion matrices for the neural networks, provided as Table 9 below, shows that the 1-year and 5-year forecast models produced false positives, but no false negatives, and were unable to correctly predict true negatives. The confusion matrix for the 10-year forecast model, on the other hand, shows both a high true negative rate and a high true positive rate, while also having a much smaller false positive rate than the models with shorter terms of

forecast. As a result, it can be concluded that the ten-year forecasting model produced the best results of the three forecasting neural networks.

Table 9

Confusion Matrix for 1-year Forecast

Actual	Predicted	
	Positive	Negative
	1-Year Forecast Model	
Positive	19	0
Negative	2	0
	5-Year Forecast Model	
Positive	24	0
Negative	7	0
	10-Year Forecast Model	
Positive	19	0
Negative	1	11

Table 10 compares the models built in this paper with each other and with the artificial neural network of similar architecture built by Ahuja and Pasari (2022). The accuracy was similar to the accuracy attained by Ahuja and Parasi (2022) for the 1-year and 10-year forecasting neural networks. Taking these results into account alongside the confusion matrices previously mentioned, the ten-year forecast performed the best both in the confusion matrix and

in terms of training speed. Although it ended with a higher cross-entropy loss than the 1-year or 5-year models, its overall classification performance was better.

Table 10

Comparison of Validation and Evaluation Metrics

Model	Test Accuracy	Cross-Entropy Loss	Training Speed (1000 epochs)
Ahuja and Pasari (2022)	90%	-	-
1-Year Forecast ANN	93.5%	0.5%	1.32s
5-Year Forecast ANN	83.9%	0.4%	1.30s
10-Year Forecast ANN	90%	0.7%	1.29s

One thing to note is that Ahuja and Pasari (2022) trained their model to predict for a time period of 100 days, which was much shorter than the three neural networks trained in this study. As the three models trained to predict for 1-year, 5-year, and 10-year time periods can be observed to have better accuracy as they predict for progressively larger time periods, it is possible that the greater number of neurons, as well as the greater number of input variables, used by Ahuja and Pasari (2022) allowed them to reach a high level of accuracy even for very short time spans. Their paper did not provide metrics for training time, so a comparison against that is omitted here.

Conclusion

The goal of this paper was to produce a lightweight, simple model that would be easy to integrate into larger programs, quick to train, and quick to run for predictions. The approach taken in this project to start simple resulted in neural networks that demonstrated both a fast runtime suitable for handling real-time data and a short training period suitable for rapid

re-training and iteration. The simple, easily automatable data pipeline makes it easy to integrate the models produced with this approach into other programs, and to update the model in real-time with data from consumed earthquake catalogs. The models in this project demonstrated good levels of test accuracy and prediction results for the selected time frames.

Limitations

While the models were found to overfit, this was primarily due to the limitations of the dataset used to train the model, which provided too few targets–in this case, earthquakes with larger magnitude. This is a shared challenge across all earthquake forecasting model training approaches

Future Scope

Future directions to consider include changing the model into a time-dependent model by adding mathematical formulas such as fault strain, training on simulated data to overcome the shortage of targets found in real-world data, and testing different models such as long short-term memory models specialized at sequence processing.

References

Asim, K.M., Martínez-Álvarez, F., Basit, A., Iqbal, T.*(2017).* Earthquake magnitude prediction in Hindukush region using machine learning techniques. *Natural Hazards* 85, 471–486. https://doi.org/10.1007/s11069-016-2579-3

Budiman, K., & Ifriza, N. Y. (2021). Analysis of earthquake forecasting using random forest. *Journal of Soft Computing Exploration*, 2(2), 153-162. https://doi.org/10.52465/joscex.v2i2.51

Bao, Z., Zhao, J., Huang, P., Yong, S., & Wang, X. F. (2021). A Deep Learning-Based Electromagnetic Signal for Earthquake Magnitude Prediction. *DOAJ (DOAJ: Directory of Open Access Journals)*, *21*(13), 4434. https://doi.org/10.3390/s21134434

Cano, E.V., Akram, J., & Peter, D.B. (2021). Automatic seismic phase picking based on unsupervised machine-learning classification and content information analysis, *Geophysics*,86(4), 299-315. https://doi.org/10.1190/geo2020-0308.1

Chen, H. J., Chen, C. C., Ouillon, G., & Sornette, D. (2021). A paradigm for developing earthquake probability forecasts based on geoelectric data. *The European Physical Journal. ST, Special Topics*, 230(1), 381–407. https://doi.org/10.1140/epjst/e2020-000258-9

Chen, H. J. and Chen, C. C. (2016). Testing the correlations between anomalies of statistical indexes of the geoelectric system and earthquakes. *Natural Hazards*, 84, 877-895, https://doi.org/10.1007/s11069-016-2460-4

Console, R., Pantosti, D., & D'Addezio, G. (2002). Probabilistic approach to earthquake prediction. *Annals of Geophysics*, 45(6), 723-731. https://doi.org/10.4401/ag-3546

Corbi, F., Sandri, L., Bedford, J., Funiciello, F., Brizzi, S., Rosenau, M., & Lallemand, S. (2019). Machine learning can predict the timing and size of analog earthquakes. *Geophysical Research Letters, 46*(3), 1303–1311. https://doi.org/10.1029/2018GL081251

Debnath, P., Chittora, P., Chakrabarti, T., Chakrabarti, P., Leonowicz, Z., Jasinski, M., Gono, R., & Jasińska, E. (2021). Analysis of earthquake forecasting in India using supervised machine learning classifiers. *Sustainability*, 13(2), 1–13.

Dumke, I., and Berndt, C. (2019). Prediction of seismic P-wave velocity using machine learning. *Solid Earth*, 10, 1989–2000. https://doi.org/10.5194/se-10-1989-2019

Essam, Y., Kumar, P., Ahmed, N. A., Murti, A. M., & El-Shafie, A. (2021). Exploring the reliability of different artificial intelligence techniques in predicting earthquake for Malaysia. *Soil Dynamics and Earthquake Engineering*, 147, 0267-7261. https://doi.org/10.1016/j.soildyn.2021.106826

Gentili, S., & Di Giovambattista, R. (2022). Forecasting strong subsequent earthquakes in California clusters by machine learning. *Physics of the Earth and Planetary Interiors*, 327, 106879. https://doi.org/10.1016/j.pepi.2022.106879

Gitis, V.G, & Derendyaev, A.B. (2019). Machine Learning Methods for Seismic Hazards Forecast. *Geosciences*, 9(7), 308. https://doi.org/10.3390/geosciences9070308

Hanks, T. C., & Kanamori, H. (1979). A moment magnitude scale. *Journal of Geophysical Research: Solid Earth*, 84(B5), 2348-2350. http://dx.doi.org/10.1029/JB084iB05p02348

He, Q., Wang, M., & Liu, K. (2021). Rapidly assessing earthquake-induced landslide susceptibility on a global scale using random forest. *Geomorphology*, 391,0169-555. https://doi.org/10.1016/j.geomorph.2021.107889

Jain, R., Nayyar, A., Arora, S., & Gupta, A. (2021). A comprehensive analysis and prediction of earthquake magnitude based on position and depth parameters using machine and deep learning models. *Multimedia Tools and Applications,* 80, 28419–28438. https://doi.org/10.1007/s11042-021-11001-z

Jakkula, V. (2006). *Tutorial on support vector machine (SVM)*. School of EECS, Washington State University, 37(2.5), 3.

Kelleher, J. D., Mac Namee, B., & D'Arcy, A. (2020). *Fundamentals of Machine Learning for Predictive Data Analytics, second edition: Algorithms, Worked Examples, and Case Studies*. MIT Press.

Kubo, H., Kunugi, T., Suzuki, W., Suzuki, S., & Aoi, S. (2020). Hybrid predictor for ground-motion intensity with machine learning and conventional ground motion prediction equation. *Scientific Reports*, 10, 11871. https://doi.org/10.1038/s41598-020-68630-x

Kwon, O., & Elnashai, A. S. (2006). A comparative study of regression analysis and neural networks for peak ground acceleration prediction. *Earthquake Engineering & Structural Dynamics,* 35(2), 139-161.

Mancini, S., Segou, M., Werner, M. J., Parsons, T., Beroza, G., & Chiaraluce, L. (2022). On the use of high‐resolution and deep‐learning seismic catalogs for short‐term earthquake forecasts: potential benefits and current limitations. *Journal of Geophysical Research. Solid Earth, 127*(11). https://doi.org/10.1029/2022JB025202

Mendicelli, A., Falcone, G., Acunzo, G., Mori, F., Naso, G., Peronace, E., Porchia, A., Romagnoli, G. & Moscatelli, M. (2022). Italian seismic amplification factors for peak

ground acceleration and peak ground velocity, *Journal of Maps*, 18(2), 497-507. https://doi.org/10.1080/17445647.2022.2101947

Mignan, A., Ouillon, G., Sornette, D., & Freund, F. (2021). Global Earthquake Forecasting System (GEFS): The challenges ahead. *The European Physical Journal. ST, Special Topics, 230*(1), 473–490. https://doi.org/10.1140/epjst/e2020-000261-8

Mir, A.A., Celebi, F.V., Alsolai, H., Qureshi, S.A., Rafique, M., Alzahrani, J.S., Mahgoub, H., & Hamza, M.A. (2022). Anomalies prediction in radon time series for earthquake likelihood using machine learning-based ensemble model. *IEEE Access*, 10, 37984–37999. https://doi.org/10.1109/ACCESS.2022.3163291

Mizrahi, L., Nandan, S., Wiemer, S. (2021). Embracing Data Incompleteness for Better Earthquake Forecasting. *Journal of Geological Research: Solid Earth*, 126(12). https://doi.org/10.1029/2021JB022379

Mousavi, S. M., & Beroza, G. C. (2020). A Machine‐Learning Approach for Earthquake Magnitude Estimation. *Geophysical Research Letters*, *47*(1). https://doi.org/10.1029/2019gl085976

Pardo, E., Garfias, C., & Malpica, N. (2019). Seismic Phase Picking Using Convolutional Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9), 7086-7092. https://doi.org/10.1109/TGRS.2019.2911402

Rouet-Leduc, B., Hulbert, C., Lubbers, N., Barros, K., Humphreys, C. J., & Johnson, P. A. (2017). Machine learning predicts laboratory earthquakes. *Geophysical Research Letters, 44*(18), 9276–9282. https://doi.org/10.1002/2017GL074677

Rundle, J. B., Stein, S., Donnellan, A., Turcotte, D. L., Klein, W., & Saylor, C. (2021). The complex dynamics of earthquake fault systems: new approaches to forecasting and

nowcasting of earthquakes. *Reports on Progress in Physics, 84*(7), 76801–. https://doi.org/10.1088/1361-6633/abf893

Tezcan, J. & Cheng, Q. (2012). Support vector regression for estimating earthquake response spectra. *Bulletin of Earthquake Engineering,* 10, 1205–1219. https://doi.org/10.1007/s10518-012-9350-2

Xiong, P., Tong, L., Zhang, K., Shen, X., Battiston, R., Ouzounov, D., Iuppa, R., Crookes, D., Long, C., & Zhou, H. (2021). Towards advancing the earthquake forecasting by machine learning of satellite data. *The Science of the Total Environment*, 771, 145256–145256. https://doi.org/10.1016/j.scitotenv.2021.145256

Velasco Herrera, V.M., Rossello, E.A., Orgeira, M.J., Arioni, L., Soon, W., Velasco, G., Rosique-de la Cruz, L., Zúñiga, E. and Vera, C. (2022). Long-Term Forecasting of Strong Earthquakes in North America, South America, Japan, Southern China and Northern India With Machine Learning. *Frontiers in Earth Science*, 10. https://doi.org/10.3389/feart.2022.905792

Appendix

Source Code

https://drive.google.com/drive/folders/1RLE-_ZLRspwk3h_9Q-9IilQDSp_ui5xW