**COURSE: ARTIFICIAL INTELLIGENCE**

**TITLE: EARTHQUAKE PREDICTION MODEL USING PYTHON**

**PHASE 4 SUBMISSION : DEVELOPMENT PART 2**

**Team members:**

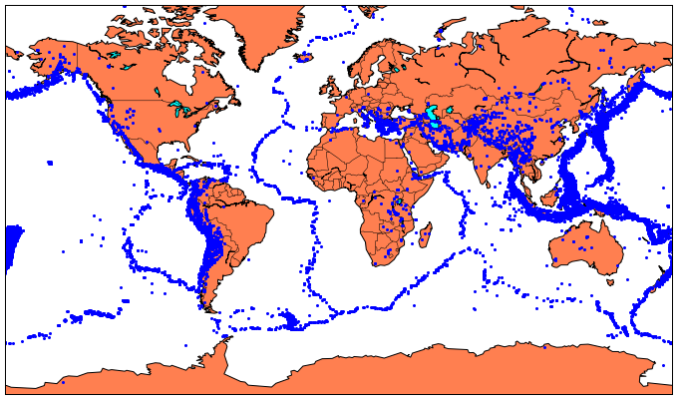
**Shravya. G. P (au311521106091)**

**Sakthi Sowmiya. S (au311521106083)** **Rekha Sekar (au311521106075)**

**Saali Deivanai.L (au 311521106080)**

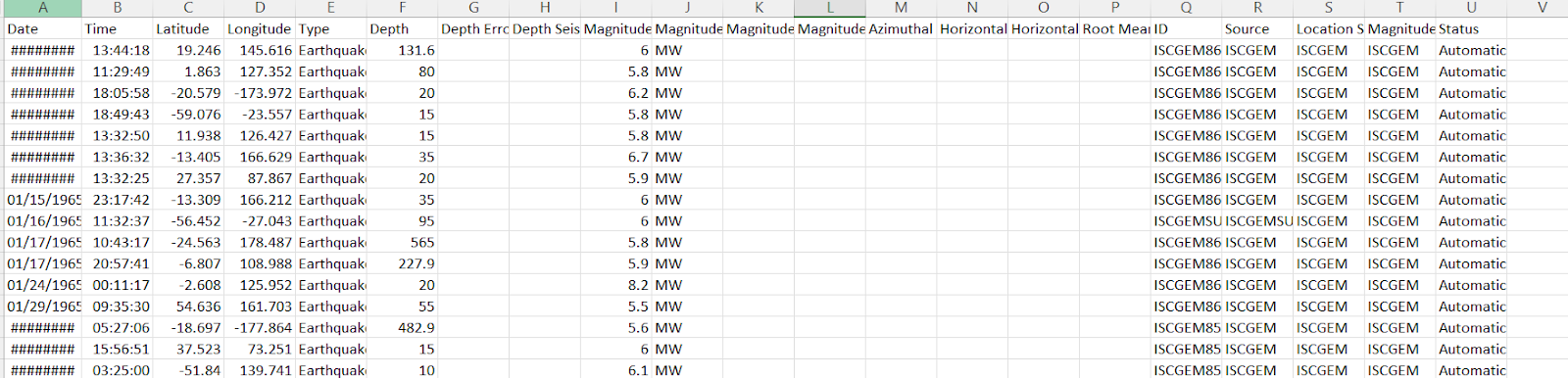
**INTRODUCTION:**

Predicting earthquakes using python, or any other programming language, is a challenging scientific problem because short-term earthquake prediction remains uncertain and not currently feasible with high accuracy. Earthquake prediction often involves making forecasts about the occurrence, timing, and location of future seismic events, which is a complex task due to the unpredictable nature of earthquakes. Nevertheless, researchers use python and various data analysis and machine learning techniques to work on related tasks like earthquake forecasting and seismic hazard assessment.



**DATA SET LINK:** [**https://www.kaggle.com/datasets/usgs/earthquake\*database**](https://www.kaggle.com/datasets/usgs/earthquake-database)

**DATA SET:**



**OVERVIEW:**

Here's an overview of the project:

1. Data Collection:

   - Start by gathering earthquake-related data. You can use earthquake catalogs, seismic sensor data, and various geological features as potential data sources.

   - Consider using APIs or datasets from organizations like the United States Geological Survey (USGS) or other geological agencies.

2. Data Preprocessing:

   - Clean the data by handling missing values and outliers.

   - Convert data into a suitable format for analysis, which may involve converting timestamps and geographical coordinates into meaningful features.

   - Consider data augmentation techniques if you have limited data.

3. Feature Engineering:

   - Feature engineering is a crucial step. You need to extract relevant features that can help your model make accurate predictions.

   - Common earthquake-related features may include seismic activity history, fault line proximity, geological composition, and geospatial information.

   - You can calculate additional features like earthquake magnitude trends, frequency, and patterns.

4. Feature Selection:

   - After engineering a wide range of features, you need to select the most relevant ones to improve model performance and reduce computational complexity.

   - Techniques like correlation analysis, feature importance from tree-based models, or domain knowledge can guide the feature selection process.

5. Model Selection:

   - Choose an appropriate machine learning or deep learning model for earthquake prediction. Common choices include:

     - Random Forest, Gradient Boosting, or other ensemble methods.

     - Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks for time series data.

   - You may also consider hybrid models that combine multiple techniques.

6. Model Training:

   - Split the data into training, validation, and testing sets.

   - Train your selected model(s) on the training data. Tune hyperparameters and use cross-validation to optimize the model's performance.

   - Regularly monitor for overfitting and adjust the model accordingly.

7. Evaluation and Metrics:

   - Assess your model's performance using appropriate metrics, such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), or area under the Receiver Operating Characteristic curve (AUC-ROC).

   - Consider the practical implications of your predictions, such as false positives and false negatives.

8. Deployment:

   - Once your model is trained and performs well, you can deploy it in a real-time or near-real-time environment where it can continuously make predictions based on incoming data.

9. Monitoring and Maintenance:

   - Regularly monitor the model's performance in a production environment and retrain it as needed with fresh data.

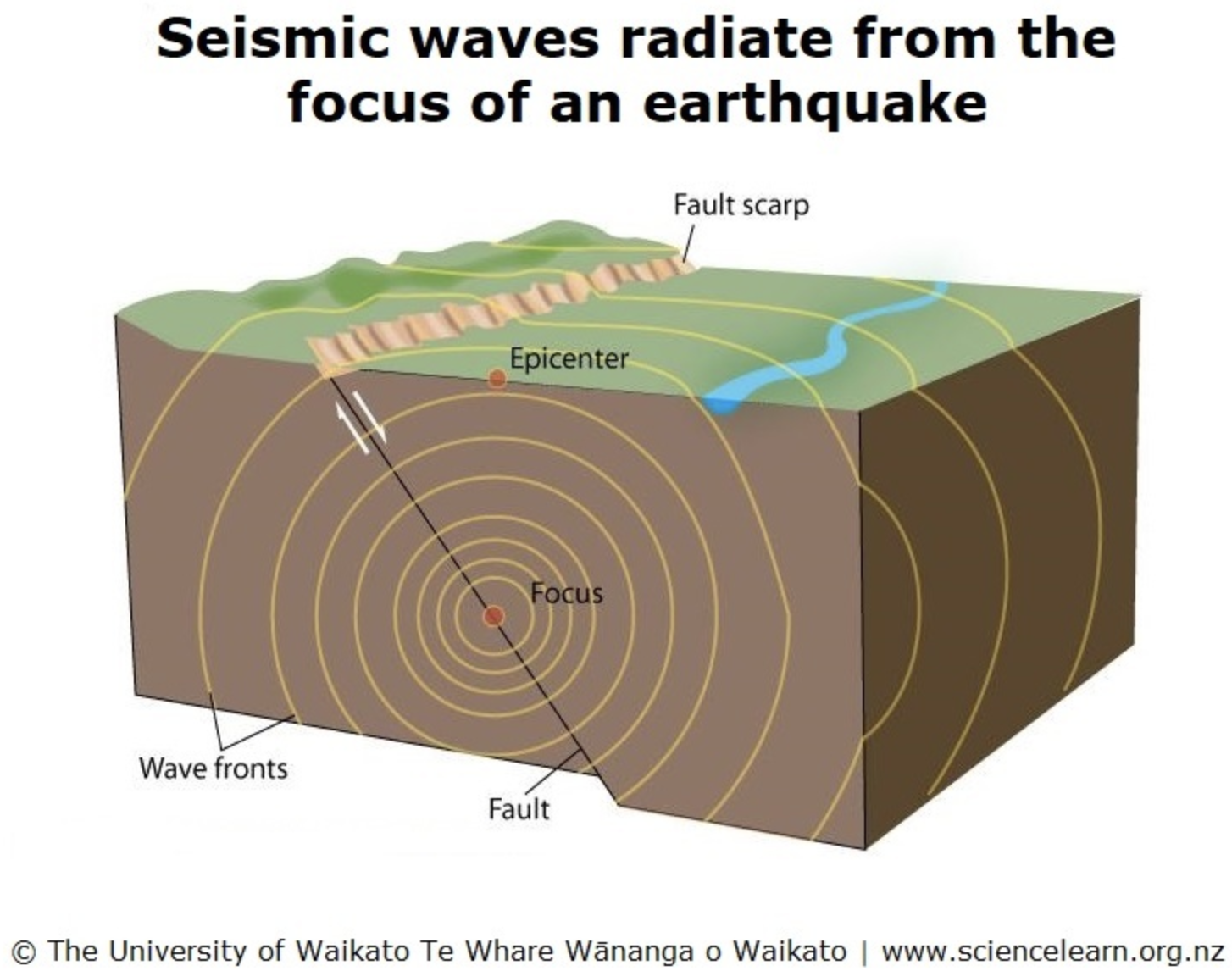
   - Stay updated on advancements in earthquake prediction research and incorporate improvements into your model.

10. Documentation and Reporting:

    - Document your project thoroughly, including the data sources, preprocessing steps, feature engineering, model selection, and performance metrics.

    - Create clear reports and visualizations to communicate your findings and results.

Remember that earthquake prediction is a complex problem, and accurate predictions can be challenging. It's essential to consult domain experts and continuously refine your model to improve its accuracy and reliability. Additionally, you may need to comply with ethical and regulatory considerations, especially when dealing with seismic data.



**PROCEDURE:**

**FEATURE SELECTION:**

Feature selection is a crucial step in building a machine learning model for earthquake prediction. The choice of features can significantly impact the model's performance. Here's a program example using python with libraries like Pandas, NumPy, and Scikit-Learn to perform feature selection. Note that this is a simplified example, and in a real project, you would need to thoroughly analyze your data and domain knowledge to choose the most relevant features.

CODE

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestRegressor

from sklearn.feature\_selection import SelectFromModel

# Load your earthquake dataset

data = pd.read\_csv('earthquake\_data.csv')

# Separate the features (X) and target (y)

X = data.drop(columns=['earthquake\_occurred'])

y = data['earthquake\_occurred']

# Initialize a Random Forest Regressor model

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

# Fit the model to the data to perform feature selection

model.fit(X, y)

# Use feature importances to select the most important features

feature\_importances = model.feature\_importances\_

feature\_names = X.columns

feature\_selection = SelectFromModel(model, threshold='median')

feature\_selection.fit(X, y)

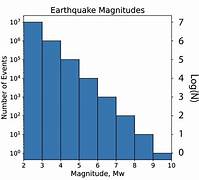
# Get the selected features

selected\_features = X.columns[feature\_selection.get\_support()]

# Print the selected features

print("Selected Features:")

print(selected\_features)



1. Load your earthquake dataset (replace `'earthquake\_data.csv'` with your actual data file).

2. Separate the features (X) and the target (y), where `'earthquake\_occurred'` is the binary indicator for whether an earthquake occurred.

3. Initialize a Random Forest Regressor model. Random forests are commonly used for feature selection because they can provide feature importances.

4. Fit the model to the data.

5. Calculate feature importances using the model.

6. Use the `SelectFromModel` class to select features based on their importance, with a threshold set to the median importance.

7. Get the selected features based on the threshold.

This python will give you a list of features that the model deems most important for predicting earthquakes. You can adjust the model and threshold according to your data and problem requirements. Additionally, domain knowledge can help you further refine the feature selection process.

**MODEL TRAINING:**

Model training is a critical step in building an earthquake prediction model. In this example, I'll demonstrate how to train a Random Forest classifier using python and Scikit-Learn. This is a simplified example, and you should customize it for your specific dataset and problem. Ensure you have your data preprocessed, and feature engineering and feature selection steps have been completed.

CODE

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

# Load your preprocessed and feature-selected dataset

data = pd.read\_csv('preprocessed\_earthquake\_data.csv')

# Separate the features (X) and target (y)

X = data.drop(columns=['earthquake\_occurred'])

y = data['earthquake\_occurred']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize a Random Forest classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the model on the training data

clf.fit(X\_train, y\_train)

# Make predictions on the test data

y\_pred = clf.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

# Print the evaluation metrics

print("Accuracy:", accuracy)

print("Classification Report:\n", report)

1. Load your preprocessed and feature-selected dataset (replace `'preprocessed\_earthquake\_data.csv'` with your actual preprocessed data file).

2. Separate the features (X) and the target (y), where `'earthquake\_occurred'` is the binary indicator for whether an earthquake occurred.

3. Split the data into training and testing sets using `train\_test\_split`. Adjust the `test\_size` parameter to set the portion of data for testing.

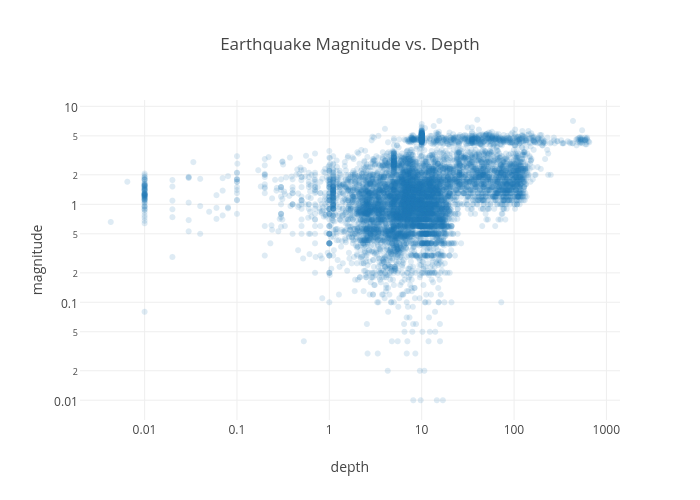
4. Initialize a Random Forest classifier. You can adjust hyperparameters such as `n\_estimators` and other parameters according to your specific needs.

5. Train the model on the training data using `clf.fit(X\_train, y\_train)`.

6. Make predictions on the test data using `clf.predict(X\_test)`.

7. Evaluate the model's performance using metrics like accuracy and a classification report.

Customize the hyperparameters and model selection based on your data and the nature of your earthquake prediction problem. Additionally, you may want to consider other evaluation metrics, such as precision, recall, F1-score, or area under the ROC curve, depending on your specific requirements.



**MODEL EVALUATION:**

Model examination, often referred to as model evaluation or model assessment, is a crucial step in the machine learning pipeline. It involves assessing the performance of your trained earthquake prediction model to determine how well it generalizes to unseen data. Several evaluation metrics and techniques can help you understand the model's strengths and weaknesses. Here are details on how to examine your model:

1. Split the Data:

   - Divide your dataset into at least two subsets: a training set and a testing set. The training set is used for model training, while the testing set is reserved for evaluation. You can also consider using a validation set if you need to tune hyperparameters.

2. Select Evaluation Metrics:

   - Choose appropriate evaluation metrics based on the nature of your problem. Common metrics for earthquake prediction include:

     - Accuracy: The proportion of correctly predicted earthquakes.

     - Precision: The ability of the model to correctly predict earthquakes (True Positives) relative to all earthquake predictions.

     - Recall (Sensitivity): The ability of the model to correctly predict earthquakes (True Positives) relative to all actual earthquakes.

     - F1-Score: The harmonic mean of precision and recall, balancing both metrics.

     - ROC-AUC: Area under the Receiver Operating Characteristic curve, which is useful for binary classification problems.

     - Mean Absolute Error (MAE) or Root Mean Square Error (RMSE): Common for regression tasks, measuring the average magnitude of prediction errors.

3. Model Evaluation:

   - Use the chosen metrics to evaluate the model's performance on the testing set. You can use libraries like Scikit-Learn to calculate these metrics.

CODE

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score, mean\_absolute\_error, mean\_squared\_error

# Example for a classification model

y\_true = ...  # True labels

y\_pred = ...  # Predicted labels

accuracy = accuracy\_score(y\_true, y\_pred)

precision = precision\_score(y\_true, y\_pred)

recall = recall\_score(y\_true, y\_pred)

f1 = f1\_score(y\_true, y\_pred)

roc\_auc = roc\_auc\_score(y\_true, y\_pred)

# Example for a regression model

y\_true = ...  # True values

y\_pred = ...  # Predicted values

mae = mean\_absolute\_error(y\_true, y\_pred)

rmse = mean\_squared\_error(y\_true, y\_pred, squared=False)

4. Confusion Matrix:

   - In binary classification, the confusion matrix provides a detailed breakdown of True Positives, True Negatives, False Positives, and False Negatives. It can help you understand the model's performance.

5. Visualizations:

   - Create visualizations, such as ROC curves, precision-recall curves, or calibration plots, to gain deeper insights into the model's behavior and performance.

6. Cross-Validation:

   - If your dataset is limited, consider using cross-validation techniques like k-fold cross-validation to assess how well your model generalizes across different subsets of data.

7. Model Robustness:

   - Test the model's robustness by introducing noise or perturbations to the input data and evaluating its performance under these conditions.

8. Domain Expert Consultation:

   - Consult with domain experts to validate whether the model's predictions align with domain knowledge and expectations.

9. Monitoring Over Time:

   - In real-time prediction systems, continuously monitor the model's performance and re-evaluate it with fresh data. Implement a system for model retraining as needed.

10. A/B Testing (for deployment):

    - If your model is deployed in a live system, use A/B testing or other experimental designs to compare the performance of the model with different versions or against other prediction methods.

**PREDICTING THE DATA SET:**

The evaluation of a predicted dataset is a crucial step to assess the performance and reliability of your model's predictions. Here's a guide on how to evaluate the predicted dataset for your earthquake prediction model:

1. Load the Predicted Dataset:

   - Load the dataset containing the model's predictions and any associated features used for prediction. This dataset should ideally be separate from the training and testing datasets used during model development.

2. Load the True Labels or Ground Truth:

   - If available, load the true earthquake occurrence labels or ground truth data corresponding to the same time periods or geographical locations as your predictions. This will be used for direct model evaluation.

3. Select Evaluation Metrics:

   - Choose appropriate evaluation metrics based on the nature of your problem (e.g., classification or regression). The choice of metrics depends on the format of your predictions and ground truth data (e.g., binary labels, numerical values).

   - For classification tasks, common metrics include accuracy, precision, recall, F1-score, and ROC-AUC.

   - For regression tasks, common metrics include Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared.

4. Calculate Evaluation Metrics:

   - Compare the model's predictions with the true labels or ground truth data to calculate the chosen evaluation metrics. You can use libraries like Scikit-Learn for this purpose.

   For a classification task in python:

   CODE

   from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

   y\_true = ...  # True labels

   y\_pred = ...  # Predicted labels

   accuracy = accuracy\_score(y\_true, y\_pred)

   precision = precision\_score(y\_true, y\_pred)

   recall = recall\_score(y\_true, y\_pred)

   f1 = f1\_score(y\_true, y\_pred)

   roc\_auc = roc\_auc\_score(y\_true, y\_pred)

   For a regression task in python:

   CODE

   from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

   y\_true = ...  # True values

   y\_pred = ...  # Predicted values

   mae = mean\_absolute\_error(y\_true, y\_pred)

   rmse = mean\_squared\_error(y\_true, y\_pred, squared=False)

5. Confusion Matrix (for classification):

   - If you are evaluating a classification model, analyze the confusion matrix to understand the distribution of true positives, true negatives, false positives, and false negatives.

6. Visualizations:

   - Create visualizations, such as ROC curves or precision-recall curves for classification, or scatter plots for regression, to gain insights into the model's behavior and performance.

7. Domain Expert Validation:

   - Consult with domain experts to validate the model's predictions and ensure they align with domain knowledge and expectations.

8. Comparison with Baselines:

   - Compare your model's performance metrics with baseline models or alternative prediction methods to assess its effectiveness.

9. Consider Ethical and Practical Implications:

   - Consider the practical implications of your model's predictions, such as false positives and false negatives, and assess whether the model's performance is acceptable within these contexts.

10. Continuous Monitoring (for deployment):

    - If your model is deployed in a real-time prediction system, continuously monitor its performance and periodically re-evaluate it with fresh data. Implement a system for model retraining or updates as necessary.

**FEATURE ENGINEERING:**

Feature engineering is a critical step in building an effective earthquake prediction model. It involves creating new features from your raw data or transforming existing features to make them more informative. Here are some feature engineering scopes and techniques relevant to earthquake prediction:

1. Time-Based Features:

   - Temporal Trends: Calculate moving averages, exponential moving averages, or other time-based statistics to capture trends in seismic activity.

   - Time Lags: Include lagged values of seismic features to capture the temporal relationship between past and future events.

2. Geospatial Features:

   - Distance to Fault Lines: Calculate the distance of each data point to known fault lines or tectonic plate boundaries.

   - Geological Features: Incorporate information about the geological composition of the region, such as soil types, rock densities, or landform features.

3. Seismic Features:

   - Magnitude Aggregation: Summarize earthquake magnitudes over specific time periods or areas.

   - Frequency Analysis: Use Fourier or wavelet transformations to identify seismic frequency patterns.

   - Seismic Energy: Calculate the energy release associated with seismic events.

4. Clustering and Spatial Aggregation:

   - Cluster data points based on geographical proximity and aggregate statistics within clusters.

   - Identify regions with high seismic activity and encode them as categorical features.

5. Historical Earthquake Data:

   - Include historical earthquake occurrence data, such as the number of earthquakes in a region over the past year.

   - Calculate earthquake recurrence intervals.

6. Meteorological Data:

   - Incorporate meteorological data like temperature, humidity, and atmospheric pressure, which can affect seismic activity.

   - Extract seasonal and weather-related patterns.

7. Geodetic Data:

   - Utilize data from GPS sensors to monitor ground deformation, which can be a precursor to seismic activity.

   - Extract features related to crustal deformation and strain rates.

8. Topographic Data:

   - Use digital elevation models (DEMs) to extract elevation, slope, and aspect features.

   - Consider features related to surface roughness and terrain characteristics.

9. Social and Economic Factors:

   - Explore features related to population density, land use, infrastructure, and socio-economic factors, which can impact earthquake vulnerability and response.

10. Time of Day and Seasonality:

    - Encode diurnal and seasonal patterns, as seismic activity may vary at different times of day or year.

11. Feature Scaling and Normalization:

    - Ensure all features are on the same scale to prevent dominance by certain features. Techniques like min-max scaling or standardization can be helpful.

12. Interactions and Polynomial Features:

    - Create interaction terms between relevant features or add polynomial features to capture non-linear relationships.

13. Domain-Specific Features:

    - Consult with geologists, seismologists, and domain experts to identify domain-specific features that are crucial for earthquake prediction.

**VARIOUS FEATURES TO PERFORM MODEL TRAINING:**

When preparing for model training in a machine learning project, you need to understand the various features and data attributes that are crucial for the training process. Features are the input variables that the model uses to make predictions. Here are various types of features and considerations for model training:

1. Numerical Features:

   - These are continuous numerical values like earthquake magnitude, depth, latitude, and longitude.

   - Ensure that numerical features are properly scaled and centered to have similar ranges and units.

2. Categorical Features:

   - These are non-numeric variables like the earthquake's type (e.g., tectonic, volcanic, or induced) or the region.

   - Use techniques like one-hot encoding to convert categorical features into numerical format for the model.

3. Temporal Features:

   - These features involve time-related information, such as the time of day, day of the week, or year.

   - Temporal features are often transformed into numerical formats, like timestamps or time differences.

4. Spatial Features:

   - These features relate to geographical information, such as latitude, longitude, elevation, distance to fault lines, or seismic sensor locations.

   - Ensure that spatial features are consistent in terms of units and scales.

5. Derived Features:

   - Features created through mathematical operations or domain-specific calculations. For example, you might derive seismic activity intensity by combining magnitude and depth.

6. Transformed Features:

   - Features obtained by applying mathematical functions, such as logarithms, square roots, or polynomial transformations, to the original data to capture non-linear relationships.

7. Aggregated Features:

   - Features that summarize data over a specific time window or region. For example, calculating the average magnitude of earthquakes in a region over the past month.

8. Interaction Features:

   - Features created by combining two or more variables to capture their interaction. For example, the product of earthquake magnitude and depth.

9. Weather Features:

   - If you're using meteorological data, features like temperature, humidity, and atmospheric pressure can be important.

   - These features may require special preprocessing, such as time alignment with seismic data.

10. Geological Features:

    - Features related to the geological characteristics of the region, including soil type, rock density, or seismic hazard maps.

11. Social and Economic Features:

    - Features related to population density, building types, infrastructure quality, and disaster preparedness can be essential for assessing earthquake risk.

12. Target-Related Features:

    - Features that directly or indirectly relate to the target variable, such as historical earthquake occurrences or proximity to known active fault lines.

13. Principal Component Analysis (PCA):

    - Reducing dimensionality using PCA can be useful to capture the most important information in the data while reducing noise.

14. Feature Importance Scores:

    - Some models, like Random Forest or XGBoost, provide feature importance scores that can help you understand which features contribute most to the model's predictions.

15. External Data:

    - In some cases, external data sources, such as geological surveys, remote sensing data, or social media sentiment, can provide valuable features.