Chettinad College of Engineering & Technology

Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Domain Name : Artificial Intelligence

Project Title : Earthquake Prediction Model using Python

1.	**Name of the Student (s)**	
	S.No Name of the Student 1 Aarthi S 2 Pavithra R 3 Lathika M 4 Loganayaki M	
2.	**Name of the Guide** : Mrs.Suganya	
	Department/ Designation : CSE/AP	
	Institutional Address : Chettinad College of Engineering and Technology NH-67, Karur-Trichy Highway, Puliyur CF, Karur	
	Phone No. & Mobile No. : 6374527207	

Title: Earthquake Prediction Model

PHASE: 5

Problem Statement : The problem is to develop an earthquake prediction model using a Kaggle Dataset. The objective is to explore and understand the key features of Earthquake data, visualize the data on a world map for a global overview, split The data for training and testing, and build a neural network model to predict Earthquake magnitudes based on the given features.

ABSTRACT:

An earthquake is shaking of the surface of the Earth, which caused as the result of movable plate boundary interactions .Earthquakes are measured using remarks from seismometers with Richter magnitude scale. Ground rupture, Landslides, Soil liquefaction and Tsunami are the main effects created by earthquakes. Today's earthquake warning systems used to provide regional notification of an earthquake in progress. Many methods have been already developed for predicting the time and place in which earthquakes will occur, but it did not predicted using big data analytics. This journal know that the Standard Deviation to identify next earthquake happening from tons of international geological survey data using data analysis in pandas & matplotlib framework. It's the top-level component of all the ones that you will consider in the following point current location shakes per minute. Other than above mentioned features separate pandas and matplot function implemented to analyze sheer number of earthquakes per day. Final result shows which location suffered from maximum number of shakes and priority of earthquake occurrence location and **INTRODUCTION:**

Earthquakes, natural disasters with the potential for widespread devastation, underscore the critical need for effective prediction models to enhance disaster preparedness and response. The inherent unpredictability of seismic activity makes earthquake prediction a challenging yet essential field of research. In this project, we embark on the development of an earthquake prediction model using the versatile programming language Python. The primary goal of this endeavor is to leverage machine learning techniques to analyze historical seismic data, identify patterns, and formulate a model capable of providing valuable insights into potential earthquake occurrences. While it's crucial to acknowledge the current limitations in predicting exact timings and locations of earthquakes, the model aims to contribute

2: **Design Thinking process**

Empathize:

Understand the stakeholders involved, including scientists, emergency responders, and the ge neral public.
Conduct interviews and surveys to gather insights into the challenges and needs related to ear thquake prediction and preparedness.

Define:

Clearly articulate the problem statement based on the insights gained during the empathize sta ge.
Define the specific goals and objectives of the earthquake prediction model, considering the pr actical requirements and constraints.

Ideate:

Brainstorm potential features and data sources that could enhance the accuracy of the predicti on model.
Encourage diverse perspectives and creative thinking to generate a wide range of ideas for im proving earthquake prediction.

Prototype:

Develop a minimal viable prototype (MVP) of the earthquake prediction model using Python.
Utilize existing libraries and frameworks for data analysis and machine learning to create a bas ic version of the model.
Focus on a proof-of-concept that demonstrates the feasibility of the selected approach.

Test:

Gather feedback from stakeholders, including domain experts, data scientists, and end-users.
Test the prototype against historical seismic data to evaluate its initial performance.
Iterate on the model based on the feedback received, refining features and algorithms as need ed.

3: DEVELOPMENT PHASES

3.1 ABOUT DATASET:

The National Earthquake Information Center (NEIC) determines the location and size of all significant earthquakes that occur worldwide and disseminates this information immediately to national and international agencies, scientists, critical facilities, and the general public. The NEIC compiles and provides to scientists and to the public an extensive seismic database that serves as a foundation for scientific research through the operation of modern digital national and global seismograph networks and cooperative international agreements. The NEIC is the national data center and archive for earthquake information.

DATASET LINK: https://www.kaggle.com/datasets/usgs/earthquake-

database **DATASET USED:**

1. Data Collection:
Obtain a comprehensive seismic dataset from reliable sources, such as earthquake databases (e.g., USGS Earthquake Catalog) or sensor networks.

Include features such as time, location, depth, magnitude, and any other relevant geological information.

Ensure the dataset spans a significant time period to capture various seismic patterns.

2. Data Preprocessing:

Handle missing data and outliers appropriately.

Convert time-related features into a suitable format for analysis.

Normalize numerical features to ensure uniform scaling.

Explore and visualize the dataset to gain insights.

3. Feature Engineering:

Extract meaningful features that might contribute to earthquake prediction.

Consider creating additional features, such as historical seismic activity trends.

Utilize domain knowledge to enhance feature selection.

4. Labeling:

Designate earthquakes as positive instances and non-earthquake events as negatives.

Determine a suitable threshold for classifying seismic events.

5. Machine Learning Model Selection:

Choose a suitable machine learning algorithm for your prediction task (e.g., Random Forest, Support Vector Machines, Neural Networks).

Split the dataset into training and testing sets.

6. Model Training:

Train the selected model on the training dataset.

Optimize hyperparameters for better performance.

Evaluate the model's performance on the testing dataset.

3.2 DATA PREPROCESSING STEPS

1: Data Collection:
Gather earthquake-related data from reliable sources, such as seismic sensors or earthquake databases.

2:Import Libraries:

Import necessary libraries like NumPy, Pandas, and scikit-learn for data manipulation and machine learning tasks. import numpy as np import pandas as pd

from sklearn.model_selection import train_test_split **3:**

Load the Dataset:

Load the earthquake dataset into a Pandas DataFrame.

data = pd.read_csv('earthquake_data.csv') **4: Handle**

Missing Values:

Check for missing values and decide on a strategy to handle them (e.g., removal or imputation).

data = data.dropna() # Example: Remove rows with missing values 5: Feature

Selection:

Select relevant features that contribute to earthquake prediction. Remove unnecessary columns.

features = data[['feature1', 'feature2', 'feature3']] 6:Feature

Scaling:

Normalize or standardize numerical features to ensure they are on a similar scale.

from sklearn.preprocessing import StandardScaler scaler = StandardScaler()

scaled_features = scaler.fit_transform(features) 7: Split

the Data:

Split the dataset into training and testing sets.

X_train, X_test, y_train, y_test = train_test_split(scaled_features, labels, test_size=0.2, random_state=42)

8:Handle Categorical Variables (if any):

If your dataset includes categorical variables, encode them appropriately (e.g., one-hot encoding).

from sklearn.preprocessing import OneHotEncoder Encoder

= One Hot Encoder()

encoded_categories = encoder.fit_transform(categories).toarray() 9:

Data Augmentation (Optional):

Consider augmenting your dataset by creating variations of existing data to improve model generalization.

10:Save Processed Data:

Save the preprocessed data for future use. processed_data.to_csv('preprocessed_data.csv',

index=False)

Remember that these steps can vary depending on the specifics of your dataset and the machine learning algorithm you plan to use for earthquake prediction.

MODEL TRAINING:

your data and explored the features, the next step is to train your earthquake prediction model. Here's a simple example using a Random Forest Classifier. Keep in mind that the choice of the model depends on your specific requirements and the nature of your data.

from sklearn.ensemble import RandomForestClassifier from

sklearn.metrics import accuracy_score, classification_report

Assuming you have X_train, X_test, y_train, y_test from the preprocessing step

Initialize the Random Forest Classifier

model = Random ForestClassifier(n_estimators=100, random_state=42)

Train the model

Model .fit(X_train, y_train)

Make predictions on the test set y_pred

= model.predict(X_test)

Evaluate the model

accuracy = accuracy_score(y_test, y_pred) print(f"Accuracy:

{accuracy}")

Display classification report print("Classification

Report:")

print(classification_report(y_test, y_pred)) In

this example:

We use a RandomForestClassifier from scikit-learn, which is an ensemble learning method.

n_ estimators is the number of trees in the forest. You can adjust it based on experimentation and cross-validation.

Random _state ensures reproducibility.

After training the model, you make predictions on the test set and evaluate its performance using accuracy or other relevant metrics. The classification report provides more detailed information, including precision, recall, and F1-score for each class.

Remember to fine-tune hyperparameters, cross-validate your model, and potentially try other algorithms based on the characteristics of your data.

EVALUATION STEPS:

1: Model Selection:

Choose a suitable machine learning algorithm for earthquake prediction (e.g., Random Forest, SVM, Neural Networks).

Implement the model using Python libraries such as scikit-learn or TensorFlow.

2: Model Training:

Train the model using the training dataset.

Adjust hyperparameters to optimize performance.

3: Model Evaluation:

Use the testing dataset to evaluate the model's performance.

Common evaluation metrics include accuracy, precision, recall, F1 score, and ROC-AUC.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score

# Assuming y_true and y_pred are the true and 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_prob)  # if your model predicts probabilities

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}") print(f"Recall:

{recall}") print(f"F1 Score: {f1}")
```

print(f"ROC-AUC Score: {roc_auc}")

Hyperparameter Tuning:

Fine-tune model hyperparameters using techniques like grid search or random search.

Cross-Validation:

Perform cross-validation to assess the model's stability and generalization.

```
from sklearn.model_selection import cross_val_score

# Assuming clf is your model
cv_scores = cross_val_score(clf, X_train, y_train, cv=5)  # 5-fold cross-validation

print(f"Cross-Validation Scores: {cv_scores}") print(f"Mean
CV Score: {cv_scores.mean()}")
```

5: Visualization:

Create visualizations to understand the model's predictions and performance.

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, roc_curve

# Confusion Matrix
cm = confusion_matrix(y_true, y_pred) print(f"Confusion
Matrix:\n{cm}")
```

```
# ROC Curve
fpr, tpr, _ = roc_curve(true, y_pred_prob)
plt.plot(for, tpr, label='ROC Curve')
plt.ylabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend() plt.show()
```

4: Deployment (Optional):

If satisfied with the model's performance, deploy it for real-time predictions.

LINK FOR PROJECT EXPLANATION

https://drive.google.com/file/d/163FSMnc3BRPhbKnc57_4ZTcIMLpAkKlL/view?usp=drivesdk