**EARTH QUAKE PREDICTION MODEL USING PYTHON**

**PROBLEM DEFINITION:-**

**1. Introduction:**

**Earthquakes pose significant threats to human life and infrastructure. Early prediction and warning systems can mitigate these risks by providing timely information to residents and authorities. This project aims to develop a machine learning-based earthquake prediction model using Python.**

**2. Objectives:**

**The main objectives of this project are:**

* **To collect and preprocess seismic data from reliable sources.**
* **To perform feature engineering and extract relevant features for earthquake prediction.**
* **To implement a machine learning model, specifically Support Vector Machines (SVM), for earthquake prediction.**
* **To evaluate the model's performance using appropriate metrics.**
* **To create a user-friendly interface for real-time prediction.**

**DESIGN THINKING:-**

**Step 1: Choose a Suitable Kaggle Dataset**

**Visit Kaggle and search for earthquake datasets. One suitable dataset might be the "Significant Earthquake Database" available at: [USGS Earthquake Database](https://www.kaggle.com/usgs/earthquake-database)**

**Step 2: Feature Exploration**

**Load the dataset into a Pandas DataFrame and analyze the distribution, correlations, and characteristics of key features:**

**python**

**import pandas as pd**

**# Load the dataset**

**df = pd.read\_csv("path/to/your/dataset.csv")**

**# Explore the dataset**

**print(df.head()) # Display the first few rows**

**print(df.describe()) # Statistical summary**

**print(df.corr()) # Correlation matrix**

**Step 3: Visualization – World Map**

**Create a world map visualization to display earthquake frequency distribution. You can use libraries like `folium` for this:**

**python**

**import folium**

**# Create a base map**

**m = folium.Map(location=[0, 0], zoom\_start=2)**

**# Add markers for each earthquake**

**for index, row in df.iterrows():**

**folium.CircleMarker(**

**location=[row['Latitude'], row['Longitude']],**

**radius=5,**

**color='red',**

**fill=True,**

**fill\_color='red',**

**fill\_opacity=0.7,**

**popup=f"Magnitude: {row['Magnitude']}"**

**).add\_to(m)**

**# Display the map**

**m.save('earthquake\_map.html')**

**Open the generated `earthquake\_map.html` file in a web browser to see the map.**

**Step 4: Data Splitting**

**Split the dataset into a training set and a test set. You can use scikit-learn for this:**

**python**

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

**# Define features and target**

**X = df[['Latitude', 'Longitude', 'Depth']]**

**y = df['Magnitude']**

**# Split the data**

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

**Step 5: Model Development - Neural Network**

**Build a neural network model for earthquake magnitude prediction. You can use TensorFlow and Keras:**

**python**

**import tensorflow as tf**

**from tensorflow import keras**

**from sklearn.preprocessing import StandardScaler**

**# Standardize features**

**scaler = StandardScaler()**

**X\_train\_scaled = scaler.fit\_transform(X\_train)**

**X\_test\_scaled = scaler.transform(X\_test)**

**# Build the model**

**model = keras.Sequential([**

**keras.layers.Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)),**

**keras.layers.Dense(1) # Output layer for regression**

**])**

**# Compile the model**

**model.compile(optimizer='adam', loss='mean\_squared\_error')**

**# Train the model**

**model.fit(X\_train\_scaled, y\_train, epochs=10, batch\_size=32, validation\_split=0.2)**

**Step 6: Training and Evaluation**

**Train the model on the training set and evaluate its performance on the test set:**

**python**

**# Evaluate the model**

**test\_loss = model.evaluate(X\_test\_scaled, y\_test)**

**print(f"Mean Squared Error on Test Set: {test\_loss}")**

**Adjust the model architecture, hyperparameters, and scaling methods based on the performance results.**

**This is a basic guide, and you might need to customize it based on your specific dataset and requirements.**