**In this part you will begin building your project by loading and preprocessing the dataset.**

**Begin building the earthquake prediction model by loading and preprocessing the dataset.**

## Step 1: Load the Dataset

You'll first need to obtain a dataset containing earthquake data. This could be in a CSV, JSON, or any other common data format. For this example, I'll assume you have a CSV file named earthquake_data.csv.

```
import pandas as pd

# Load the dataset

data = pd.read_csv('earthquake_data.csv')
```

## Step 2: Explore and Understand the Data

Before preprocessing, it's important to understand the structure and content of your dataset. You can do this by examining the first few rows, checking for missing values, and understanding the features.

```
# Display the first few rows of the dataset

print(data.head())

# Check for missing values

print(data.isnull().sum())
```

# Explore the columns and their data types

print(data.dtypes)

## Step 3: Preprocessing

Based on the nature of your dataset, you might need to perform various preprocessing steps. Here are some common preprocessing techniques:

### 3.1 Handling Missing Values

If there are missing values, you'll need to decide how to handle them. Options include filling with a specific value, using interpolation, or dropping rows or columns with missing data.

# Example: Fill missing values with the mean of the column

data.fillna(data.mean(), inplace=True)

### 3.2 Feature Selection/Engineering

You might want to select relevant features or engineer new features based on domain knowledge.

# Example: Selecting specific features

selected_features = data[['feature1', 'feature2']]

# Example: Engineering a new feature (e.g., combining existing features)

data['new_feature'] = data['feature3'] * data['feature4']

### 3.3 Scaling/Normalization

Depending on the algorithm you plan to use, it may be necessary to scale or normalize the features.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaled_features = scaler.fit_transform(data[['feature1', 'feature2']])
```

### 3.4 Train-Test Split

Split the dataset into training and testing sets.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(scaled_features, data['target'], test_size=0.2, random_state=42)
```

## Step 4: Building the Model

Once the data is preprocessed, you can start building your earthquake prediction model using an appropriate machine learning or deep learning algorithm.

```
import pandas as pd

from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

# Generate a simple example dataset (replace this with your real dataset loading)

data = pd.DataFrame({

    'Magnitude': [5.0, 6.0, 7.0, 5.5, 6.5],

    'Depth': [10, 15, 8, 12, 18],

    'Distance_From_Fault': [30, 40, 25, 35, 50],

    'Earthquake_Probability': [0, 0, 1, 0, 1]

# Step 1: Explore and Understand the Data

print(data.head())

# Step 2: Preprocessing

X = data.drop('Earthquake_Probability', axis=1)

y = data['Earthquake_Probability']

# Step 3: Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Building the Model

model = RandomForestClassifier(random_state=42)
```

```
model.fit(X_train, y_train)
```

**# Step 5: Evaluate the Model**

```
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy*100:.2f}%')
```

**Output:**

| | Magnitude | Depth | Distance_From_Fault | Earthquake_Probability |
|---|---|---|---|---|
| 0 | 5.0 | 10 | 30 | 0 |
| 1 | 6.0 | 15 | 40 | 0 |
| 2 | 7.0 | 8 | 25 | 1 |
| 3 | 5.5 | 12 | 35 | 0 |
| 4 | 6.5 | 18 | 50 | 1 |

**Accuracy: 100.00%**