Phase 4 Development - Part 2: Visualizing Data and Splitting into Training and Testing Sets

In this phase, you will create visualizations to gain insights from the earthquake dataset and split it into training and testing sets for further model development. Visualizing the data on a world map can help you understand the geographic distribution of earthquakes. Below are the steps to follow:

1. **Data Visualization on a World Map**:

To visualize the geographic distribution of earthquakes, you can use libraries like Folium or Plotly. Here's how you can create a basic world map with earthquake data points:

Python Code

```
 import folium
  # Create a base map
  world_map = folium.Map(location=[0, 0], zoom_start=2)
  # Add earthquake data points to the map
  for index, row in data.iterrows():
     folium.CircleMarker(location=[row['latitude'], row['longitude']],
                    radius=5,
                    color='red',
                    fill=True,
                    fill_color='red').add_to(world_map)
  world_map.save('earthquake_map.html')
```

This code will create an interactive map where you can click on data points to view more information about each earthquake.

2. **Data Splitting:**

To train and test your earthquake prediction model, you should split your dataset into a training set and a testing set. You've already imported the necessary libraries and loaded the data in previous phases, so here's how you can perform the split:

Python Code

```
from sklearn.model_selection import train_test_split
  # Define features (X) and target variable (y)
  X = data[['latitude', 'longitude', 'depth', 'magnitude', 'seismic_data']]
  y = data['target_variable']  # Replace 'target_variable' with your actual target variable

  # Split the data into training and testing sets (e.g., 80% training, 20% testing)
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

This code will split your dataset into two sets: `X_train` and `y_train` for training, and `X_test` and `y_test` for testing. Adjust the `test_size` parameter to set the desired proportion of your data for testing.

3. **Visualize Data Distribution**:

After splitting the data, you can create visualizations to understand the distribution of earthquakes within the training and testing sets. This can help identify any potential biases in the data split.

For example, you can use matplotlib to create histograms or scatter plots to visualize the distributions of features like magnitude, depth, and seismic data within both the training and testing sets.

4. **Save Split Data**:

It's a good practice to save the split data for future reference, especially if you plan to iterate on your model. You can use libraries like NumPy to save the training and testing sets as arrays or Pandas DataFrames.

Python Code

```
np.save('X_train.npy', X_train)

np.save('X_test.npy', X_test)

np.save('y_train.npy', y_train)

np.save('y_test.npy', y_test)
```

With the data visualized on a world map and the dataset split into training and testing sets, you're ready to proceed with model development and evaluation in the following phases. The visualizations will help you better understand the geographic distribution of earthquakes and the training/testing data distribution.

Phase 4 Development - Part 3: Feature Engineering and Model Building

Now that you have visualized the data on a world map and split it into training and testing sets, you can proceed to feature engineering and building your earthquake prediction model. In this part, you'll focus on feature selection, engineering, and model development.

1. **Feature Selection and Engineering**:

Domain Knowledge: Collaborate with domain experts to select the most relevant features. For earthquake prediction, features may include latitude, longitude, depth, magnitude, and various seismic measurements.

Additional Features: Consider engineering additional features, such as distance to fault lines, historical earthquake frequency in the region, or environmental factors that may influence seismic activity.

Normalization: Normalize or scale the features to ensure they have similar scales. Standardization or Min-Max scaling is commonly used.

2. **Model Selection:**

Choose a suitable machine learning model for your earthquake prediction task. This may involve trying different algorithms and evaluating their performance. Common choices include Random Forest, Support Vector Machines (SVM), or deep learning models.

3. **Model Development:** Initialize and train your chosen model using the training dataset.

Python Code

```
   from sklearn.ensemble import RandomForestClassifier  # Replace with your model choice

   model = RandomForestClassifier(n_estimators=100, random_state=42)

   model.fit(X_train, y_train)
```

4. **Model Evaluation:**

   - Use the testing dataset to evaluate the model's performance. Calculate metrics like accuracy, precision, recall, F1-score, and confusion matrix to assess the model's accuracy and reliability.

Python Code

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

   predictions = model.predict(X_test)

   accuracy = accuracy_score(y_test, predictions)

   precision = precision_score(y_test, predictions)

   recall = recall_score(y_test, predictions)

   f1 = f1_score(y_test, predictions)

   conf_matrix = confusion_matrix(y_test, predictions)
```

5. **Hyperparameter Tuning:**

   - Experiment with different hyperparameters to optimize the model's performance. Grid search or random search can help find the best combination of hyperparameters.

6. **Cross-Validation (Optional):**

- Implement cross-validation to ensure the model's robustness. This helps evaluate how well the model generalizes to unseen data.

7. **Visualization and Interpretation:**

- Create visualizations to interpret the model's predictions and results. You can plot ROC curves, feature importances, and maps showing predicted earthquake probabilities.

8. **Model Saving:**

- Once you are satisfied with the model's performance, save it for future use. You can use libraries like joblib or pickle to save the model.

Python Code

```
import joblib
joblib.dump(model, 'earthquake_prediction_model.pkl')
```

9. **Documentation**:

- Maintain detailed documentation of the model, including the chosen features, hyperparameters, and any domain-specific insights. This documentation is crucial for transparency and reproducibility.

10. **Iteration**: Be prepared to iterate on your model. As you gather more data and insights, you may need to revisit your feature engineering, model choice, and hyperparameter settings to improve the model's performance.

11. **Communication**:

- Regularly communicate your model's results and findings with stakeholders, experts, and the wider community. Transparency is essential, especially in earthquake prediction, where public safety is at stake.

This part of the development process focuses on the core aspects of creating the earthquake prediction model. Once the model is built and evaluated, you can proceed to deployment and continuous improvement, as discussed in earlier phases.