EARTHQUAKE PREDICTION MODEL USING PYTHON

Phase 4: Development Part 2

NAME:DHILIP KUMAR K

NM ID :513521104011

COLLEGE : AMCET

Introduction:

In this phase, we aim to visualize the earthquake data on a world map to gain geographic insights, split the data into training and testing sets for model evaluation, select a suitable machine learning algorithm, train the model using the training data, and evaluate its performance using appropriate metrics.

Procedure:

1. Visualizing Data on a World Map:

Utilize geospatial visualization libraries like `folium` to plot earthquake data points on a world map. Use colors, sizes, or heat maps to represent earthquake magnitudes. This visualization provides a global overview of earthquake occurrences.

Program:

```
import folium
```

Assuming 'data' is your Data Frame containing earthquake data with columns 'latitude', 'longitude', and 'magnitude'

```
import pandas as pd

import folium


# Load your dataset (replace "data.csv" with your actual data file)

data = pd.read_csv("data.csv")


# Assuming 'data' is your DataFrame containing earthquake data with columns 'latitude', 'longitude', and 'magnitude'

map = folium.Map(location=[0, 0], zoom_start=2)


for index, row in data.iterrows():

folium.CircleMarker(

        location=[row['latitude'], row['longitude']],

        radius=row['magnitude'],  # Adjust the radius based on magnitude

        color='crimson',

        fill=True,

fill_color='crimson',

fill_opacity=0.6

    ).add_to(map)


map.save("earthquake_map.html")
```

Output:

This code will save an interactive HTML map showing earthquake occurrences.

2. Splitting Data into Training and Testing Sets:

Use `train_test_split` from scikit-learn to split the data into training and testing sets for model evaluation.

Program:

```
import pandas as pd

from sklearn.model_selection import train_test_split


# Load your dataset (replace "data.csv" with your actual data file)

data = pd.read_csv("data.csv")
```

```
# Define features and target variable
features = ['latitude', 'longitude', 'depth', 'hour']  # Adjust features as needed
target = 'magnitude'

# Extract features (X) and target variable (y)
X = data[features]
y = data[target]

# 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)

# Now, X_train, X_test, y_train, and y_test are available for further processing
# For example, you can train your machine learning model using X_train and y_train
# and evaluate its performance using X_test and y_test
```

Output:

```
       Latitude  Longitude  Depth
18765    14.944    -61.274  156.0
21035   -14.438    -75.966   24.0
18334    38.340     20.420   15.0
16776    42.525    145.021   28.6
9152    -15.864   -172.067   27.9
...         ...        ...    ...
11964    27.995    140.700   28.2
21575   -10.682    166.381   10.0
5390     -6.847    129.634  131.0
860      -5.469    153.269   30.0
15795   -60.657    -25.843   10.0

[18729 rows x 3 columns]        Latitude  Longitude  Depth
3848      3.166     99.015  180.0
14008    43.679    -29.020   10.0
16258     1.142     98.911   77.6
18090    38.649     15.390  212.0
15192    38.457     31.351   10.0
...         ...        ...    ...
15058   -17.286   -175.176  291.8
18377    -6.742    154.889   10.0
87       36.405     70.724  207.8
10309   -21.953    174.818   10.2
14530   -30.738    -71.993   33.0


[4683 rows x 3 columns] 18765    7.4
21035    6.9
18334    5.7
16776    5.5
9152     5.7
        ...
11964    5.8
21575    5.8
5390     5.8
860      5.7
15795    5.8
Name: Magnitude, Length: 18729, dtype: float64 3848     5.6
14008    5.5
16258    5.5
18090    5.8
15192    6.0
        ...
15058    6.3
18377    5.6
87       7.4
10309    5.6
14530    6.0
Name: Magnitude, Length: 4683, dtype: float64
```

3. Selecting a Machine Learning Algorithm, Training, and Evaluation:

Choose a machine learning algorithm (e.g., Random Forest, Gradient Boosting) and train the model using the training data. Evaluate the model using appropriate metrics (e.g., Mean Squared Error for regression tasks).

Program:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean_squared_error


# Load your dataset (replace "data.csv" with your actual data file)

data = pd.read_csv("data.csv")


# Define features and target variable

features = ['latitude', 'longitude', 'depth', 'hour']  # Adjust features as needed

target = 'magnitude'


# Extract features (X) and target variable (y)

X = data[features]

y = data[target]
```

```
# 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)



# Create a Random Forest regression model

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



# Train the model using the training data

model.fit(X_train, y_train)



# Make predictions on the test set

predictions = model.predict(X_test)

# Calculate Mean Squared Error

mse = mean_squared_error(y_test, predictions)

print("Mean Squared Error:", mse)
```

Output:

```
Mean Squared Error: 0.19981056368252934
```

Conclusion:

In this phase, we successfully visualized earthquake data on a world map, split it into training and testing sets, selected a Random Forest regression model, trained the model, and evaluated its performance using Mean Squared Error. Visualizing the data geographically provides valuable insights, and the model's evaluation metrics help assess its accuracy in predicting earthquake magnitudes.