Earthquake magnitude prediction through supervised learning and decision tree regressor model

Sugam Kafle, Justin Mariki
University of North Texas

Abstract

Earthquakes are such random and unpredictable natural phenomenon that most machine learning algorithms struggle to predict when they will occur. However, the use of ML in solid Earth geosciences is growing rapidly (Bergen, Karianne J) We used a large dataset containing historical earthquake data (1965–2016), and Supervised Learning with a decision tree regressor to train a model with geospatial coordinates (longitude and latitude) as the input dataset, and the magnitude as the output dataset.

We created a decision tree regressor model to train our dataset to predict the magnitude of an earthquake corresponding to a given longitude and latitude coordinates. This was done by splitting the data into training and testing sets. We then fed our model two sets of longitude and latitude to make prediction. For the sake of measuring the reliability of our model, we scaled the data to a range of 0 to 1 ; fit the model with scaled data , and calculated the accuracy.

Datasets and Libraries

The following table is a sample dataset was extracted from Kaggle: https://www.kaggle.com/datasets/usgs/earthquake-database

	Latitude	Longitude	Magnitude
17700	-60.957	-21.606	7.4
17701	-4.883	152.343	5.6
17702	-19.926	-178.178	7.2
17703	13.841	145.287	5.8
17704	28.164	-112.117	6.6
17705	-22.855	-66.172	5.5
17706	-56.552	-24.954	5.7
17707	6.635	-82.337	6.1
17708	52.415	173.613	5.5
17709	36.311	23.212	6.7

Table 1.1 Sample Dataset

In order to manipulate our data, two python libraries were imported:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
```

Methodology & Code

This section details how the dataset was prepared, and the use of a decision tree regressor training model. Full code available on GitHub through the following link: https://github.com/sugamkafle/DiscoverAI/tree/main/DiscoverAI_Final_Project

1. PREPARING INPUT AND OUTPUT DATASETS

```
#Input data set #Conversion to arrays
X = table.drop(columns= ['Magnitude'])
X = np.array(X)
X
```

```
array([[ -60.957 ,  -21.606 ],
       [  -4.883 ,  152.343 ],
       [ -19.926 , -178.178 ],
       ...,
       [  36.9179,  140.4262],
       [  -9.0283,  118.6639],
       [  37.3973,  141.4103]])
```

```
#Output data set #Conversion to arrays
y = table['Magnitude']
y = np.array(y)
y
```

```
array([7.4, 5.6, 7.2, ..., 5.9, 6.3, 5.5])
```

2. PREDICTING WITH THE DECISION TREE REGRESSOR

```
# Importing the DecisionTreeRegressor package from sklearn
from sklearn.tree import DecisionTreeRegressor

# Define the decision tree regressor model
model = DecisionTreeRegressor()

# Fit the model with X and y
model.fit(X, y)

# With the fitted model, predict two values:
#longitude and latitude of "Nepal" and "UNT, Denton"
predictions = model.predict([[28.394857,84.124008],
                             [33.207489, -97.152588]])
print(predictions)
```

```
[7.8 5.7]
```

3. SPLITTING DATA INTO TRAINING AND TESTING SET

```
from sklearn.model_selection import train_test_split
#Train Test Split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

4. SCALING TRAINING SET AND TEST SET USING MINMAXSCALER

```
#Scaling training data
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
model.fit(X_train_scaled, y_train)

#Scaling testing data
X_test_scaled = scaler.fit_transform(X_test)
y_pred = model.predict(X_test_scaled)
```

5. CALCULATING ACCURACY

```
#calculating accuracy
from sklearn.model_selection import cross_val_score
regressor = DecisionTreeRegressor(random_state=1)
regressor.fit(X_train_scaled, y_train)

score = regressor.score(X_test_scaled, y_pred)
print(y_pred,score)
```

```
[5.9 5.7 5.8 ... 6.  5.5 6. ] 0.8082587979123442
```

6. MEAN SQUARED ERROR

```
#Calculating mean sqaured error
from sklearn.metrics import mean_squared_error
error_score = mean_squared_error(y_test, y_pred)
error_score
```

```
0.4272570428696413
```

7. Dataset entropy score

```
# Calculating Entropy of the dataset
import scipy
k = np.unique(y).size
maxE = np.log2(k)

p_data, p_counts = np.unique(y, return_counts=True)
# counts occurrence of each value

entropy = scipy.stats.entropy(p_counts)

# normalize the value to be between 0 and 1.
normalizedE = entropy/maxE
normalizedE
```

```
0.4126038347584009
```

Findings and Conclusion

Predicting earthquakes is very difficult, complex, and involves large amount of data processing. Nevertheless, our model attempts to predict the magnitude of an earthquake for a given location rather than predict the occurrence of an earthquake. The first result we want to focus on is the entropy score of the dataset that we utilized. Entropy is the measure of randomness of data. ***Our findings show that our dataset scored approximately 0.413***, This suggests that our dataset might not be random enough, however, this can also be attributed to the use of such a large dataset that contains 51 years' worth of data. Because of such a long time period, our dataset is likely a considerable number of earthquake occurrences in the same vicinity. For instance, the Honshu region in Japan is one of the most earthquake prone region of the world. We see earthquake data for Honshu several times in our dataset.

Another highlighted finding is the prediction accuracy of our trained model. ***The model scored roughly 0.808 which translates to 80.8% accuracy***. Further, we calculated the mean squared error which is the measure of how close a regression line is to a set of points. ***This error was calculated to be roughly 0.43*** which suggests that there is a fair amount of correlation between our testing dataset and the predictions made by the model.

While making predictions, we used two sets of geospatial coordinates for the country Nepal, and the location of the University of North Texas in Denton, Texas. ***The predicted magnitude were 7.8 and 5.7 respectively***. The 7.8 magnitude prediction was very favorable for our prediction since Nepal had seen a 7.8 magnitude earthquake in 2015. Meanwhile, the geography of Denton is such that earthquakes are very uncommon, and a low 5.7 magnitude prediction is also favorable for our model.

Here, we re-emphasize that study of earthquake prediction is still in its nascent stage despite the rigorous and historical study of earthquakes. However, if we were to assume an occurrence of an earthquake in any location, our model can, with fair certainty, predict the magnitude of the assumed earthquake.

Resources and Acknowledgements

The following resources have provided great assistance in our research:
1. **Kaggle:** https://www.kaggle.com/datasets/usgs/earthquake-database
2. **Machine learning for data-driven discovery in solid Earth geoscience** https://www.science.org/doi/full/10.1126/science.aau0323

We would also like to acknowledge the following people from Discover AI for their guidance:
- **Dr. Hannah Lee—Instructor**
- **Hannah Helgelsen—Teaching Assistant**
- **Sanjay Adhikari—Teaching Assistant**