

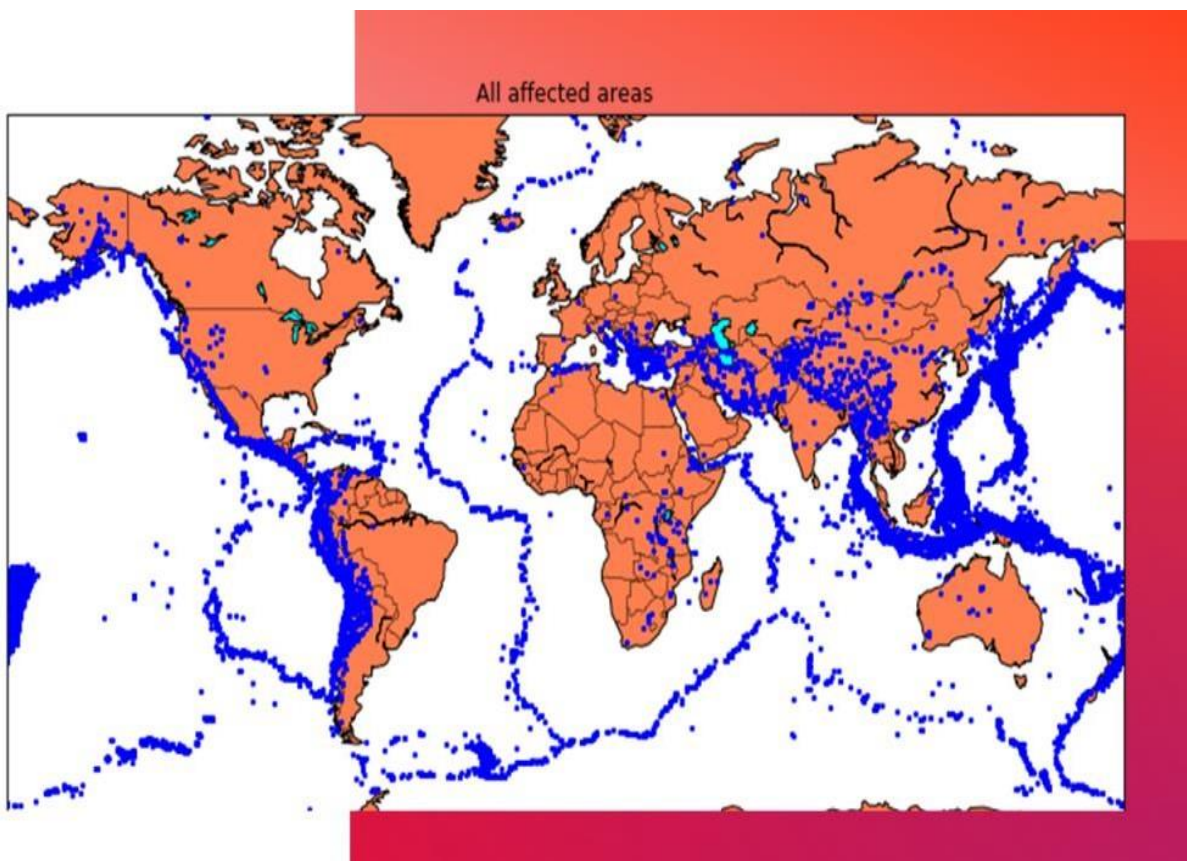
EARTHQUAKE PREDICTION MODEL USING PYTHON

Phase 3 submission document

Project Title: Earthquake Predictions Model Using Python

Phase 3: Development Part 1

Topic: Start building the earthquake prediction model by loading and pre-processing the dataset.



INTRODUCTION:

The SOCR Earthquake Dataset can be used to build machine learning models to predict earthquakes or to better understand earthquake patterns and characteristics. Here are a few possible ways machine learning models can be used with this dataset:

1. Earthquake prediction: You can use this dataset to build a model that predicts when and where an earthquake might occur based on past earthquake data. You could use techniques such as time series analysis, clustering, or classification to identify patterns in the data and make predictions.
2. Magnitude prediction: You can use this dataset to build a model that predicts the magnitude of an earthquake based on other factors such as location, depth, or the number of seismic stations that recorded the earthquake. You could use regression techniques to build this model.
3. Risk assessment: You can use this dataset to identify areas that are at higher risk of earthquakes based on historical earthquake data. You could use clustering or classification techniques to identify patterns in the data and identify areas with similar characteristics.
4. Anomaly detection: You can use this dataset to detect anomalies or outliers in the data, which could represent earthquakes that are unusual or unexpected. You could use techniques such as clustering or classification to identify patterns in the data and detect anomalies.
5. Data visualization: You can use this dataset to create visualizations of earthquake data, which could help you identify patterns and relationships in the data. You could use techniques such as scatter plots, heat maps, or geographic information systems (GIS) to visualize the data.

These are just a few examples of the many ways that machine learning models can be used with the SOCR Earthquake Dataset. The specific approach you take will depend on your research question and the goals of your analysis. In this project we focus mainly on earthquake prediction and Magnitude prediction.

GIVEN DATA SET:

	Date	Time	Latitude	Longitude	Type	Depth	Depth Err	Depth Std	Magnitude	Magnitude	Magnitude	Magnitude	Azimuth	Horizontal	Horizontal	Root Mean	ID	Source	Location
2		01-02-1955	13:44:18	19.246	145.616 Earthquak	131.5			6 MW								ISCGEM86	ISCGEM	ISCGEM
3		01-04-1955	11:29:49	1.853	127.352 Earthquak	80			5.8 MW								ISCGEM86	ISCGEM	ISCGEM
4		01-05-1955	18:05:58	-20.579	-173.972 Earthquak	20			6.2 MW								ISCGEM86	ISCGEM	ISCGEM
5		01-08-1955	13:49:43	-59.076	-23.557 Earthquak	15			5.8 MW								ISCGEM86	ISCGEM	ISCGEM
6		01-09-1955	13:32:50	11.938	126.427 Earthquak	15			5.8 MW								ISCGEM86	ISCGEM	ISCGEM
7		01-10-1955	13:36:32	-13.405	166.629 Earthquak	35			6.7 MW								ISCGEM86	ISCGEM	ISCGEM
8		01-12-1955	13:32:25	27.357	87.867 Earthquak	20			5.9 MW								ISCGEM86	ISCGEM	ISCGEM
9	01/15/1965	23:17:42	-13.309	166.212 Earthquak	35				6 MW								ISCGEM86	ISCGEM	ISCGEM
10	01/16/1965	11:32:37	-56.452	-27.043 Earthquak	95				6 MW								ISCGEM86	ISCGEM	ISCGEM
11	01/17/1965	10:43:17	-24.553	178.487 Earthquak	565				5.8 MW								ISCGEM86	ISCGEM	ISCGEM
12	01/17/1965	20:57:41	-6.807	108.988 Earthquak	227.9				5.9 MW								ISCGEM86	ISCGEM	ISCGEM
13	01/24/1965	00:11:17	-2.608	125.952 Earthquak	20				8.2 MW								ISCGEM86	ISCGEM	ISCGEM
14	01/29/1965	09:35:30	54.636	161.703 Earthquak	55				5.5 MW								ISCGEM86	ISCGEM	ISCGEM
15		02-01-1955	05:27:06	-18.697	-177.864 Earthquak	482.9			5.6 MW								ISCGEM85	ISCGEM	ISCGEM
16		02-02-1955	15:56:51	37.523	73.251 Earthquak	15			6 MW								ISCGEM85	ISCGEM	ISCGEM
17		02-04-1955	03:25:00	-51.34	139.741 Earthquak	10			6.1 MW								ISCGEM85	ISCGEM	ISCGEM
18		02-04-1955	05:01:22	51.251	178.715 Earthquak	30.3			8.7 MW								OFFICIAL1	OFFICIAL	ISCGEM
19		02-04-1955	05:04:59	51.639	175.055 Earthquak	30			6 MW								ISCGEM85	ISCGEM	ISCGEM
20		02-04-1955	05:37:06	52.528	172.007 Earthquak	25			5.7 MW								ISCGEM85	ISCGEM	ISCGEM
21		02-04-1955	05:39:32	51.626	175.746 Earthquak	25			5.8 MW								ISCGEM85	ISCGEM	ISCGEM
22		02-04-1955	07:11:23	51.037	177.848 Earthquak	25			5.9 MW								ISCGEM85	ISCGEM	ISCGEM

23413 Rows * 21 Columns

NECESSARY STEP TO FOLLOW:

1. Import the Necessary Libraries:

First, you need to import the libraries you'll be using. You can use libraries such as Pandas, NumPy, and scikit-learn for this task.

Program:

```
import pandas as pd
```

```
import numpy as np
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
```

2. Load the Dataset:

For this example, let's assume you have a CSV file with earthquake-related data. You can use Pandas to load the dataset.

Program:

```
# Load the dataset (replace 'your_dataset.csv' with the actual file path)
```

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

3. Explore the Dataset:

It's important to understand your data. Check the first few rows, data types, and summary statistics.

Program:

```
print(data.head())
```

```
print(data.info())
```

```
print(data.describe())
```

4. Data Preprocessing:

Data preprocessing is a crucial step in machine learning. It involves handling missing values, encoding categorical variables, and scaling numerical features. Here's a basic example.

Program:

```
# Handle missing values (replace NaN with mean or other suitable strategies)
```

```
data = data.fillna(data.mean())
```

```
# Split the dataset into features (X) and target (y)

X = data.drop('earthquake_occurred', axis=1) # Replace
'earthquake_occurred' with your target column

y = data['earthquake_occurred']
```

```
# Encode categorical variables if necessary (e.g., using one-hot encoding or
label encoding)
```

```
# Scale the numerical features to have mean=0 and variance=1

scaler = StandardScaler()

X = scaler.fit_transform(X)
```

5. Split the Data into Training and Testing Sets:

It's essential to split your data into training and testing sets to evaluate your model's performance.

Program:

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

Data Visualization:

Now, before we create the earthquake prediction model, let's visualize the data on a world map that shows a clear representation of where the earthquake frequency will be more:

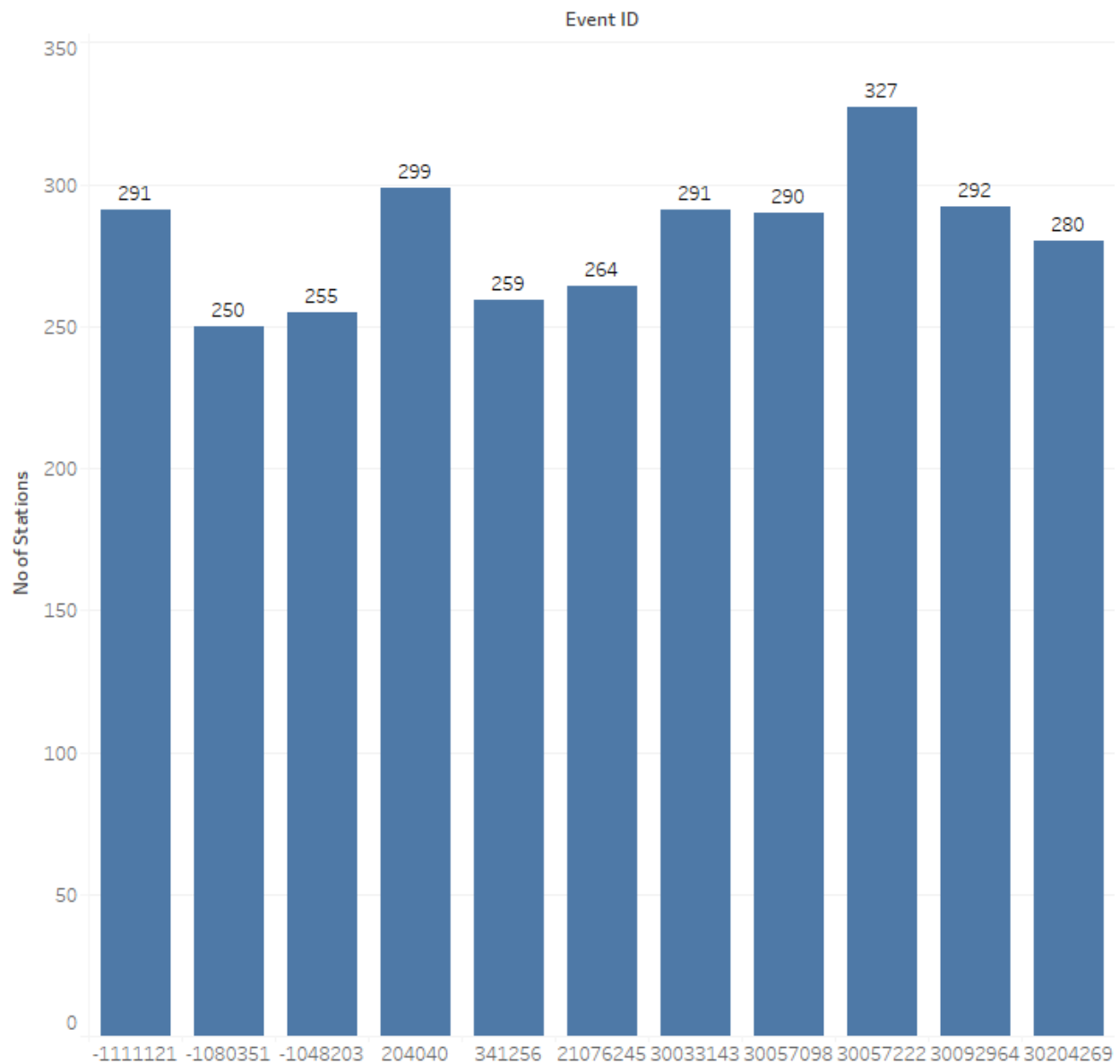
```
from mpl_toolkits.basemap import Basemap
```

```
m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=180,lat_ts=20,resolution='c')
```

```
longitudes = data["Longitude"].tolist()
latitudes = data["Latitude"].tolist()
#m = Basemap(width=12000000,height=9000000,projection='lcc',
             #resolution=None,lat_1=80.,lat_2=55,lat_0=80,lon_0=-107.)
x,y = m(longitudes,latitudes)
```

```
fig = plt.figure(figsize=(12,10))
plt.title("All affected areas")
m.plot(x, y, "o", markersize = 2, color = 'blue')
m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
m.drawmapboundary()
m.drawcountries()
plt.show()
```

Earthquake (identified by Event ID) and the number of stations recording it



Sum of No of Stations for each Event ID. The data is filtered on No of Stations, which includes values greater than or equal to 250.

Figure 1
Earthquake (identified by Event ID) and the number of stations recording it

Earthquake based on its magnitude

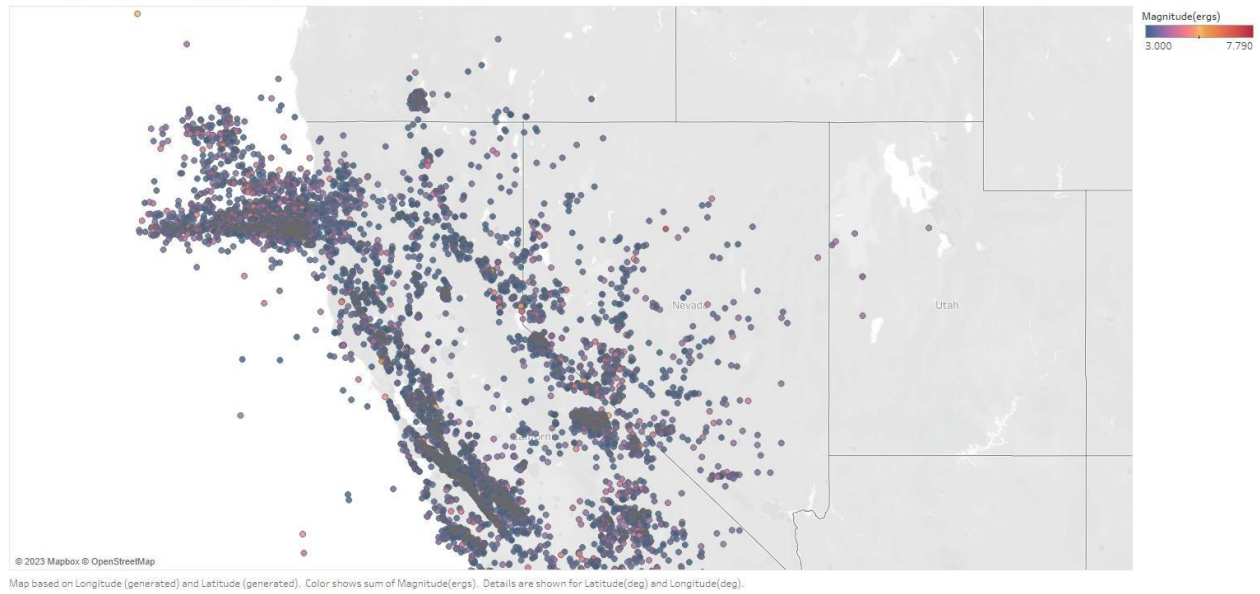


Figure 2
Earthquake based on its magnitude

Earthquake based on its magnitude type

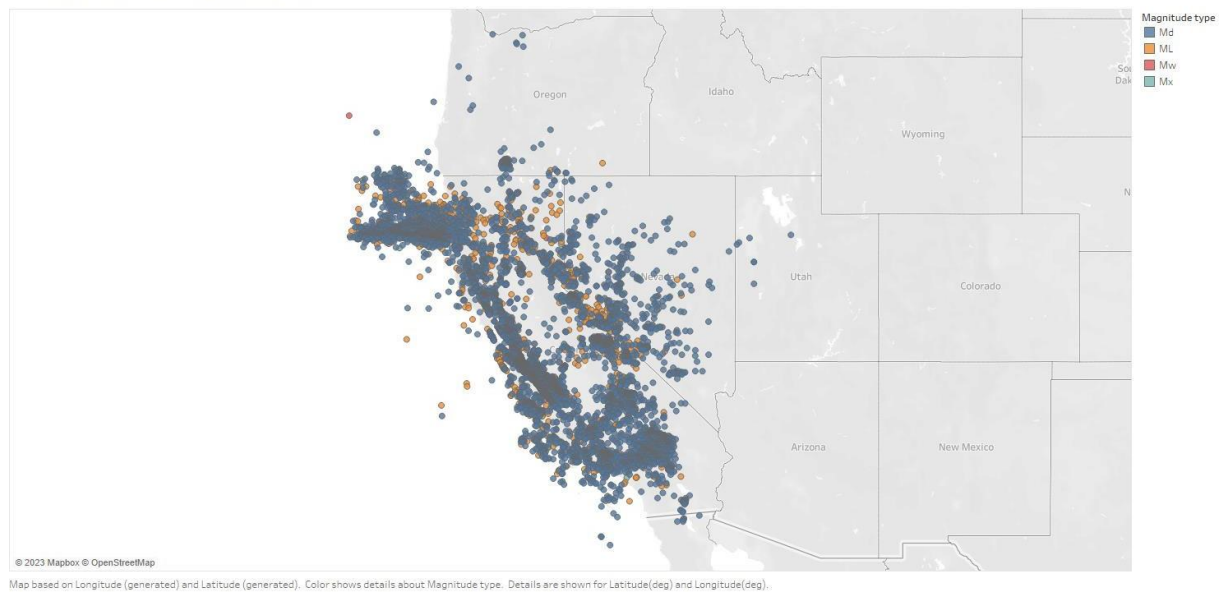
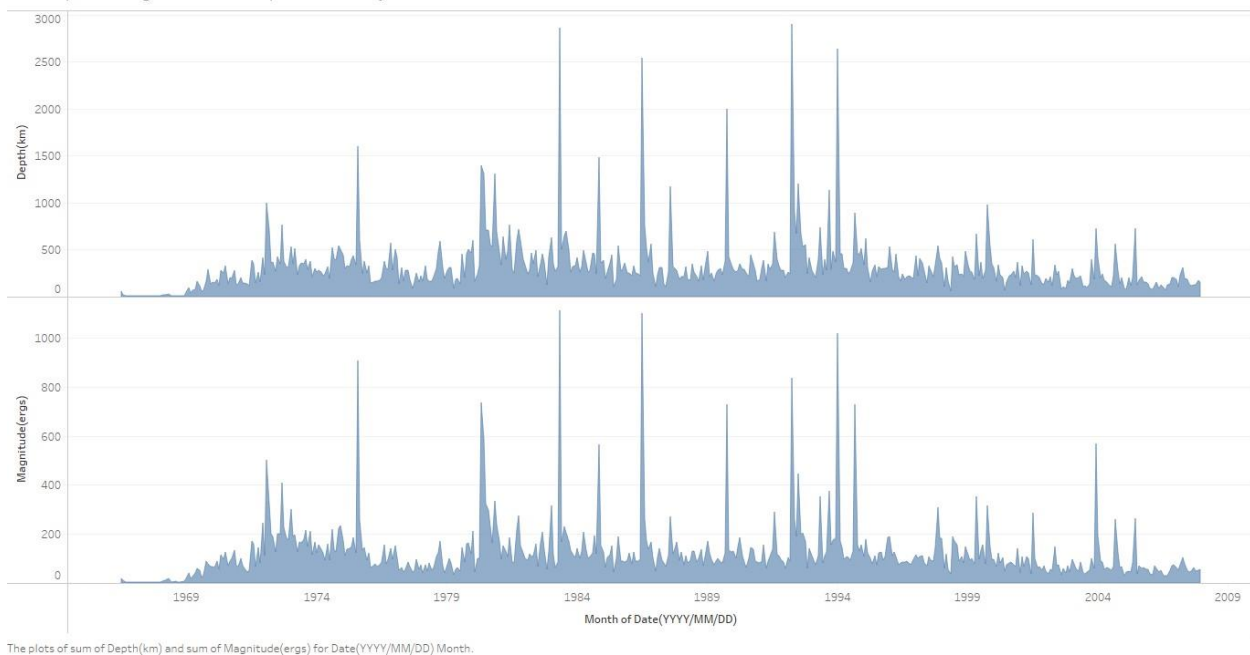


Figure 3
Earthquake based on its magnitude type

Earthquake magnitude and depth over the years



The plots of sum of Depth(km) and sum of Magnitude(ergs) for Date(YYYY/MM/DD) Month.

Figure 4
Earthquake magnitude and depth over the years

Splitting the Dataset:

Now, to create the earthquake prediction model, we need to divide the data into Xs and ys which respectively will be entered into the model as inputs to receive the output from the model.

Here the inputs are Timestamp, Latitude and Longitude and the outputs are Magnitude and Depth. I'm going to split the xs and ys into train and test with validation. The training set contains 80% and the test set contains 20%:

```
X = final_data[['Timestamp', 'Latitude', 'Longitude']]
y = final_data[['Magnitude', 'Depth']]
from sklearn.cross_validation import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

CONCLUSION:

When comparing two models, both the mean squared error (MSE) and R-squared (R^2) score can be used to evaluate the performance of the models.

In general, a model with a lower MSE and a higher R^2 score is considered a better model. This is because the MSE measures the average difference between the predicted and actual values, and a lower MSE indicates that the model is making more accurate predictions. The R^2 score measures the proportion of the variance in the target variable that is explained by the model, and a higher R^2 score indicates that the model is able to explain more of the variability in the target variable.

From the results of this project we can conclude that random forest is the most accurate model for predicting the magnitude of Earthquake compared to all other models used in this project.

However, it's important to keep in mind that the relative importance of MSE and R^2 score may vary depending on the specific problem and the context in which the models are being used. For example, in some cases, minimizing the MSE may be more important than maximizing the R^2 score, or vice versa. It's also possible that one model may perform better on one metric and worse on another, so it's important to consider both metrics together when evaluating the performance of the models.