EARTH QUAKE PREDICTION USING PYTHON

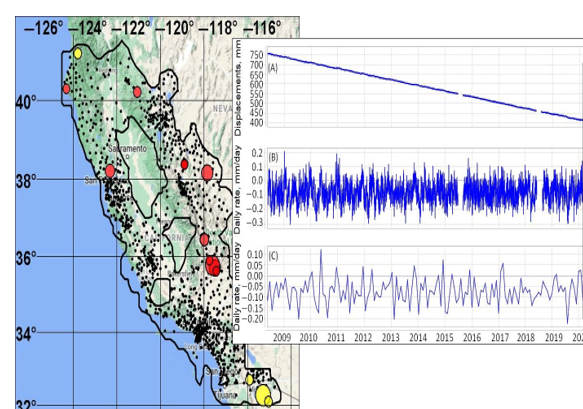
*BATCH MEMBER*

621121104012: GOWRI.K

Phase 3 Submission

***Project Title****: Earth Quake Prediction*

Topic: Earth quake Prediction model by loading and pre-processing the dataset.



*Earth Quake Prediction*

*Abstract:*

An earthquake is shaking of the surface of the Earth, which caused as the result of movable plate boundary interactions Earthquakes are measured using remarks from seismometers with Richter magnitude scale.

Ground rupture, Landslides, Soil liquefaction and Tsunami are the main effects created by earthquakes. Today's earthquake warning systems used to provide regional notification of an earthquake in progress. Many methods have been already developed for predicting the time and place in which earthquakes will occur.

Program:

In [1]:

import pandas as pd

Preprocessing steps:

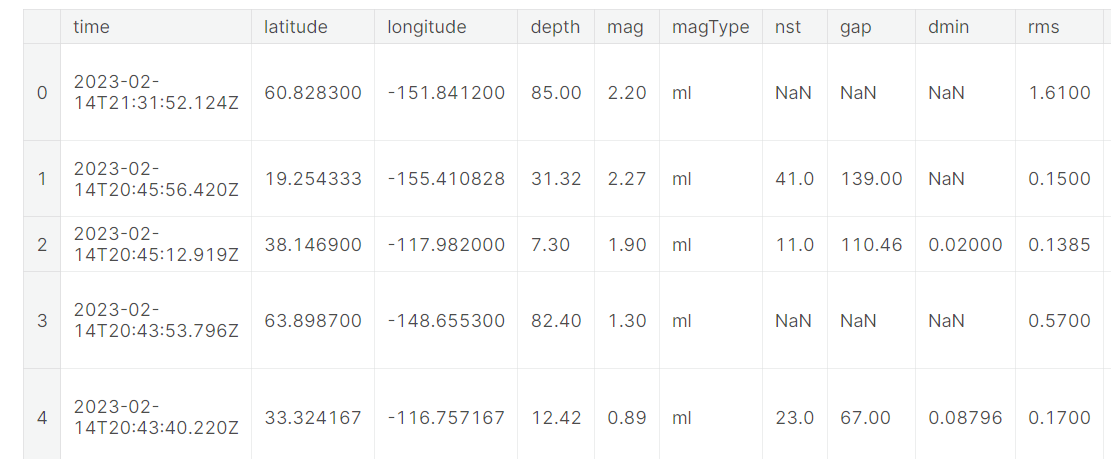
In [2]:

df = pd.read\_csv

In [3]:

df.head()

out [3]:



In [4]:

df.tail()

Out [4]: 

In [5]:

df.shape

Out [5]:

(10153, 22)

In [6]:

df.columns

Out[6]:

Index (['time', 'latitude', 'longitude', 'depth', 'mag', 'magType', 'nst',

'gap', 'dmin', 'rms', 'net', 'id', 'updated', 'place', 'type',

'horizontalError', 'depthError', 'magError', 'magNst', 'status',

'locationSource', 'magSource'],

dtype='object')

In[7]:

df.dtypes

Out [7]:

time object

latitude float64

longitude float64

depth float64

mag float64

magType object

nst float64

gap float64

dmin float64

rms float64

In [8]:

df['time'] = pd.to\_datetime(df['time'])

In [9]:

df.dtypes

Out [9]:

time datetime64[ns, UTC]

latitude float64

longitude float64

depth float64

mag float64

magType object

nst float64

gap float64

dmin float64

rms float64

Data Visualization:

In [10]:

import seaborn as sns

import matplotlib.pyplot as plt

# Set style for all visualizations

sns.set\_style("darkgrid")

# Scatter plot to show relationship between magnitude and depth

sns.scatterplot(data=df, x="mag", y="depth")

# Bar plot to show distribution of magnitudes

sns.histplot(data=df, x="mag")

# Box plot to show distribution of magnitudes by type

sns.boxplot(data=df, x="magType", y="mag")

# Heatmap to show correlation between variables

corr = df.corr()

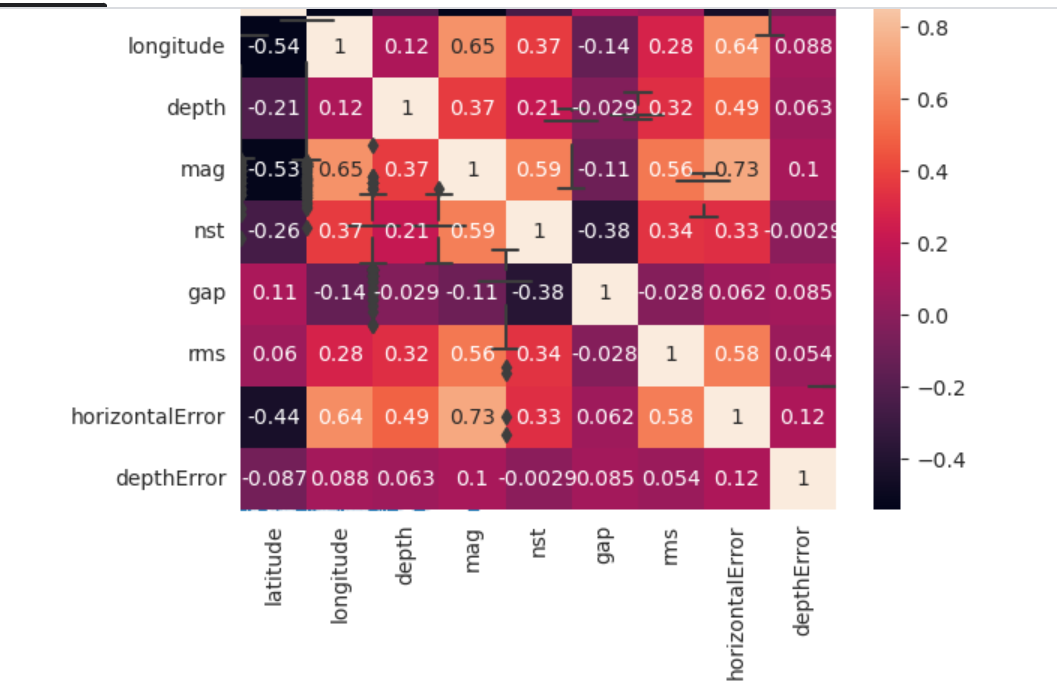
sns.heatmap(corr, annot=True)

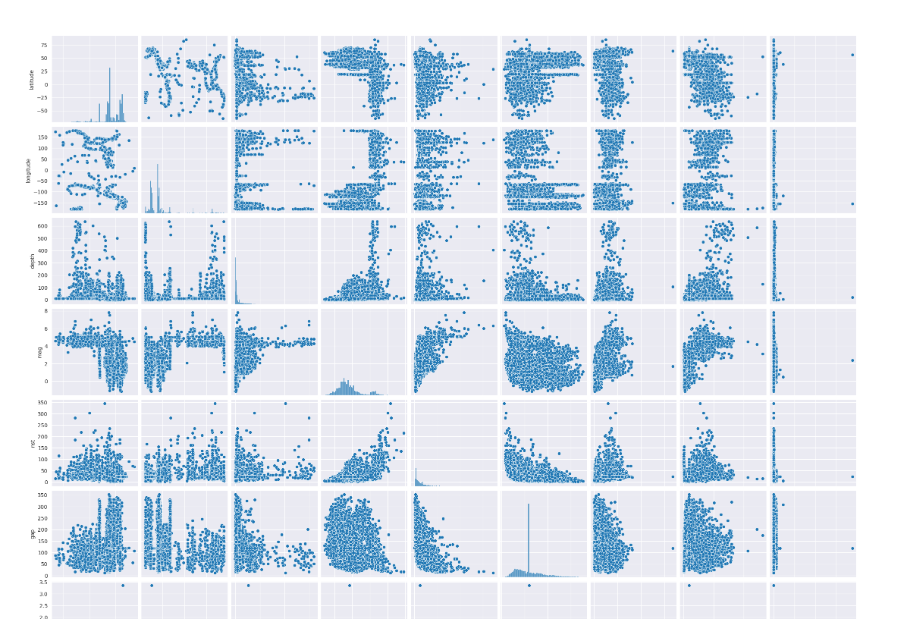
# Pairplot to show scatterplots of all possible variable combinations

sns.pairplot(df)

# Show all visualizations

plt.show()





*Statistical Analysis to test hypothesis.*

In [11]:

from scipy.stats import ttest\_ind

# creating a new 'region' column by extracting region from 'place' column

df['region'] = df['place'].str.extract(',\s(.\*$)')

# grouping by 'region' and calculating mean of 'mag' column for each group

mean\_mag\_by\_region = df.groupby('region')['mag'].mean()

# separating the two groups based on the 'mag' column

group1 = df[df['mag'] < mean\_mag\_by\_region.mean()]

group2 = df[df['mag'] >= mean\_mag\_by\_region.mean()]

# performing independent t-test between the two groups

t\_stat, p\_val = ttest\_ind(group1['mag'], group2['mag'], equal\_var=False)

print("T-test statistic: ", t\_stat)

print("P-value: ", p\_val)

T-test statistic: -210.99984844737443

P-value: 0.0

Earthquake Prediction Analysis:

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

# Select the features we want to use

X = df[['latitude', 'longitude', 'depth', 'gap']]

y = df['mag']

# 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)

# Fit the model on the training set

model = LinearRegression()

model.fit(X\_train, y\_train)

# Evaluate the model on the testing set

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print('Mean squared error:', mse)

print('R-squared score:', r2)

Mean squared error: 0.7318282185361162

R-squared score: 0.5632730346912534

In[12]:

import matplotlib.pyplot as plt

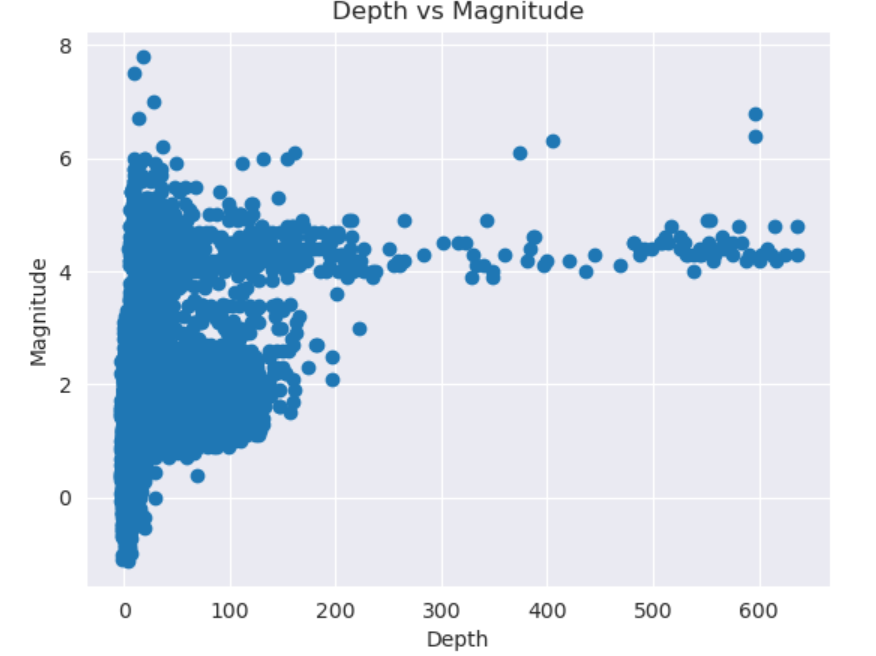
plt.scatter(df['depth'], df['mag'])

plt.xlabel('Depth')

plt.ylabel('Magnitude')

plt.title('Depth vs Magnitude')

plt.show()



In[13]:

import matplotlib.pyplot as plt

import seaborn as sns

# Scatter plot of magnitude vs. latitude

plt.scatter(df['latitude'], df['mag'], alpha=0.2)

plt.xlabel('Latitude')

plt.ylabel('Magnitude')

plt.title('Magnitude vs. Latitude')

plt.show()

# Scatter plot of magnitude vs. longitude

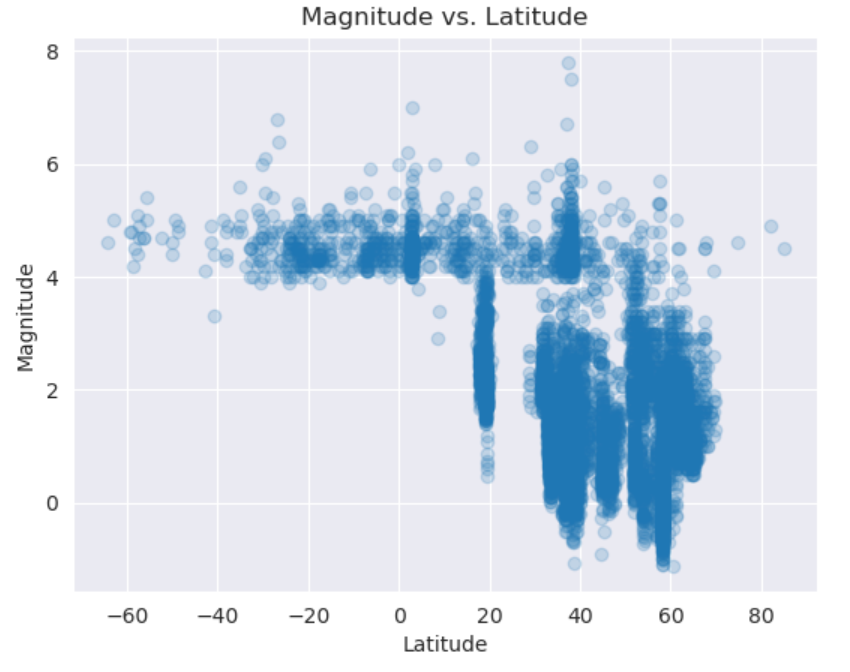
plt.scatter(df['longitude'], df['mag'], alpha=0.2)

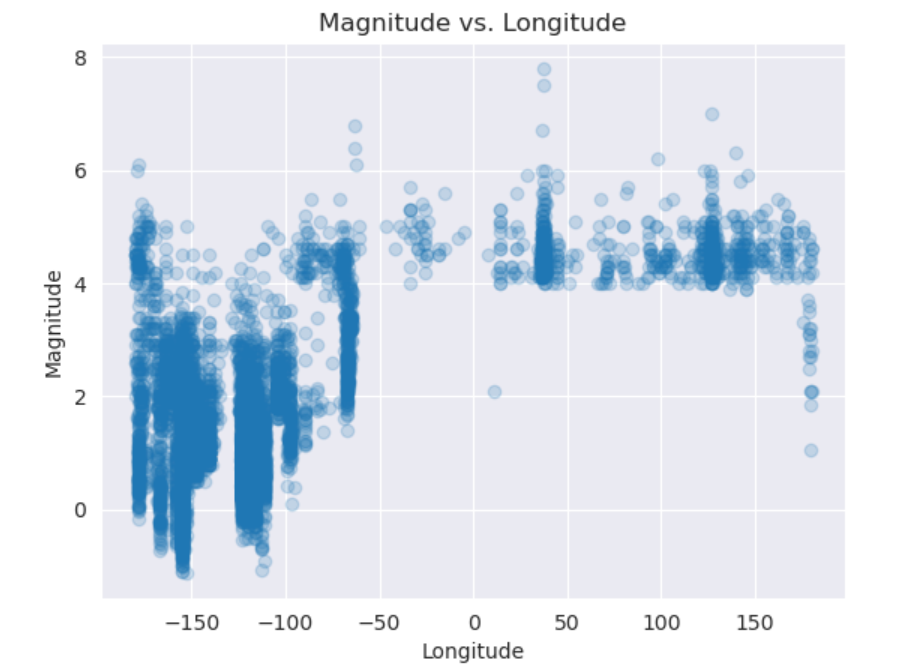
plt.xlabel('Longitude')

plt.ylabel('Magnitude')

plt.title('Magnitude vs. Longitude')

plt.show()



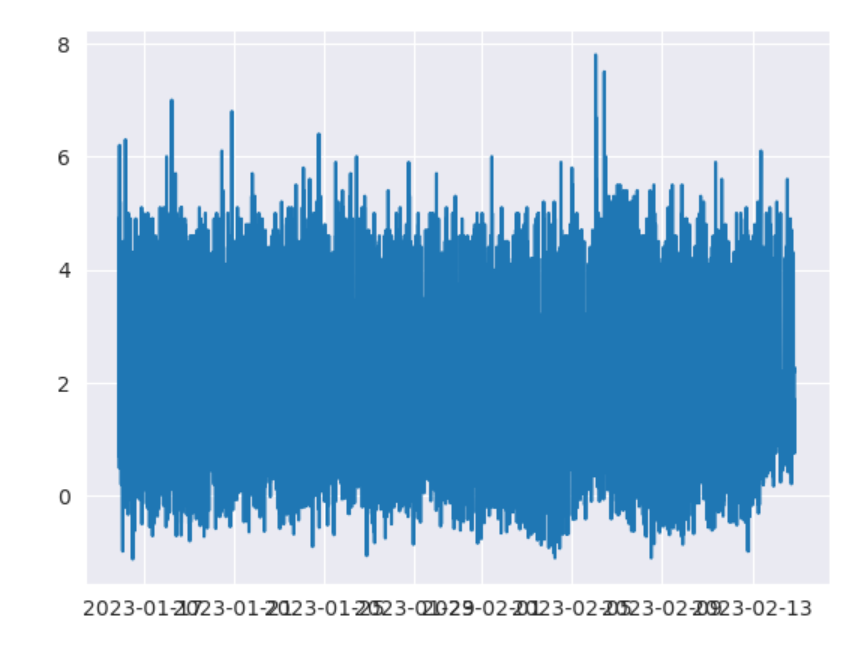


In [14]:

# Line plot of magnitude and time

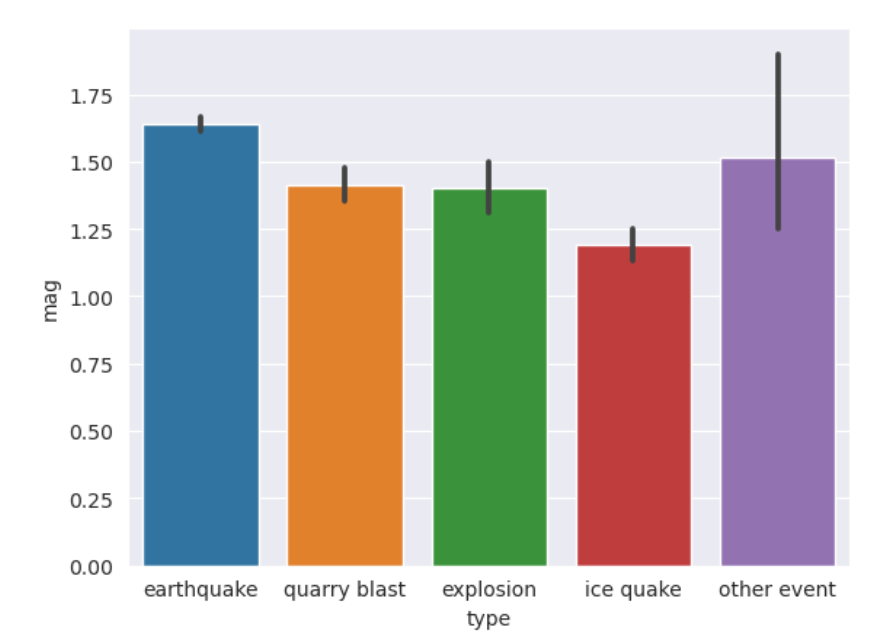
plt.plot(df['time'], df['mag'])

plt.show()

In [15]:

sns.barplot(data=df, x='type', y='mag')

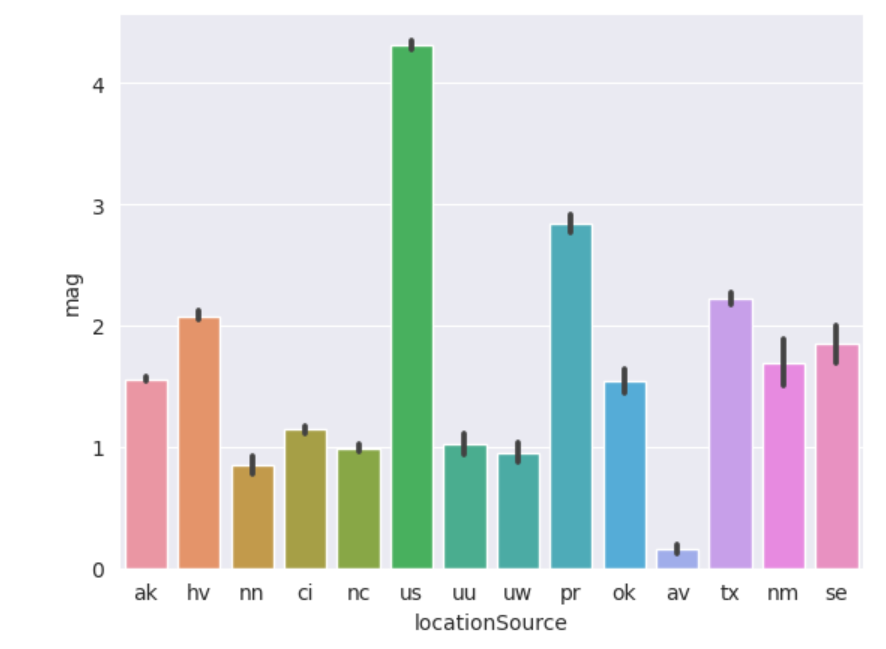
plt.show()



In [16]:

sns.barplot(data=df, x='locationSource', y='mag')

plt.show()



In [17]:

# Create a bar chart of the mean magnitude for each earthquake type

mean\_mag\_by\_type = df.groupby('type')['mag'].mean().sort\_values()

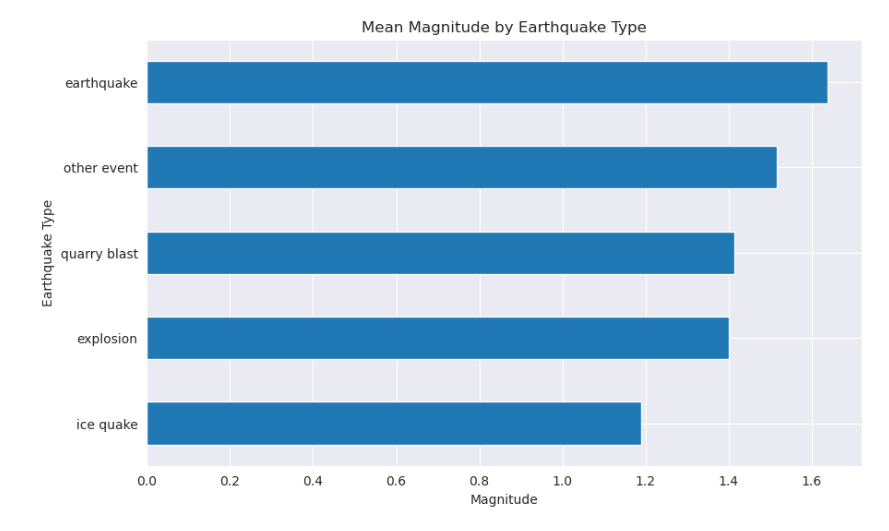
mean\_mag\_by\_type.plot(kind='barh', figsize=(10,6))

plt.title('Mean Magnitude by Earthquake Type')

plt.xlabel('Magnitude')

plt.ylabel('Earthquake Type')

plt.show()



In [18]:

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

# Prepare the input and target variables for the model

X = df[['latitude', 'longitude', 'depth']]

y = df['mag']

# 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 model and fit it to the training data

rf = RandomForestRegressor(n\_estimators=100, random\_state=42)

rf.fit(X\_train, y\_train)

# Use the model to make predictions on the test data

y\_pred = rf.predict(X\_test)

# Evaluate the model's performance

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Mean squared error: ", mse)

print("R-squared score: ", r2)

Mean squared error: 0.23816798930658079

R-squared score: 0.8578704939642665

latitude = 34.05

longitude = -118.25

depth = 10

year= 2023

import numpy as np

# Convert the predictor data to a numpy array

new\_data = np.array([[latitude, longitude, depth, year]])

# Use the trained model to make the prediction

predicted\_mag = model.predict(new\_data)

print("The predicted magnitude of the earthquake using the Random Forest model is: ", predicted\_mag[0])

The predicted magnitude of the earthquake using the Random Forest model is: 1.3789007159051776