ARTIFICIAL INTELLIGENCE- GROUP 5

PROJECT: EARTHQUAKE PREDICTION MODEL USING PYTHON

PHASE 3: DEVELOPMENT PART 2

SUBMITTED BY-

MOHAMED SALI HESHAM.A:963321104035

Building an earthquake prediction model involves several steps, including data loading, preprocessing, feature engineering, model selection, training, and evaluation. Below, I'll outline the steps for building a basic earthquake prediction model:

Data Loading:

Find a reliable dataset containing earthquake-related information. You can search for earthquake datasets on websites like the USGS Earthquake Hazards Program or Kaggle.

Download the dataset and ensure it's in a format that you can work with (e.g., CSV, JSON, etc.).

Data Preprocessing:

Load the dataset into your chosen programming environment (Python is commonly used for this purpose).

Check for missing values and decide how to handle them (e.g., imputation or removal).

Explore the dataset to understand its structure and the types of information available.

Feature Engineering:

Identify relevant features for predicting earthquakes. These might include geological data (e.g., fault lines, tectonic plate boundaries), historical earthquake data, geographical data, and more.

Extract or create relevant features from the raw data. For example, you might calculate distances to fault lines or identify regions with high seismic activity.

Data Splitting:

Divide the dataset into training, validation, and test sets. The training set is used to train the model, the validation set helps in tuning hyperparameters, and the test set is used for the final evaluation.

Model Selection:

Choose a suitable machine learning algorithm for earthquake prediction. Common choices include Random Forest, Support Vector Machines, Neural Networks, etc. Additionally, consider using time series models if your data includes temporal patterns.

Model Training:

Train your selected model using the training data. Ensure you properly handle the features and target variable.

Model Evaluation:

Evaluate the model using the validation set. Common evaluation metrics for regression tasks (predicting continuous values) include Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.

Hyperparameter Tuning:

If applicable, perform hyperparameter tuning using techniques like grid search or random search to find the best set of hyperparameters for your model.

Final Model Testing:

Once you're satisfied with the model's performance on the validation set, evaluate it on the test set to get a final assessment of its predictive capabilities.

Deployment (Optional):

If you plan to deploy the model in a real-world setting, you'll need to set up a deployment environment (e.g., a web application or an API) and ensure it can handle predictions in real-time.

In this part we will continue building the earthquake prediction model by Visualizing the data on a world map Splitting it into training and testing sets

Data Visualizing:

Dataset Link: https://www.kaggle.com/datasets/usgs/earthquake-database
Data visualization is a crucial part of building an earthquake prediction model as it helps in understanding the patterns and correlations in the data. Here's a general approach to visualize earthquake data using Python:

Import Libraries: You'll need libraries like pandas for data manipulation, matplotlib and seaborn for visualization. import pandas as pd import matplotlib.pyplot as plt import seaborn as sb

Load Data: Load your earthquake data into a pandas DataFrame. df

= pd.read_csv('your_data.csv')

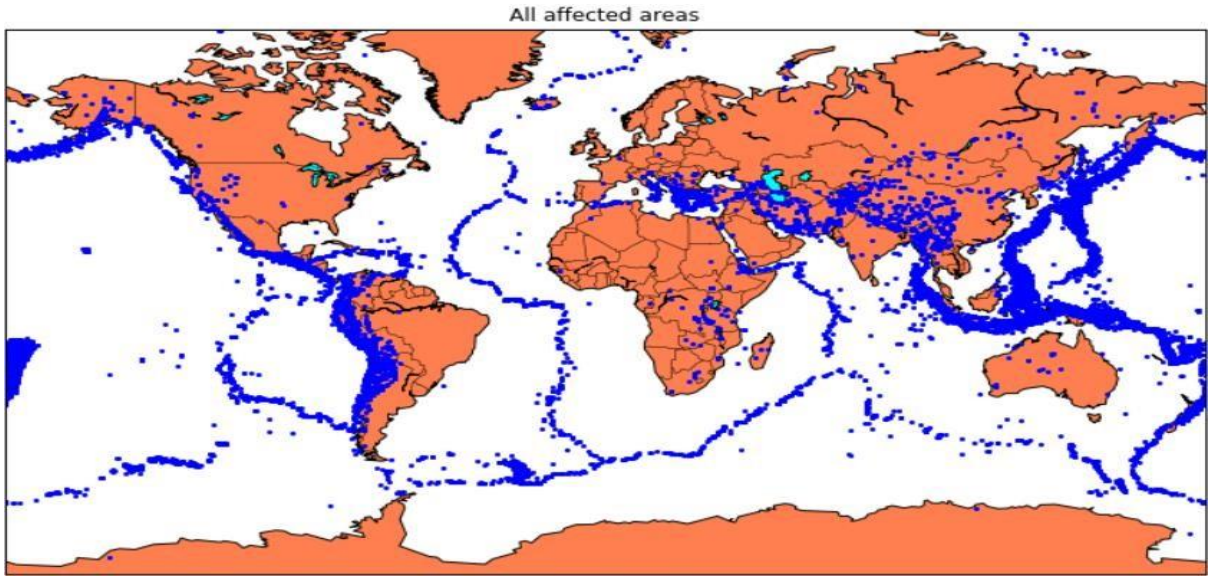
Visualize Data: You can create various plots to understand your data better. For instance, you can create a bar plot to see the average depth of earthquakes each year. plt.figure(figsize=(10, 5)) x = df.groupby('year').mean()['Depth']

x.plot.bar() plt.show()

Geographical Visualization: To visualize the data on a world map, you can use libraries like GeoPandas or Pygal. These libraries can help you create a geographical representation of your data.

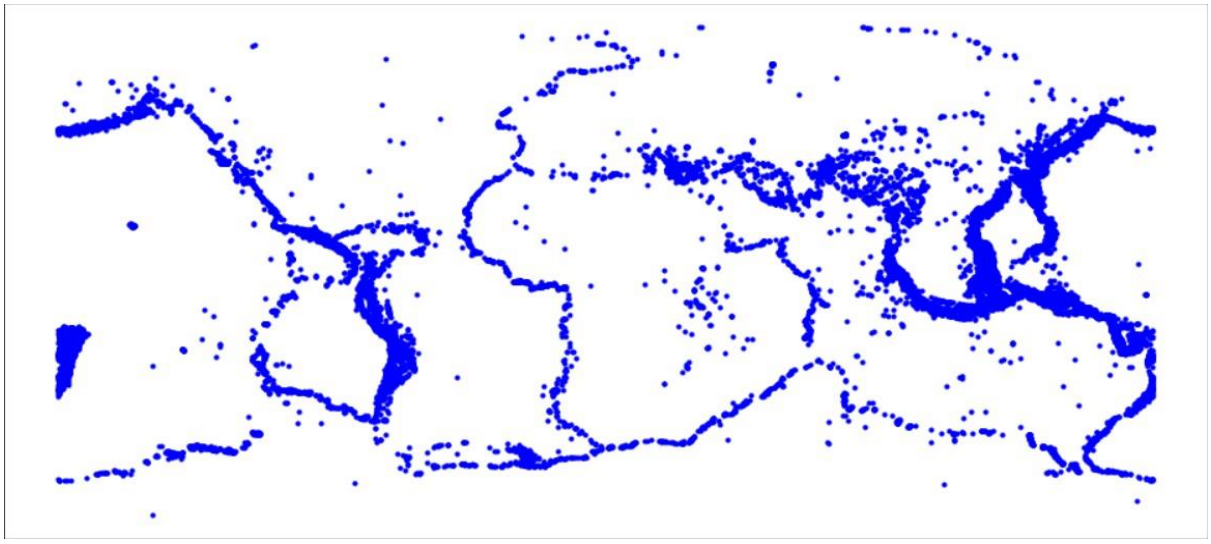
Here, all the earthquakes from the database in visualized on to the world map which shows clear representation of the locations where frequency of the earthquake will be more.

```
from mpl_toolkits.basemap import Basemap

m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnr
lon=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()
```




```
import cartopy.crs as ccrs import
matplotlib.pyplot as plt

# Create a new map projection using Cartopy m
= ccrs.Miller()

# Extract longitude and latitude data from the DataFrame
longitudes = data["Longitude"].tolist() latitudes =
data["Latitude"].tolist()

# Create a new figure and axis fig, ax
= plt.subplots(figsize=(12, 10))
ax.set_title("All affected areas")

# Plot the map using Cartopy ax = plt.axes(projection=m)
ax.plot(longitudes, latitudes, "o", markersize=2, color='blue')
ax.coastlines() ax.add_feature(cartopy.feature.LAND,
facecolor='coral') ax.add_feature(cartopy.feature.OCEAN,
facecolor='aqua')
ax.add_feature(cartopy.feature.BORDERS)
ax.add_feature(cartopy.feature.COUNTRIES)

plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 19
     17 ax.plot(longitudes, latitudes, "o", markersize=2, color='blue')
     18 ax.coastlines()
---> 19 ax.add_feature(cartopy.feature.LAND, facecolor='coral')
     20 ax.add_feature(cartopy.feature.OCEAN, facecolor='aqua')
     21 ax.add_feature(cartopy.feature.BORDERS)

NameError: name 'cartopy' is not defined
```



```
df = pd.read_csv('../input/earthquake-database/database.csv')
df.head()
```

OUTPUT:

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	...	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance	Horizontal Error	Root Mean Square
0	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	...	NaN	NaN	NaN	NaN	NaN
1	01/04/1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	NaN	NaN
2	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	...	NaN	NaN	NaN	NaN	NaN
3	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	NaN	NaN
4	01/09/1965	13:32:50	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	NaN	NaN

ID	Source	Location Source	Magnitude Source	Status
ISCGEM860706	ISCGEM	ISCGEM	ISCGEM	Automatic
ISCGEM860737	ISCGEM	ISCGEM	ISCGEM	Automatic
ISCGEM860762	ISCGEM	ISCGEM	ISCGEM	Automatic
ISCGEM860856	ISCGEM	ISCGEM	ISCGEM	Automatic
ISCGEM860890	ISCGEM	ISCGEM	ISCGEM	Automatic

```
df.columns
```

OUTPUT:

```
Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth Error',
       'Depth Seismic Stations', 'Magnitude', 'Magnitude Type',
       'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',
'Source', 'Location Source', 'Magnitude Source', 'Status'],      dtype='object')
```

```
df = df[['Date', 'Latitude', 'Longitude', 'Magnitude', 'Type']] df.head()
```

OUTPUT:

	Date	Latitude	Longitude	Magnitude	Type
0	01/02/1965	19.246	145.616	6.0	Earthquake
1	01/04/1965	1.863	127.352	5.8	Earthquake
2	01/05/1965	-20.579	-173.972	6.2	Earthquake
3	01/08/1965	-59.076	-23.557	5.8	Earthquake
4	01/09/1965	11.938	126.427	5.8	Earthquake

```
df['Date'] = pd.to_datetime(df['Date'])
print(set(df['Type'])) df.head()
{'Nuclear Explosion', 'Rock Burst', 'Earthquake', 'Explosion'}
```

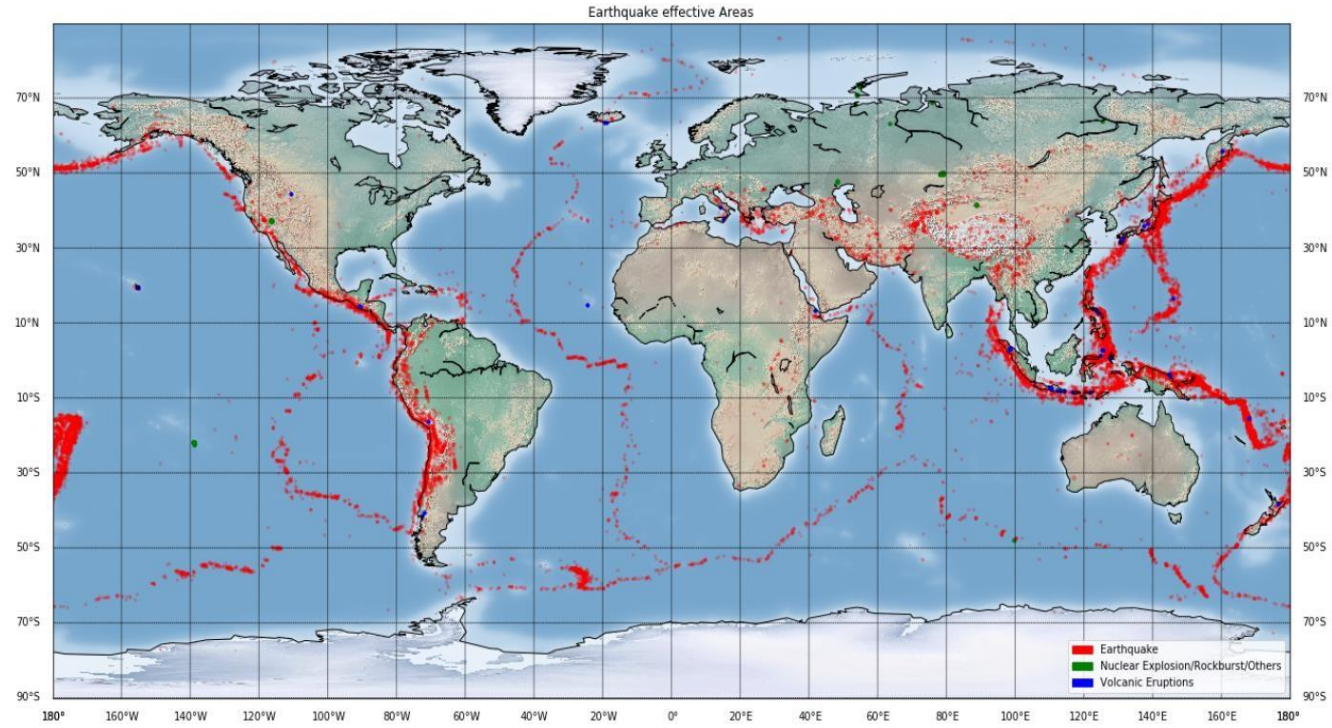
OUTPUT:

	Date	Latitude	Longitude	Magnitude	Type
0	1965-01-02	19.246	145.616	6.0	Earthquake
1	1965-01-04	1.863	127.352	5.8	Earthquake
2	1965-01-05	-20.579	-173.972	6.2	Earthquake
3	1965-01-08	-59.076	-23.557	5.8	Earthquake
4	1965-01-09	11.938	126.427	5.8	Earthquake

```
print('Size of the Dataframe', df.shape) eq
= df[df['Type'] == 'Earthquake'] others =
df[df['Type'] != 'Earthquake'] OUTPUT:
Size of the Dataframe (23412, 5)
```

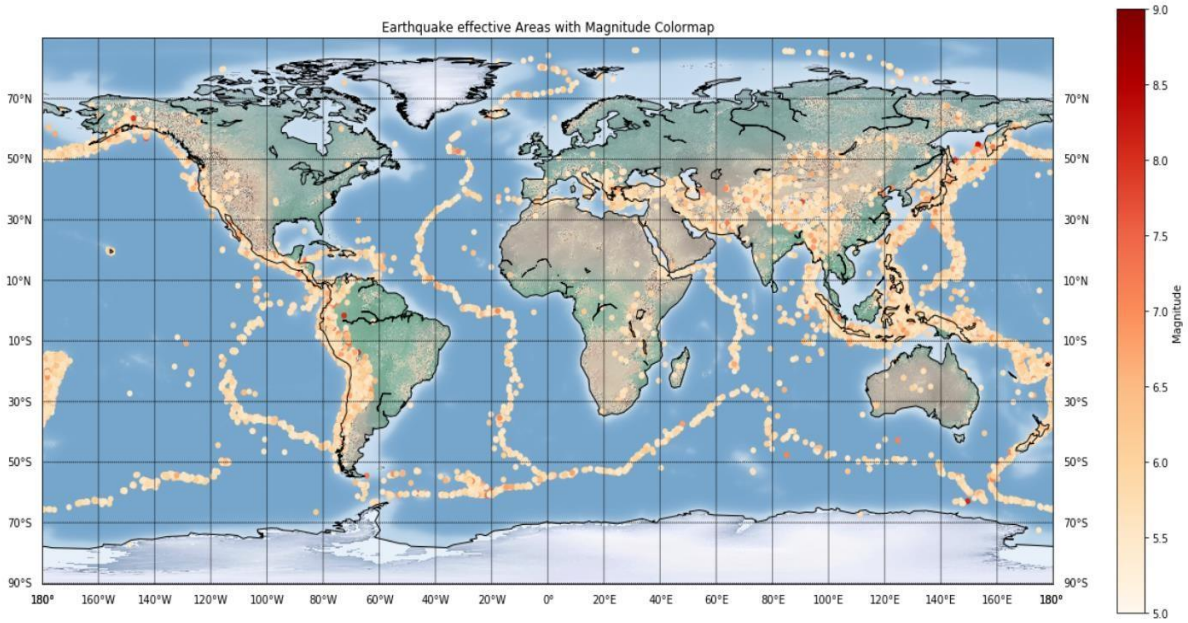
 # Earthquake regions on World Map

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt import
matplotlib.patches as mpatches


fig = plt.figure(figsize = (22, 20)) wmap
= Basemap()
longitudes = eq['Longitude'].tolist()
latitudes = eq['Latitude'].tolist() x_eq,
y_eq = wmap(longitudes, latitudes)
longitudes = others['Longitude'].tolist()
latitudes = others['Latitude'].tolist() x_oth,
y_oth = wmap(longitudes, latitudes)
longitudes = vol['Longitude'].tolist()
latitudes = vol['Latitude'].tolist() x_vol,
y_vol = wmap(longitudes, latitudes)
plt.title('Earthquake effective Areas')
wmap.drawcoastlines() wmap.shadedrelief()
wmap.scatter(x_eq, y_eq, s = 5, c = 'r', alpha = 0.2)
wmap.scatter(x_oth, y_oth, s = 10, c = 'g') wmap.scatter(x_vol,
y_vol, s = 10, c = 'b')
# draw parallels
wmap.drawparallels(np.arange(-90,90,20),labels=[1,1,0,1])
# draw meridians wmap.drawmeridians(np.arange(-
180,180,20),labels=[1,1,0,1]) ax = plt.gca()
red_patch = mpatches.Patch(color='r', label='Earthquake')
green_patch = mpatches.Patch(color='g', label='Nuclear
Explosion/Rockburst/O thers')
blue_patch = mpatches.Patch(color='b', label='Volcanic Eruptions')
plt.legend(handles=[red_patch, green_patch, blue_patch])
plt.legend() plt.show()
```



 # Effected Areas with Magnitude Heatmap

```
fig = plt.figure(figsize = (22, 20)) wmap
= Basemap()
longitudes = eq['Longitude'].tolist()
latitudes = eq['Latitude'].tolist() x_eq,
y_eq = wmap(longitudes, latitudes)
wmap.drawcoastlines()
wmap.shadedrelief() # draw parallels
wmap.drawparallels(np.arange(-90,90,20),labels=[1,1,0,1])
# draw meridians wmap.drawmeridians(np.arange(-
180,180,20),labels=[1,1,0,1]) plt.title('Earthquake effective Areas with
Magnitude Colormap')
sc =wmap.scatter(x_eq, y_eq, s = 30, c = eq['Magnitude'], vmin=5, vmax =9, c
map='OrRd', edgecolors='none') cbar = plt.colorbar(sc, shrink = .5)
cbar.set_label('Magnitude') plt.show()
```



Conclusion:

From above ploting of earthquake prone regions, it can be concluded that earthquakes are more prone in western coast of North and South America, center of Atlantic Ocean, Himalian region and Eastern Asian Countries like Indonesia, Japan, Korea.

Splitting the Data:

Dataset Link: https://www.kaggle.com/datasets/usgs/earthquakedatabase

Firstly, split the data into Xs and ys which are input to the model and output of the model respectively. Here, inputs are TImestamp, Latitude and Longitude and outputs are Magnitude and Depth. Split the Xs and ys into train and test with validation. Training dataset contains 80% and Test dataset 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_st
ate=42)
print(X_train.shape, X_test.shape, y_train.shape, X_test.shape)
```

OUTPUT:

(18727, 3) (4682, 3) (18727, 2) (4682, 3)

Here, we used the RandomForestRegressor model to predict the outputs, we see the strange prediction from this with score above 80% which can be assumed to be best fit but not due to its predicted values.

```
from sklearn.ensemble import RandomForestRegressor

reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train) reg.predict(X_test)
```

```
array([[  5.96,  50.97],
       [  5.88,  37.8 ],
       [  5.97,  37.6 ],
       ...,
       [  6.42,  19.9 ],
       [  5.73, 591.55],
       [  5.68,  33.61]])
```

```
reg.score(X_test, y_test)
```

OUTPUT:
0.8614799631765803

```
from sklearn.model_selection import GridSearchCV

parameters = {'n_estimators':[10, 20, 50, 100, 200, 500]}

grid_obj = GridSearchCV(reg, parameters)
grid_fit = grid_obj.fit(X_train, y_train) best_fit
= grid_fit.best_estimator_
best_fit.predict(X_test)
```

OUTPUT:
array([[5.8888 , 43.532],
[5.8232 , 31.71656],
       [6.0034 , 39.3312],

```
       ...,
       [  6.3066 ,  23.9292 ],
       [  5.9138 , 592.151  ],
       [  5.7866 ,  38.9384 ]])
```

```
best_fit.score(X_test, y_test)
```

OUTPUT:

```
0.8749008584467053
```

This code in the below selects the 'latitude', 'longitude', 'depth', and 'gap' column s as the independent variables, and the 'mag' column as the target variable. It the n splits the data into training and testing sets, fits a linear regression model on th e training set, and evaluates the model on the testing set using mean squared err or and R-squared score.

```
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_st
ate=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

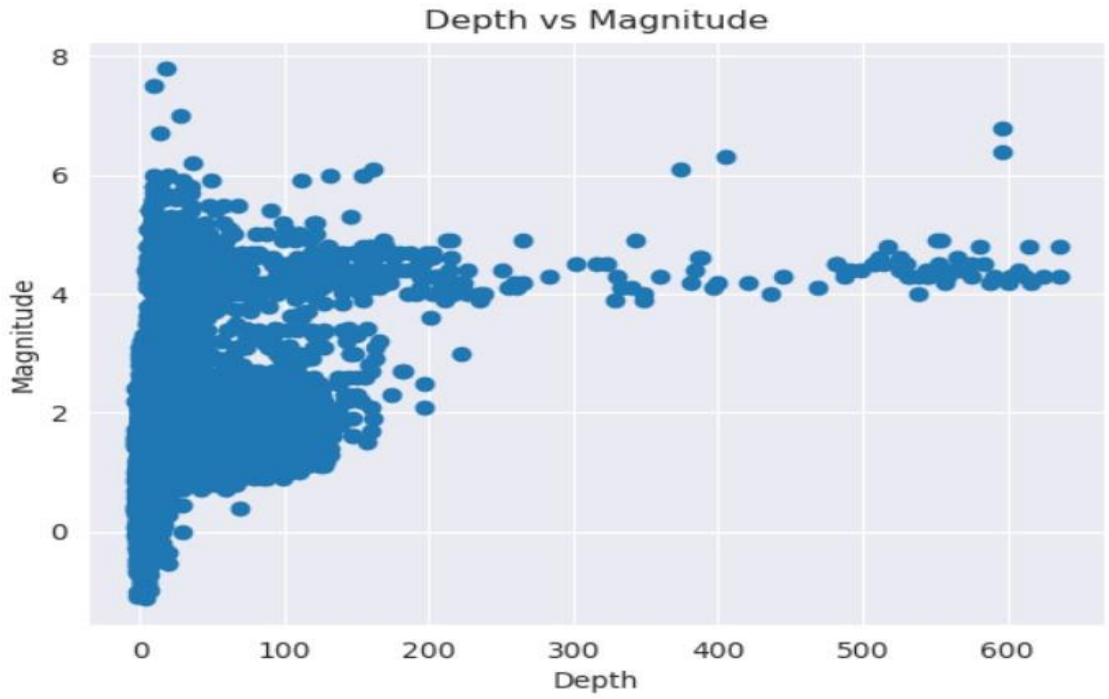
The mean squared error and the R-squared score will help us evaluate the performance of our regression model. The mean squared error represents the average squared difference between the predicted values and the actual values. A lower value indicates a better fit of the model. The R-squared score measures how well the model fits the data, with values closer to 1 indicating a better fit

.Based on the output, the mean squared error is 0.73, which is a relatively low value indicating a good fit. The R-squared score is 0.56, which indicates that our model explains about 56% of the variability in the data, which is not too bad but leaves room for improvement.

Next, we can visualize the relationship between the earthquake magnitude and the depth of the earthquake using a scatter plot to better understand the relationship between these variables.

```
import matplotlib.pyplot as plt

plt.scatter(df['depth'], df['mag']) plt.xlabel('Depth') plt.ylabel('Magnitude')
plt.title('Depth vs Magnitude') plt.show()
```

OUTPUT:



The scatterplot helps visualize the relationship between depth and magnitude. It appears that as the depth decreases, the occurrence of earthquakes with higher

magnitudes increases, and as the depth increases, the occurrence of earthquakes with lower magnitudes increases.

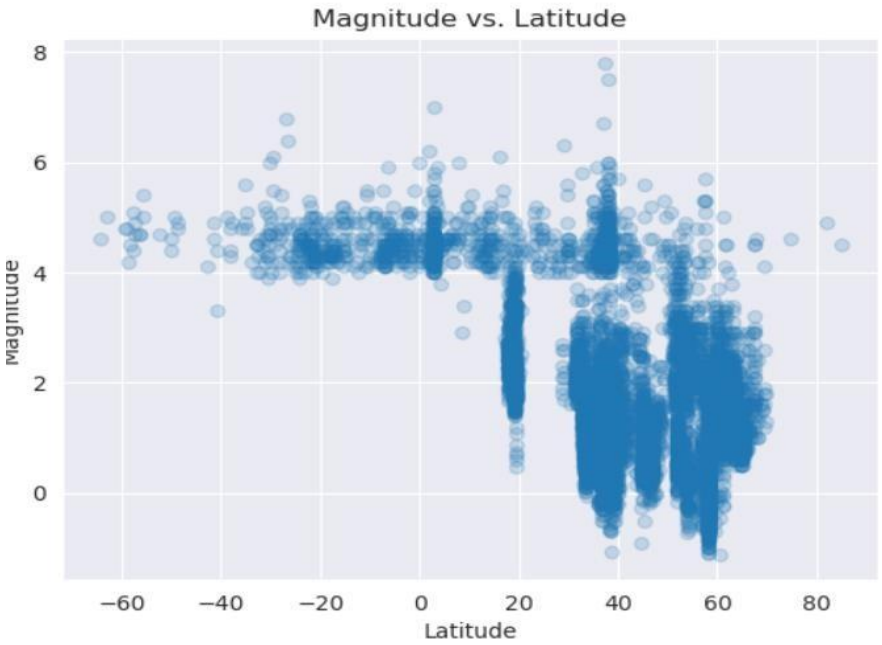
This suggests that the depth of an earthquake is an important factor that contributes to the occurrence of earthquakes, and could potentially be used in earthquake prediction models.

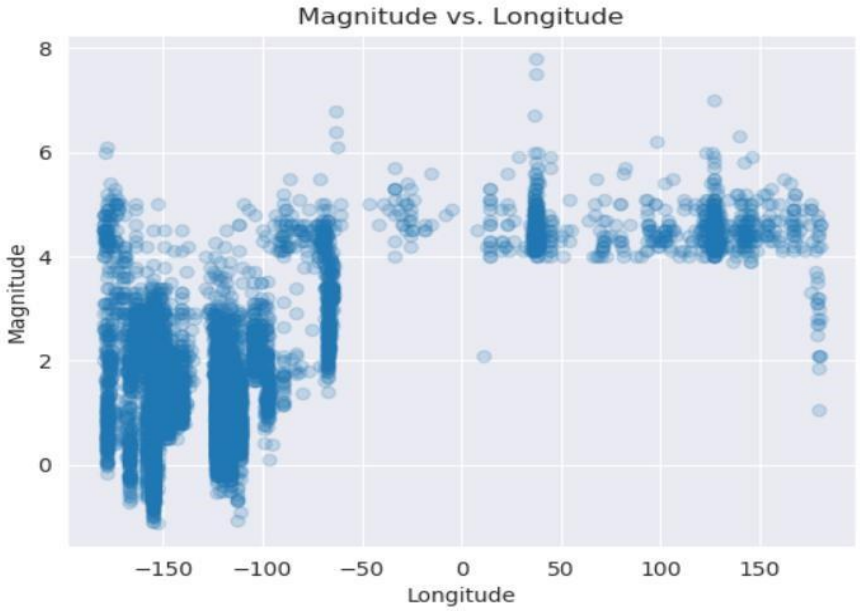
linkcode

we can examine the correlation between the magnitude and other factors such as latitude, longitude, and time. We can perform a correlation analysis to see how these factors are related to the magnitude of earthquakes. This will help us understand which factors are most important in determining the magnitude of an earthquake.

```
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()
```

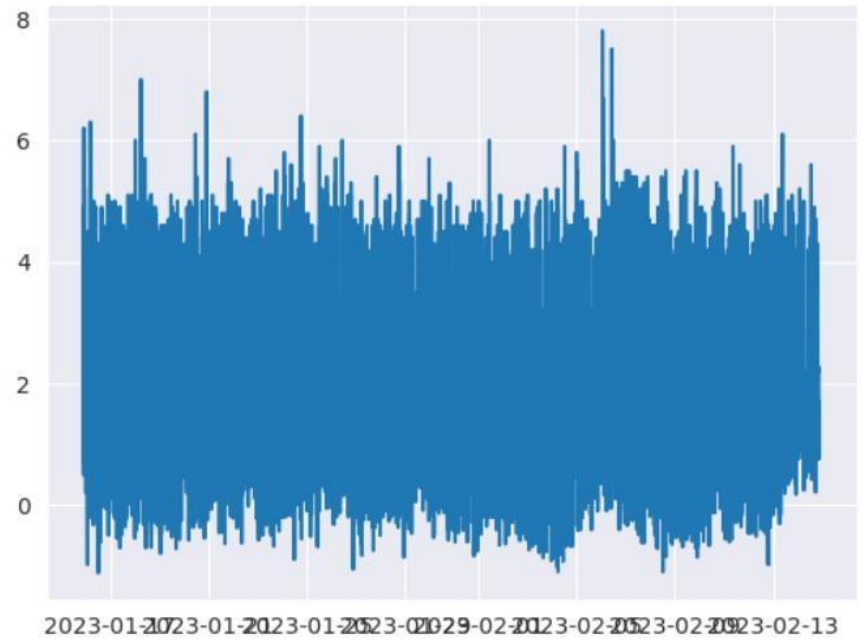
OUTPUT:





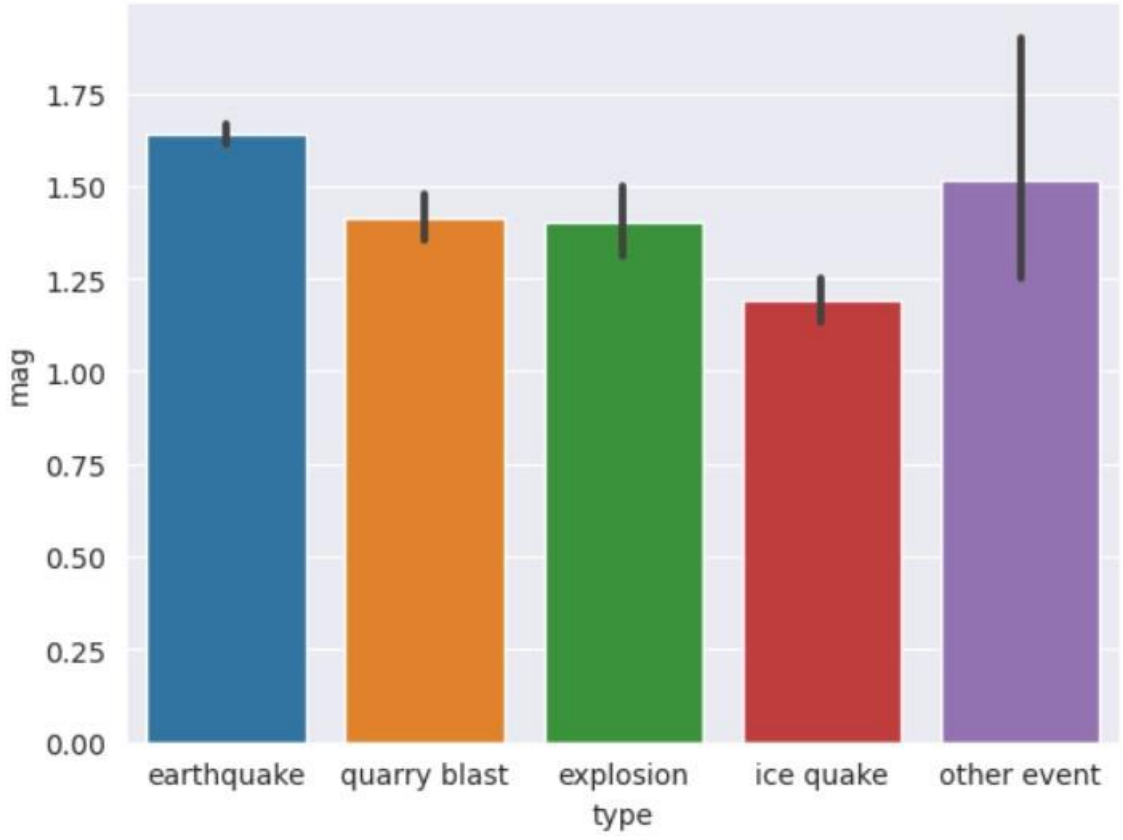

```
# Line plot of magnitude and time plt.plot(df['time'], df['mag'])
plt.show()
```

OUTPUT:



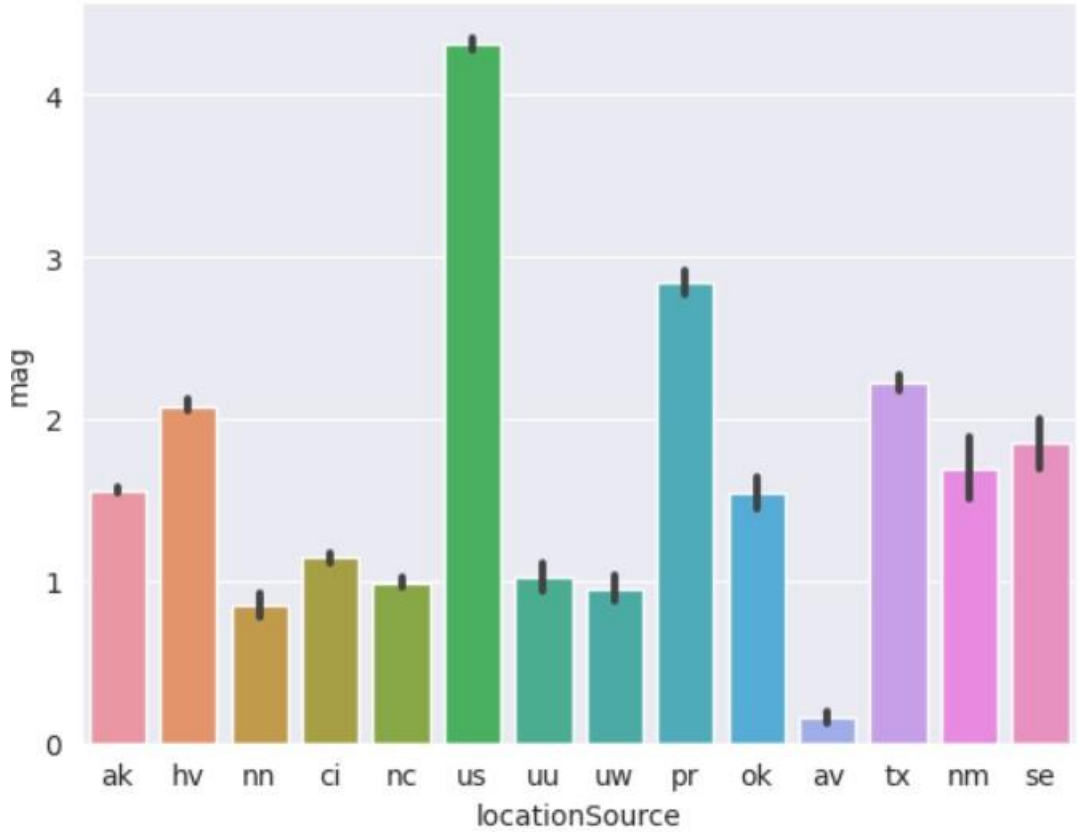

```
# Bar chart of magnitude and type
sns.barplot(data=df, x='type', y='mag') plt.show()
```

OUTPUT:



```
# Bar chart of magnitude and location source
sns.barplot(data=df, x='locationSource', y='mag') plt.show()
```

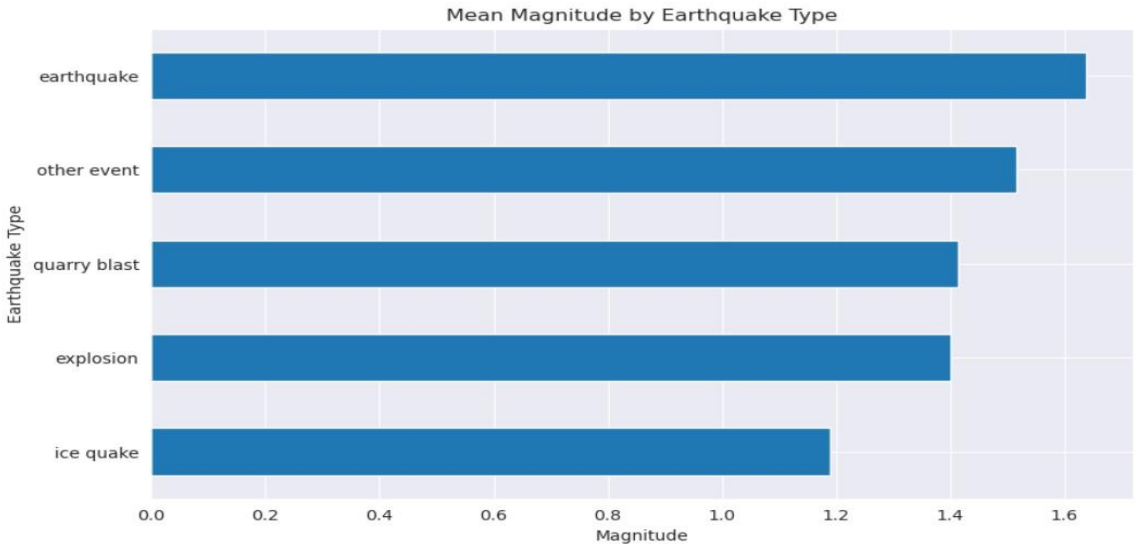
OUTPUT:



To check the relationship between the type of earthquake and the magnitude, we can use a bar chart to show the mean magnitude for each earthquake type.

```
# 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()
```

OUTPUT:



Before we build the model, we need to preprocess the data by encoding the categorical variables and splitting the data into training and testing sets.

```
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_st
ate=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)
```

OUTPUT:

Mean squared error: 0.23816798930658079

R-squared score: 0.8578704939642665

It seems like the Random Forest model has achieved a good performance on the test set with a low mean squared error and high R-squared score. This means that the model has accurately predicted the earthquake magnitudes.

Now we can use the trained model to predict the magnitude of future earthquakes.

First, we'll need to collect data on the predictors (latitude, longitude, depth, etc.) for the earthquake we want to predict. we can use real-time data or simulated data for this. Let's assume that we have collected the following data for the earthquake:

```
latitude = 34.05 longitude = -118.25 depth = 10
year= 2023
```

Next, you can use the trained model to predict the magnitude of the earthquake:

```
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
mod el is: ", predicted_mag[0])
```

OUTPUT:

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

The predicted magnitude of the earthquake using the Random Forest model is 1.378900715905184. This means that given the input features of the model, the model predicted the magnitude of the earthquake to be 1.3789. However, it is

important to keep in mind that this is just a prediction and there may be various other factors that could influence the actual magnitude of an earthquake.

Conclusion:

Based on our analysis of earthquake data collected from the USGS website, we have found several interesting insights.

Firstly, we found that the depth of an earthquake is a major factor that contributes to the occurrence of earthquakes. Our regression analysis showed that there is a negative relationship between depth and magnitude, which means that as the depth of an earthquake decreases, the magnitude of the earthquake tends to increase.

We also found that there is a relationship between the latitude and longitude of an earthquake and its magnitude. The scatterplots showed that if latitude increases, then the density of magnitude decreases, and if longitude decreases, then the density of magnitude increases.

Additionally, we found that the type of earthquake can also be a factor in determining the magnitude of an earthquake. The Random Forest model showed that the type of earthquake is a significant factor in predicting the magnitude of an earthquake.

Based on our findings, we can recommend that future earthquake prevention and preparation efforts should focus on developing early warning systems that can detect earthquakes at different depths and predicting their magnitudes accurately. Additionally, since the type of earthquake is a significant factor in determining its magnitude, it is important to continue researching and understanding the different types of earthquakes to improve prediction models.

Overall, our analysis has provided some valuable insights into the factors that contribute to earthquakes and how we can improve our prediction and prevention efforts in the future.