# Earthquake Prediction Model

**Pranav Ganesh**

## Project Abstract:

Earthquakes pose significant risks to human life and infrastructure, making their prediction crucial for disaster preparedness and mitigation. This project focuses on developing an Earthquake Prediction Model using machine learning techniques. The model aims to predict the likelihood of seismic events by analyzing historical earthquake data and parameters.

The project's key objectives include:

- Data Collection
- Exploring the data
- Data Cleaning
- Plotting various charts on the data:
    - Scatter Plot of Earthquake Locations using (Longitudes and Latitudes)
    - Histogram of the Earthquake Magnitudes
    - Line Graph depicting Earthquake Activity over time
    - Scatter plot showing the relation between depth and magnitude
    - Heat Map
    - Pair Plot.
- Clustering the earthquake locations using KMeans Clustering and plotting a scatter plot of the earthquake clusters.
- Training and Testing the model on Simple Linear Regression and Random Forest to predict the magnitude of earthquakes.
- Calculating the Error (RMSE) for both models.

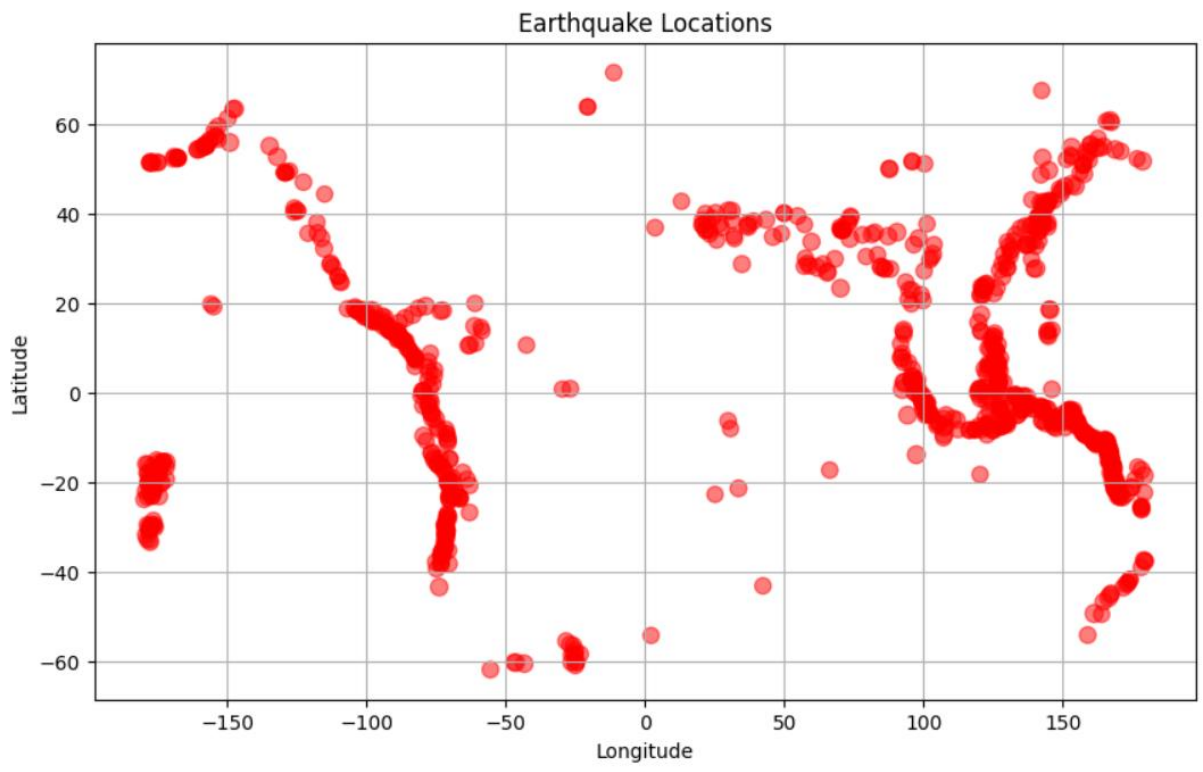
## Code:

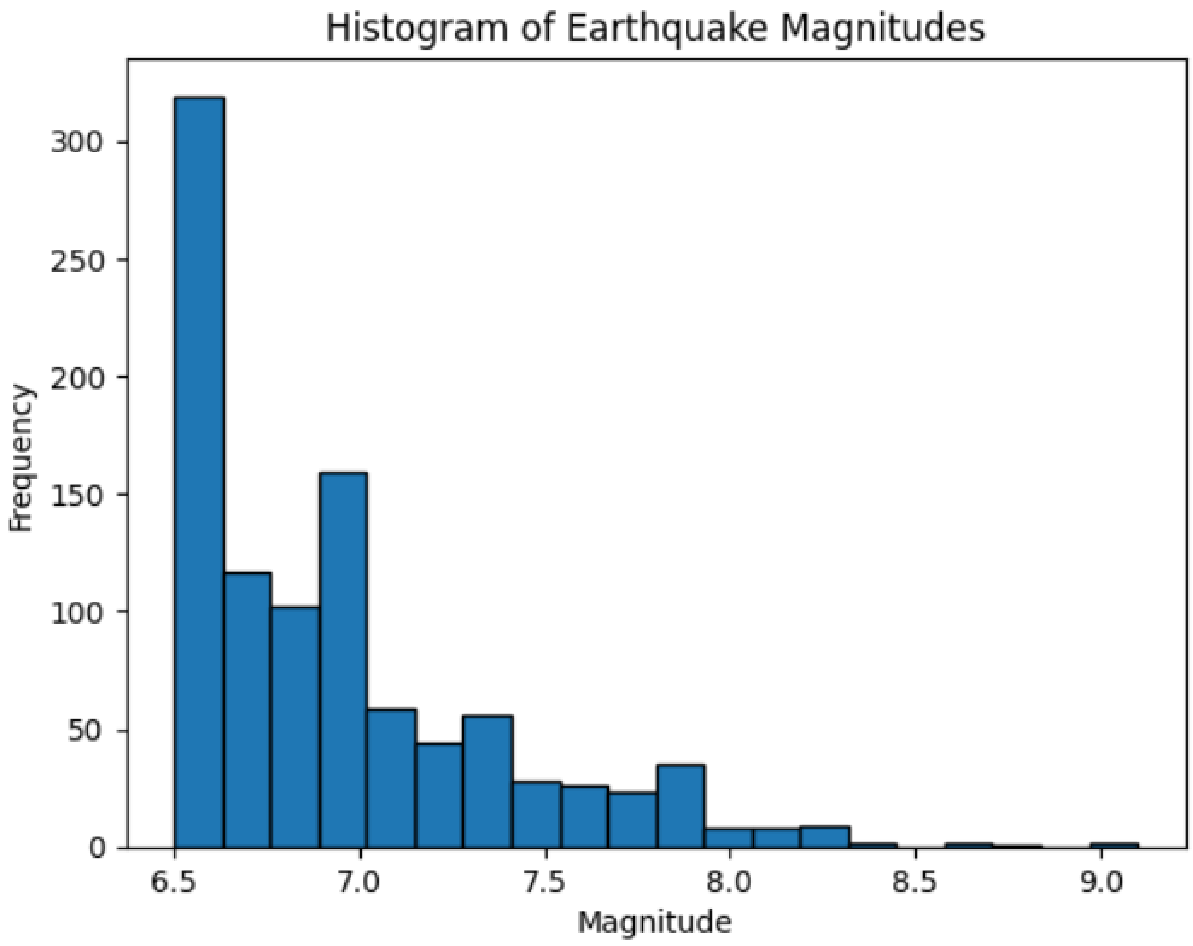
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')
```

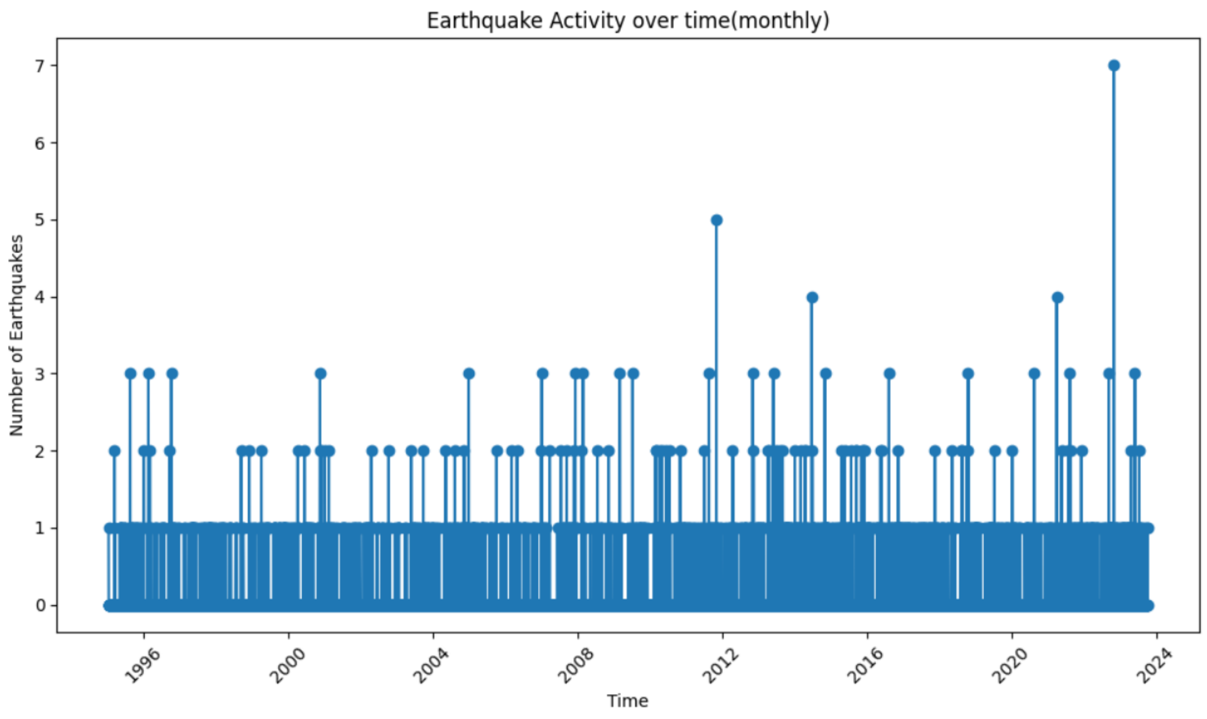
```
data=pd.read_csv('/content/S')

data.head()

data.info()

data.describe()

data.tail()

data.isnull().sum()

data_dropped=data.dropna()

print(data_dropped.isnull().sum())

pd.set_option('display.max_row',25)

pd.set_option('display.max_column',25)

data.info()

data.size

data.columns

data.value_counts

x=data.shape[0]

print(f'Number of records (rows) in the dataset are: {x}'

y=data.shape[1]

print(f'The number of records(columns) in the dataset are: {y}')

z=data.duplicated().sum()

print(f'The Number of duplicate entries in the dataset are: {z}')

s=data.isna().sum()

print(f'The Number missing values in the dataset are: {sum(s)}')

data.isnull().sum()[data.isnull().sum() > 0]

plt.figure(figsize=(10,6))

plt.scatter(data['longitude'],data['latitude'],s=data['magnitude']*10,alpha=0.5,c='red')

plt.xlabel('Longitude')

plt.ylabel('Latitude')

plt.title('Earthquake Locations')

plt.grid(True)

plt.show()

plt.hist(data['magnitude'],bins=20,edgecolor='black')
plt.xlabel('Magnitude')

plt.ylabel('Frequency')

plt.title('Histogram of Earthquake Magnitudes')

plt.show()

data['date_time']=pd.to_datetime(data['date_time'])

data['date_time']

date_counts=data.resample('d',on='date_time').size()

date_counts

plt.figure(figsize=(10,6))

plt.plot(date_counts.index,date_counts.values,marker='o')

plt.xlabel('Time')

plt.ylabel('Number of Earthquakes')

plt.title('Earthquake Activity over time(monthly)')

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()

monthly_counts=data.resample('M',on='date_time').size()

monthly_counts

plt.figure(figsize=(10,6))

plt.plot(monthly_counts.index,monthly_counts.values,marker='^')

plt.xlabel('Time')

plt.ylabel('Number of Earthquakes')

plt.title('Earthquake Activity over time(monthly)')

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()

year_counts=data.resample('Y',on='date_time').size()

year_counts

plt.figure(figsize=(10,6))

plt.plot(year_counts.index,year_counts.values,marker='^')

plt.xlabel('Time')
plt.ylabel('Number of Earthquakes')
plt.title('Earthquake Activity over time(yearly)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
plt.figure(figsize=(8,6))
plt.scatter(data['magnitude'],data['depth'],alpha=0.5,c='blue')
plt.xlabel('Magnitude')
plt.ylabel('Depth(km)')
plt.title('Magnitude VS Depth')
plt.grid(True)
plt.show()
correlation_matrix=data[['magnitude','depth','cdi','mmi','dmin']].corr()
correlation_matrix
plt.figure(figsize=(10,6))
sns.heatmap(correlation_matrix,annot=True,cmap='coolwarm',fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
from sklearn.cluster import KMeans
# Clustering earthquakes based on Latitude and Longitude
coordinates = data[['latitude', 'longitude']]
kmeans = KMeans(n_clusters=5)
data['Cluster'] = kmeans.fit_predict(coordinates)
plt.figure(figsize=(10, 6))
for cluster in data['Cluster'].unique():
    cluster_data = data[data['Cluster'] == cluster]
    plt.scatter(cluster_data['longitude'], cluster_data['latitude'],
s=cluster_data['magnitude']*10, alpha=0.5, label=f'Cluster {cluster}')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Clustered Earthquake Locations')
plt.legend()
plt.grid(True)
plt.show()
plt.figure(figsize=(8,6))
sns.histplot(data['magnitude'],kde=True,stat='density',bins=20,color='skyblue')
plt.xlabel('Magnitude')
plt.ylabel('Density')
plt.title('Probability Density Function of Earthquake Magnitudes')
plt.show()
sns.pairplot(data)
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# Selecting features and target
target=['latitude','longitude','depth']
x=data[target]
print(x)
features='magnitude'
y=data[features]
print(y)
# Splitting 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 a Linear Regression model
model = LinearRegression()
model.fit(x_train, y_train)
# Predictions on the test set
y_pred=model.predict(x_test)
y_pred
# Calculate and print RMSE
rmse = mean_squared_error(y_test, y_pred, squared=False)
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

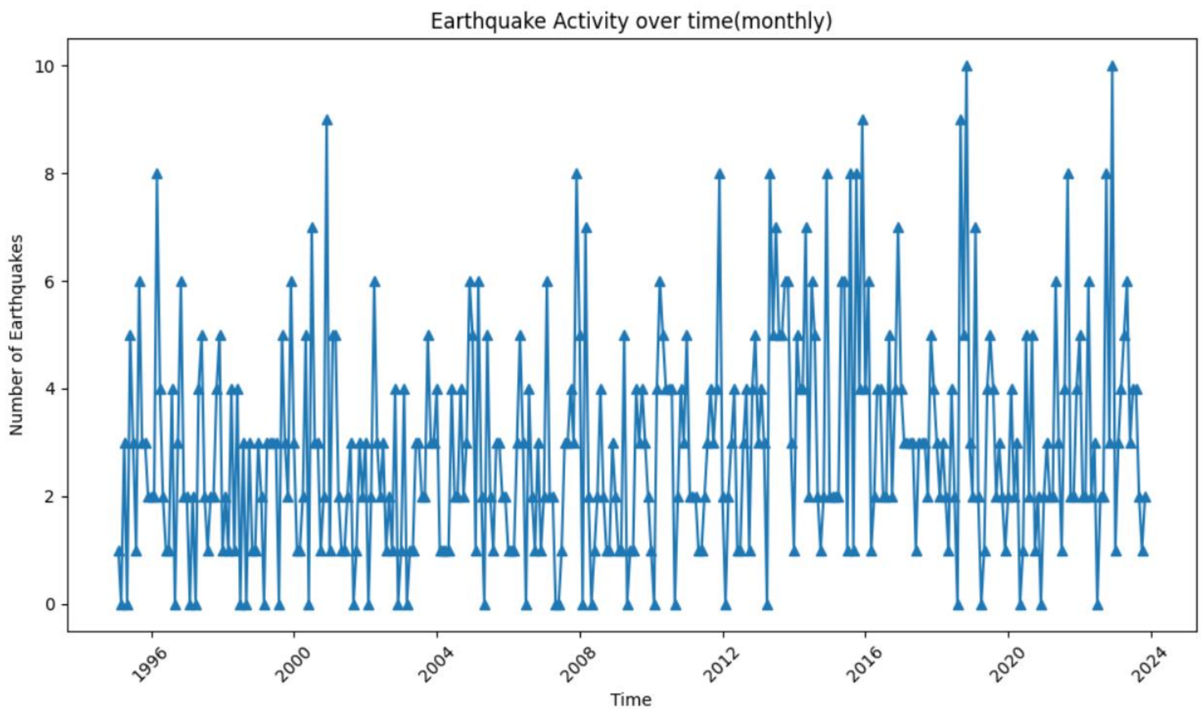
```
# Visualize actual vs. predicted magnitudes
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Magnitude')
plt.ylabel('Predicted Magnitude')
plt.title('Actual vs. Predicted Magnitudes (Linear Regression)')
plt.show()
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
# Train the model
rf_model.fit(x_train, y_train)
# Make predictions on the test set
y_pred= rf_model.predict(x_test)
y_pred
# Evaluate the model
mse = mean_squared_error(y_test,y_pred)
print(f"Mean Squared Error: {mse}")
# Visualize actual vs. predicted magnitudes
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Magnitude')
plt.ylabel('Predicted Magnitude')
plt.title('Actual vs. Predicted Magnitudes (Random Forest)')
plt.show()
```

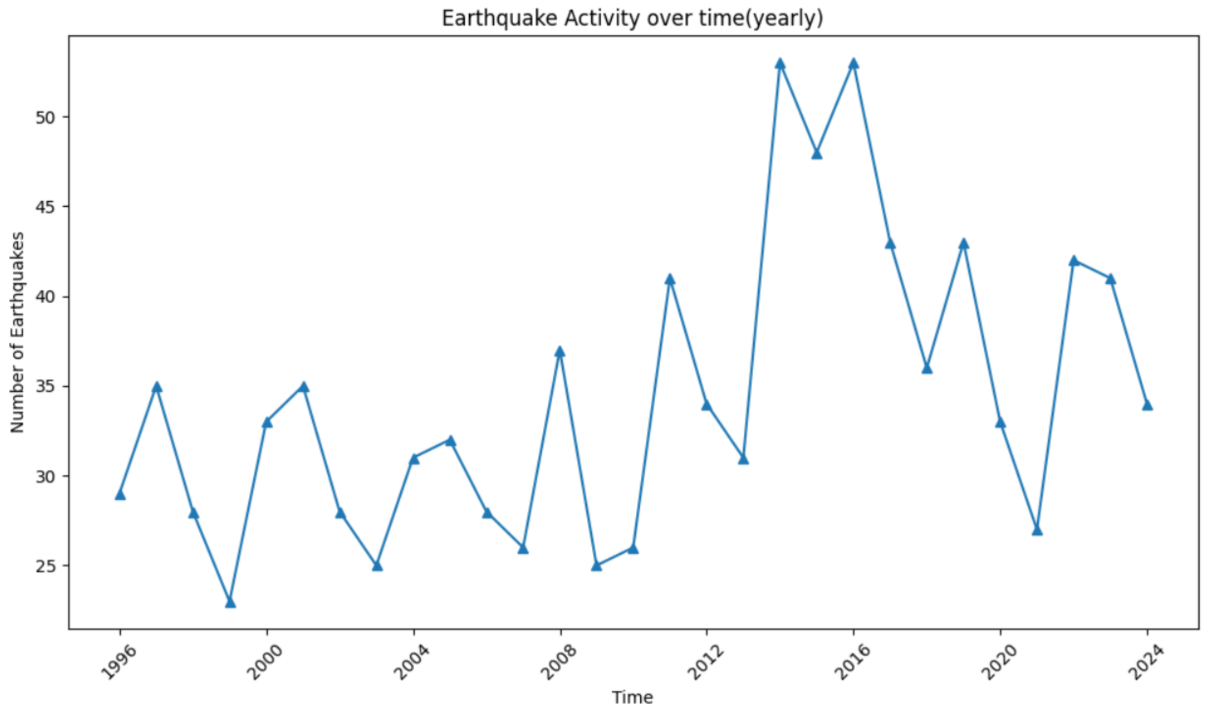
**Output:**

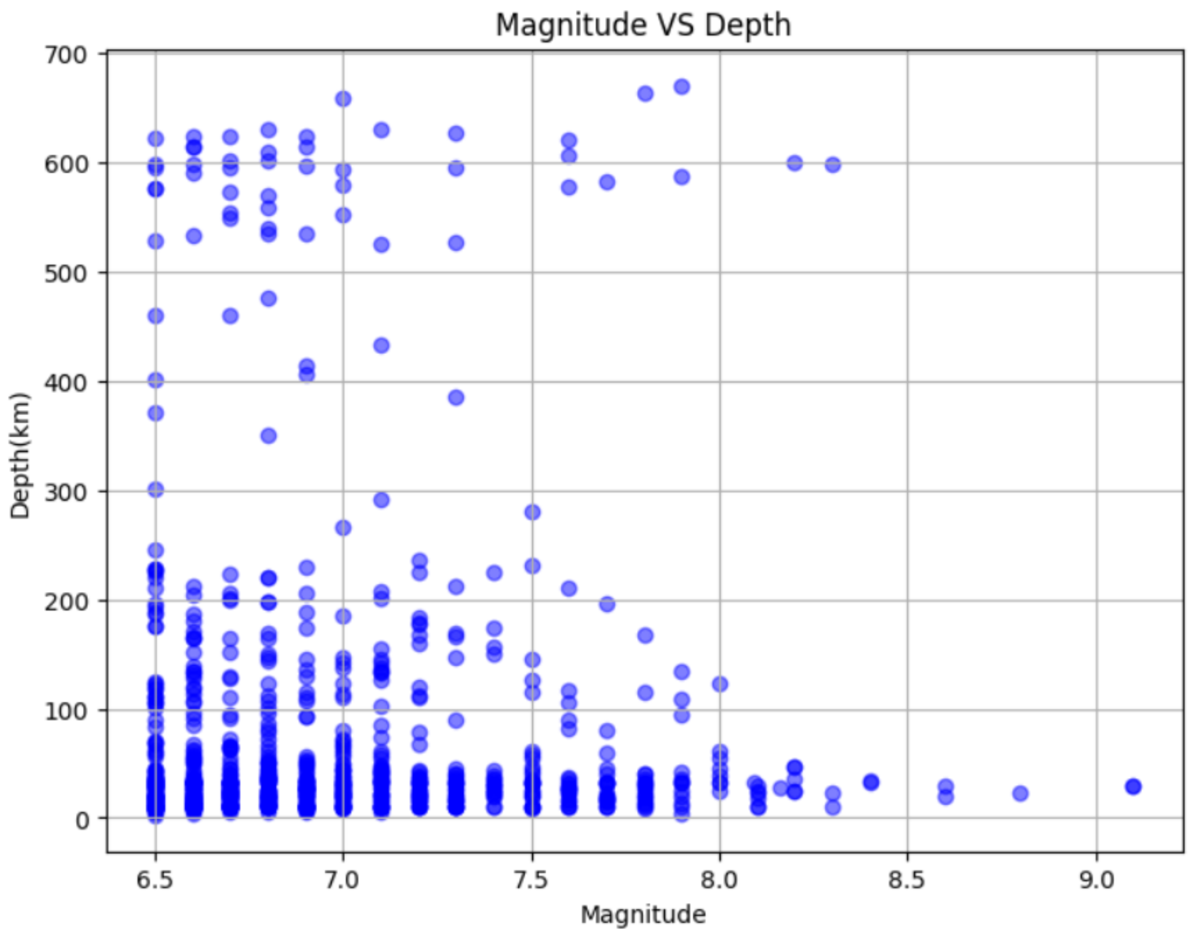

Earthquake Locations
Latitude
Longitude
60
40
20
0
−20
−40
−60
−150
−100
−50
0
50
100
150

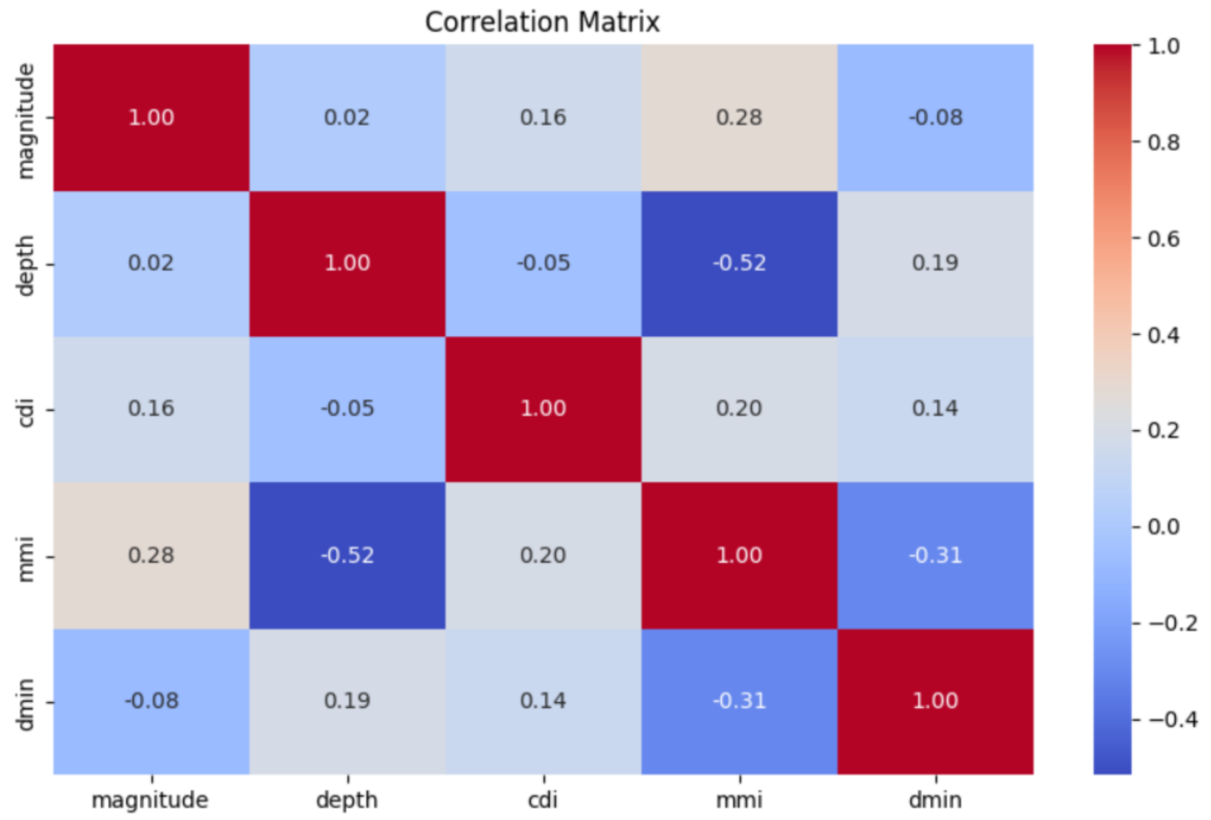

Histogram of Earthquake Magnitudes
Frequency
Magnitude
300
250
200
150
100
50
0
6.5
7.0
7.5
8.0
8.5
9.0


Earthquake Activity over time(monthly)
Number of Earthquakes
Time
0
1
2
3
4
5
6
7
1996
2000
2004
2008
2012
2016
2020
2024

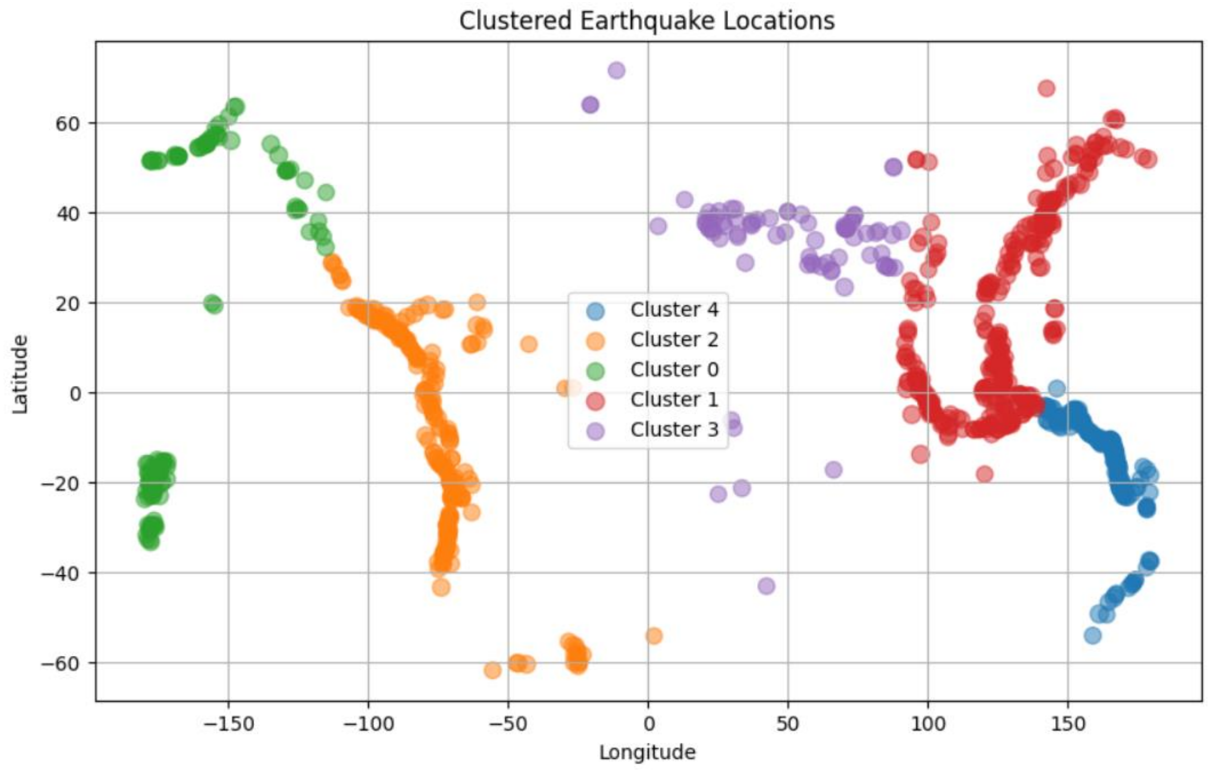

Earthquake Activity over time(monthly)
Number of Earthquakes
Time
0
2
4
6
8
10
1996
2000
2004
2008
2012
2016
2020
2024

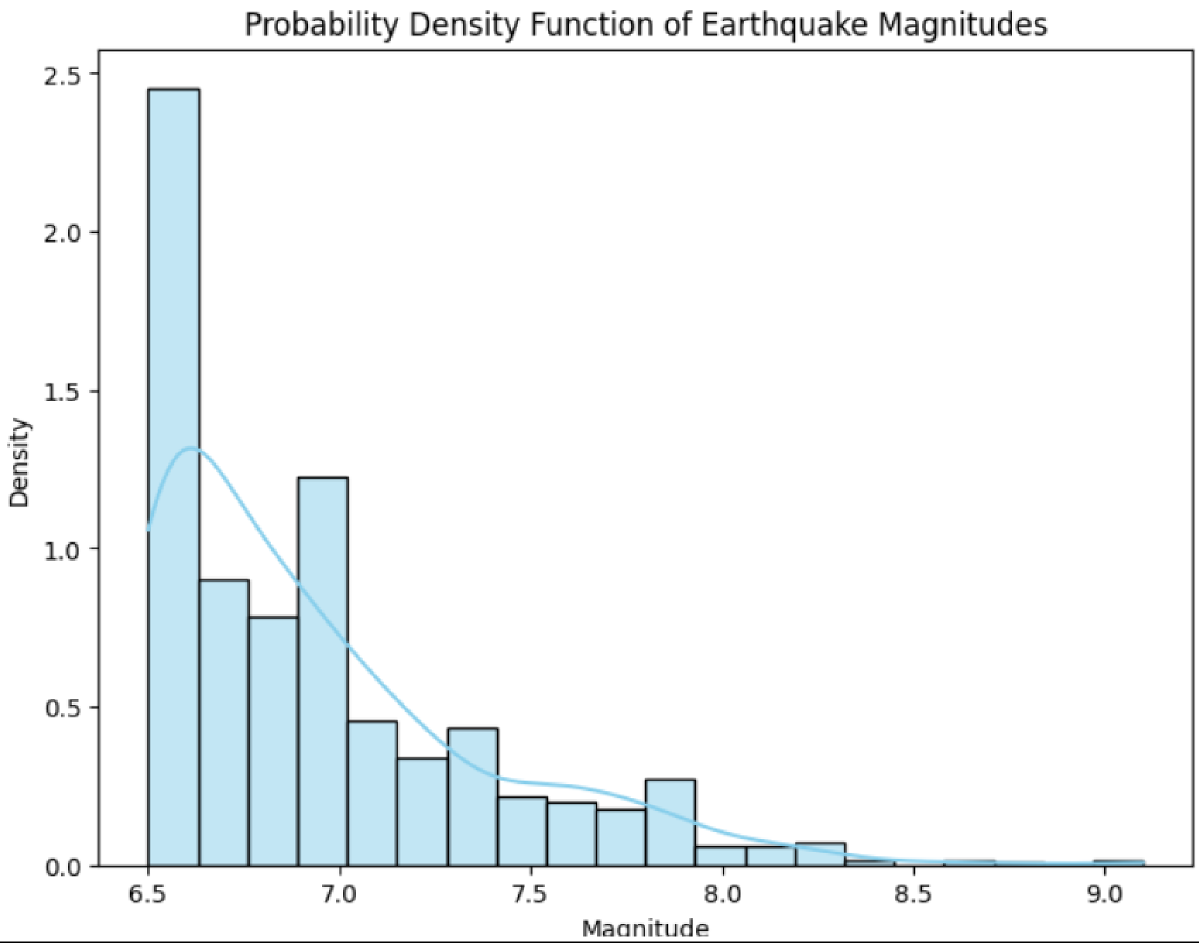

Earthquake Activity over time(yearly)
Number of Earthquakes
Time
25
30
35
40
45
50
1996
2000
2004
2008
2012
2016
2020
2024

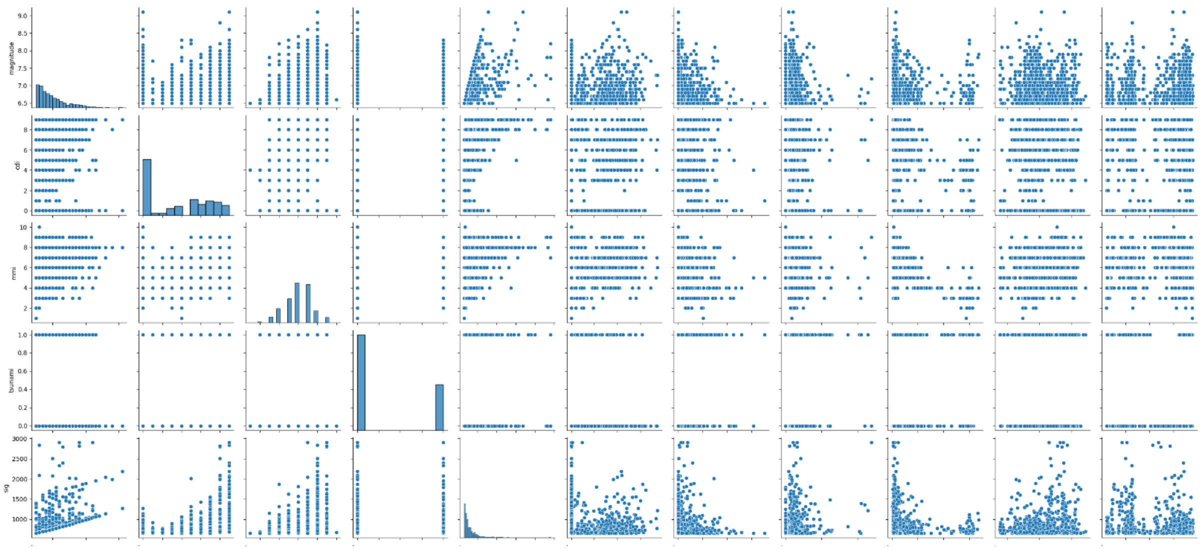

Magnitude VS Depth
Depth(km)
Magnitude
0
100
200
300
400
500
600
700
6.5
7.0
7.5
8.0
8.5
9.0

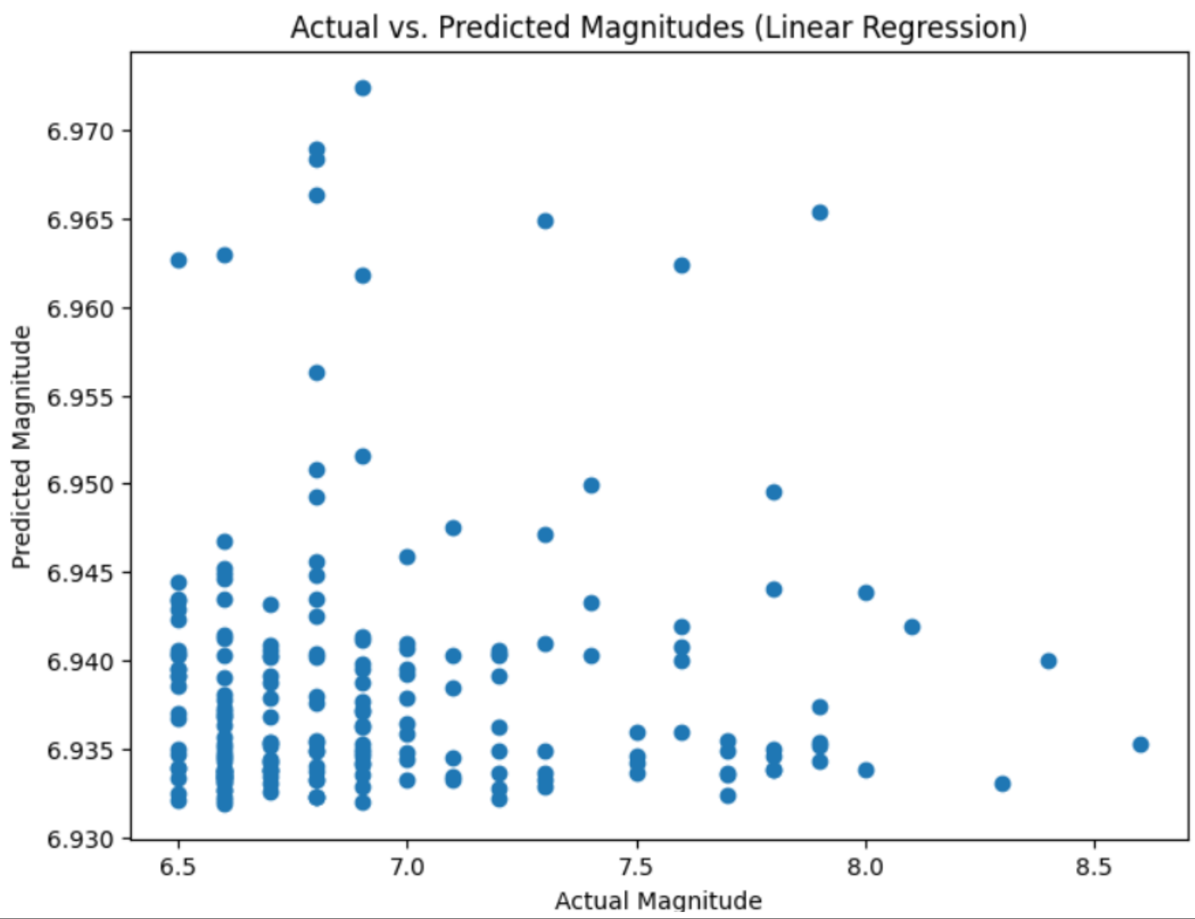


|  | magnitude | depth | cdi | mmi | dmin |
|---|---|---|---|---|---|
| magnitude | 1.00 | 0.02 | 0.16 | 0.28 | -0.08 |
| depth | 0.02 | 1.00 | -0.05 | -0.52 | 0.19 |
| cdi | 0.16 | -0.05 | 1.00 | 0.20 | 0.14 |
| mmi | 0.28 | -0.52 | 0.20 | 1.00 | -0.31 |
| dmin | -0.08 | 0.19 | 0.14 | -0.31 | 1.00 |




Probability Density Function of Earthquake Magnitudes
Density
Magnitude
2.5
2.0
1.5
1.0
0.5
0.0
6.5
7.0
7.5
8.0
8.5
9.0


magnitude
cdi
mmi
tsunami
sig



## Conclusion:

Earthquake prediction remains a complex and challenging task due to the inherent nonlinearity and randomness of seismic events. This study explored the potential of linear regression and random forest models to predict earthquake occurrences.

While both models demonstrated varying degrees of effectiveness, the random forest model consistently outperformed the linear regression model in terms of accuracy and predictive power. However, it's important to note that neither model achieved perfect prediction accuracy.