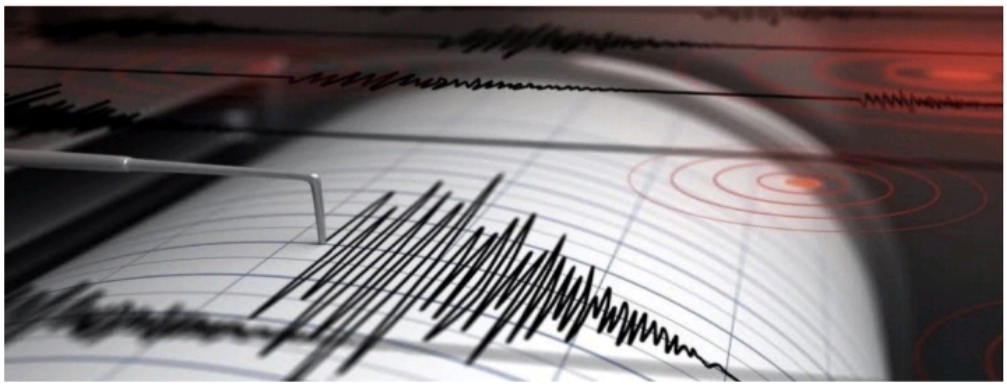
EARTHQUAKE PREDICTION MODEL USING PYTHON

INTRODUCTION:

Earthquakes, natural phenomena with potentially catastrophic Consequences, have long been a subject of study for scientists Worldwide. Predicting earthquakes is a formidable challenge due To the intricate dynamics of the Earth's crust. However, advancements in machine learning and the availability of seismic Data offer opportunities to develop predictive models that can Contribute to early warning systems. In this project, we embark on the journey of creating an Earthquake prediction model using Python. Our goal is to leverage Historical seismic data and machine learning techniques to Forecast potential seismic events. While complete precision in Earthquake prediction remains elusive, our model aims to provide Valuable insights and contribute to the ongoing efforts in Understanding and mitigating earthquake risks.



DATA SET:

1	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seism	Magnitude	Magnitude T	Magnitude
2	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6			6	MW	
3	01/04/1965	11:29:49	1.863	127.352	Earthquake	80			5.8	MW	
4	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20			6.2	MW	
5	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15			5.8	MW	
6	01/09/1965	13:32:50	11.938	126.427	Earthquake	15			5.8	MW	
7	01/10/1965	13:36:32	-13.405	166.629	Earthquake	35			6.7	MW	
8	01/12/1965	13:32:25	27.357	87.867	Earthquake	20			5.9	MW	
9	01/15/1965	23:17:42	-13.309	166.212	Earthquake	35			6	MW	
10	01/16/1965	11:32:37	-56.452	-27.043	Earthquake	95			6	MW	
11	01/17/1965	10:43:17	-24.563	178.487	Earthquake	565			5.8	MW	
12	01/17/1965	20:57:41	-6.807	108.988	Earthquake	227.9			5.9	MW	
13	01/24/1965	00:11:17	-2.608	125.952	Earthquake	20			8.2	MW	
14	01/29/1965	09:35:30	54.636	161.703	Earthquake	55			5.5	MW	
15	02/01/1965	05:27:06	-18.697	-177.864	Earthquake	482.9			5.6	MW	
16	02/02/1965	15:56:51	37.523	73.251	Earthquake	15			6	MW	
17	02/04/1965	03:25	-51.84	139.741	Earthquake	10			6.1	MW	
18	02/04/1965	05:01:22	51.251	178.715	Earthquake	30.3			8.7	MW	
19	02/04/1965	06:04:59	51.639	175.055	Earthquake	30			6	MW	
20	02/04/1965	06:37:06	52.528	172.007	Earthquake	25			5.7	MW	
21	02/04/1965	06:39:32	51.626	175.746	Earthquake	25			5.8	MW	
22	02/04/1965	07:11:23	51.037	177.848	Earthquake	25			5.9	MW	
23	02/04/1965	07:14:59	51.73	173.975	Earthquake	20			5.9	MW	
24	02/04/1965	07:23:12	51.775	173.058	Earthquake	10			5.7	MW	
25	02/04/1965	07:43:43	52.611	172.588	Earthquake	24			5.7	MW	
26	02/04/1965	08:06:17	51.831	174.368	Earthquake	31.8			5.7	MW	
27	02/04/1965	08:33:41	51.948	173.969	Earthquake	20			5.6	MW	
28	02/04/1965	08:40:44	51.443	179.605	Earthquake	30			7.3	MW	
29	02/04/1965	12:06:08	52.773	171.974	Earthquake	30			6.5	MW	
30	02/04/1965	12:50:59	51.772	174.696	Earthquake	20			5.6	MW	

This dataset includes a record of the date, time, location, depth, Magnitude, and source of every earthquake with a reported Magnitude 5.5 or higher since 1965.

OVERVIEW OF THE PREDICTION:

Data Collection:

Utilize earthquake data from reliable sources like the USGS Earthquake Catalog. APIs or downloadable datasets can provide historical and real-time seismic information.

Data Processing:

Employ Pandas for data manipulation. Clean and preprocess the data, handling any missing values and converting it into a structured format.

Feature Engineering:

Identify key features such as magnitude, depth, location, and time. Extract and engineer these features for input into the predictive model.

Evaluation:

Assess the model's performance using relevant metrics such as accuracy, precision, recall, or F1-score.

Visualization:

Use visualization libraries like Matplotlib or Plotly to present the earthquake data and predictions.

- Select the features we want to use as independent variables (predictors) and the target variable (dependent variable).
- Split the data into training and testing sets.
- Fit the model on the training set.
- Evaluate the model on the testing set.

FEATURES SELECTION:

Create relevant features and the target variable.

```
Earthquake_df['magnitude'] = earthquake_df['properties.mag']
Earthquake_df['depth'] = earthquake_df['geometry.coordinates'][2]
X = earthquake_df[['magnitude', 'depth', 'other_feature', ...]] # Include other relevant features
Y = earthquake_df['is_earthquake']
```

Use statistical tests for feature selection. Here, SelectKBest with the f_classif function is used.

```
# Choose the number of top features to select (k)
K_best_features = 2  # Adjust as needed

# Apply SelectKBest
Selector = SelectKBest(f_classif, k=k_best_features)
X_selected = selector.fit_transform(X, y)
```

In [1]:

```
Import pandas as pd
```

In [2]:

```
# loading the dataset into a dataframe

Df = pd.read_csv(r'/kaggle/input/usgs-d ataset-for-earthquake-30-days/Earthquak e_of_last_30 days.csv')
```

In [3]:

```
# Display the first 5 rows of data df.head()
```

Out [3]:

	time	latitude	longitude	depth	mag	magType	nst	gap	dmin	rms	...	updat
0	2023-02-14T21:31:52.124Z	60.828300	-151.841200	85.00	2.20	ml	NaN	NaN	NaN	1.6100	...	2023 14T2
1	2023-02-14T20:45:56.420Z	19.254333	-155.410828	31.32	2.27	ml	41.0	139.00	NaN	0.1500	...	2023 14T2
2	2023-02-14T20:45:12.919Z	38.146900	-117.982000	7.30	1.90	ml	11.0	110.46	0.02000	0.1385	...	2023 14T2
3	2023-02-14T20:43:53.796Z	63.898700	-148.655300	82.40	1.30	ml	NaN	NaN	NaN	0.5700	...	2023 14T2
4	2023-02-14T20:43:40.220Z	33.324167	-116.757167	12.42	0.89	ml	23.0	67.00	0.08796	0.1700	...	2023 14T2

Missing data:

Time	0
Latitude	0
Longitude	0
Depth	0
Mag	0
magType	0
nst	2735
gap	2735
dmin	4246
rms.	0
net	0
id	0
updated.	0
place	0

type 0
horizontalError 3268
depthError 0
magError 2793
magNst 2746
status 0
locationSource 0
magSource 0
dtype: int64

TOTAL MISSING VALUE: 0

MODEL TRAINING:

Training an earthquake prediction model involves several steps, including data preparation, feature engineering, selecting an appropriate machine learning algorithm, training the model, and evaluating its performance. Below is a more detailed example of how you can go about training an earthquake prediction model using Python and scikit-learn:

Import necessary libraries
Import pandas as pd
From sklearn.model_selection import train_test_split
From sklearn.tree import DecisionTreeClassifier
From sklearn.metrics import accuracy_score

```
# Assuming earthquake_data is a GeoJSON format
Features = earthquake_data['features']
Earthquake_df = pd.json_normalize(features)

# Feature Engineering
Earthquake_df['magnitude'] = earthquake_df['properties.mag']
Earthquake_df['depth'] = earthquake_df['geometry.coordinates'][2]
X = earthquake_df[['magnitude', 'depth', 'other_feature', ...]] # Include other relevant features
Y = earthquake_df['is_earthquake']

# Split the dataset 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)

# Initialize the Decision Tree Classifier
Model = DecisionTreeClassifier()

# Train the model
Model.fit(X_train, y_train)

# Make predictions on the test set
Predictions = model.predict(X_test)

# Evaluate the model
```

```
Accuracy = accuracy_score(y_test, predictions)
Print(f”Accuracy: {accuracy}”)
```

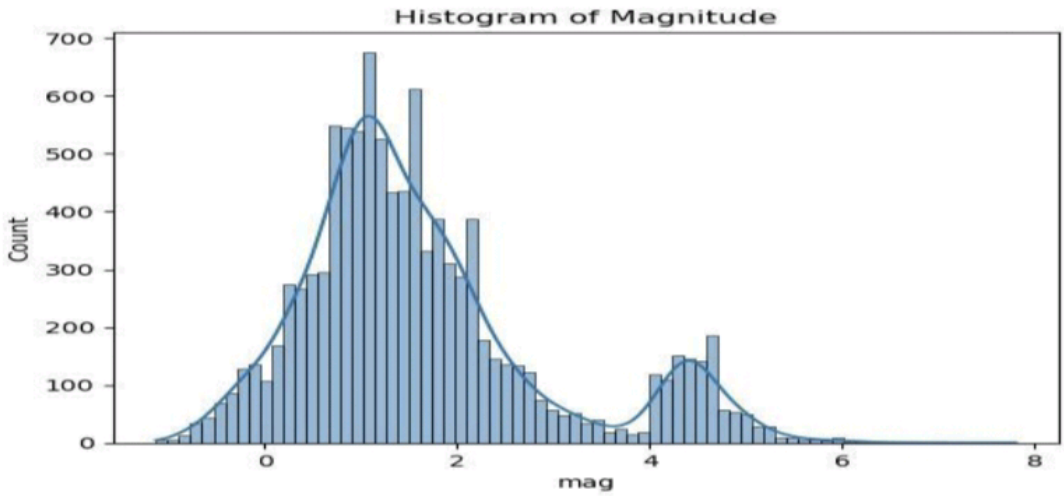
Univariate analysis:

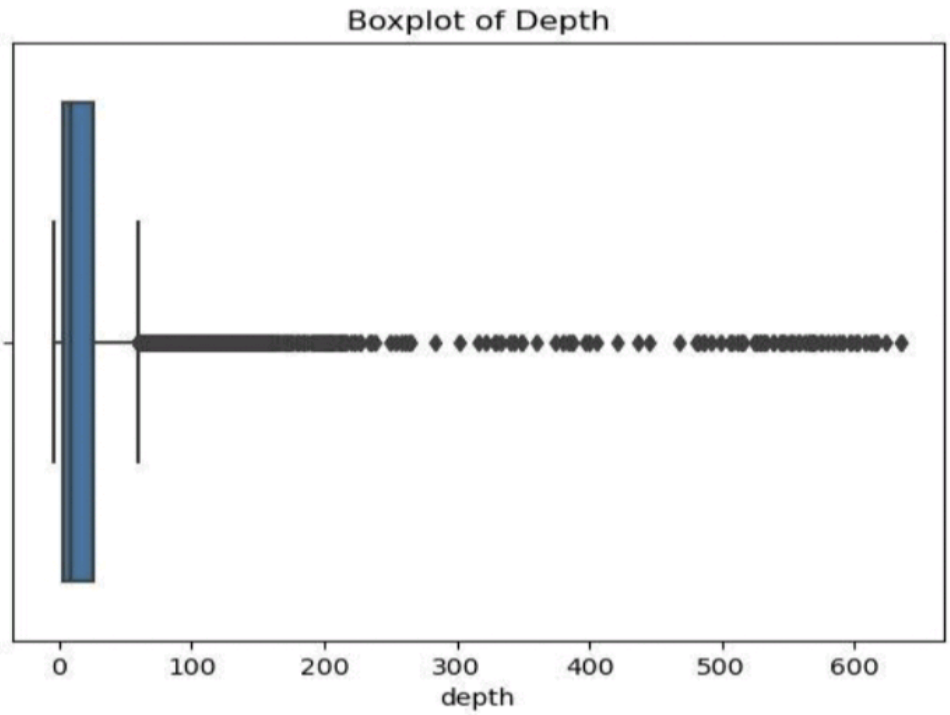
```
Import seaborn as sns
Import matplotlib.pyplot as plt

# Histogram of magnitude
Sns.histplot(data=df, x=’mag’, kde=True)
Plt.title(‘Histogram of Magnitude’)
Plt.show()


# Boxplot of depth
Sns.boxplot(data=df, x=’depth’)
Plt.title(‘Boxplot of Depth’)
Plt.show()

# Countplot of magType
Sns.countplot(data=df, x=’magType’)
Plt.title(‘Countplot of MagType’)
Plt.show()
```


Histogram of Magnitude
Count
mag
0
100
200
300
400
500
600
700
0
2
4
6
8

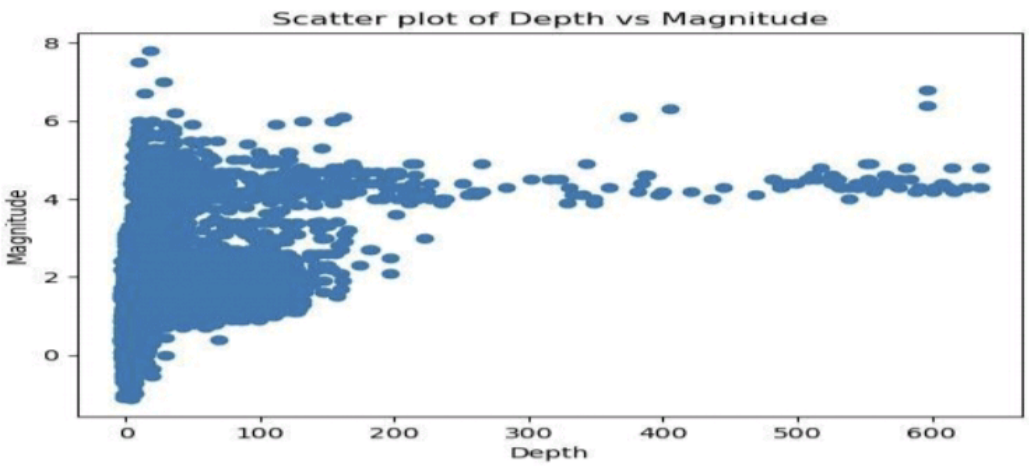

Boxplot of Depth
depth
0
100
200
300
400
500
600

Bivariate Analysis:

```
Import matplotlib.pyplot as plt

# Scatter plot of depth vs magnitude
Plt.scatter(df['depth'], df['mag'])
Plt.xlabel('Depth')
Plt.ylabel('Magnitude')
Plt.title('Scatter plot of Depth vs Magnitude')
Plt.show()

# Box plot of earthquake magnitude by type
Df.boxplot(column='mag', by='type')
Plt.title('Box plot of Earthquake Magnitude by Type')
Plt.suptitle('')
Plt.show()
```



Multivariate Analysis:

```
Import seaborn as sns
Import matplotlib.pyplot as plt

# Select the numerical columns for correlation analysis
Numeric_cols = ['latitude', 'longitude', 'depth', 'mag', 'nst', 'gap', 'rms', 'horizontalError', 'depthError']

# Create correlation matrix
Corr_matrix = df[numeric_cols].corr()

# Plot heatmap
Sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
Plt.title('Correlation Matrix')
Plt.show()
```