EARTHQUAKE PREDICTION USING PYTHON

PHASE- III: Loading and Preprocessing the Dataset.

INTRODUCTION :-

- Earthquake is a natural phenomenon whose occurrence predictability is still a hot topic in academia. This is because of the destructive power it holds.
- In this phase we'll learn how to loading and preprocessing earthquake data with Python and Matplotlib.
- The visual representation of data is more readily incorporated by human brains than the verbal representation of data.
- When something is pictured for us, we have an easier time understanding it.
- Datasets are the raw collection of information about a particular topic. In this article, we have an earthquake dataset that is in the form of a CSV file.
- We need to loading and preprocessing the dataset to know the trends and patterns in the dataset so that we can make a prediction of what could happen in the future.
- For example, in this phase we will be using an earthquake dataset, using matplotlib we will be loading and preprocessing the data to know the pattern that what was the intensity of the earthquake in previous years, and then we can predict the intensity of future earthquakes.
- For loading and preprocessing the dataset we use a Python library called Matplotlib. We will discuss in detail what matplotlib is and how it is used to loading and preprocessing the dataset.

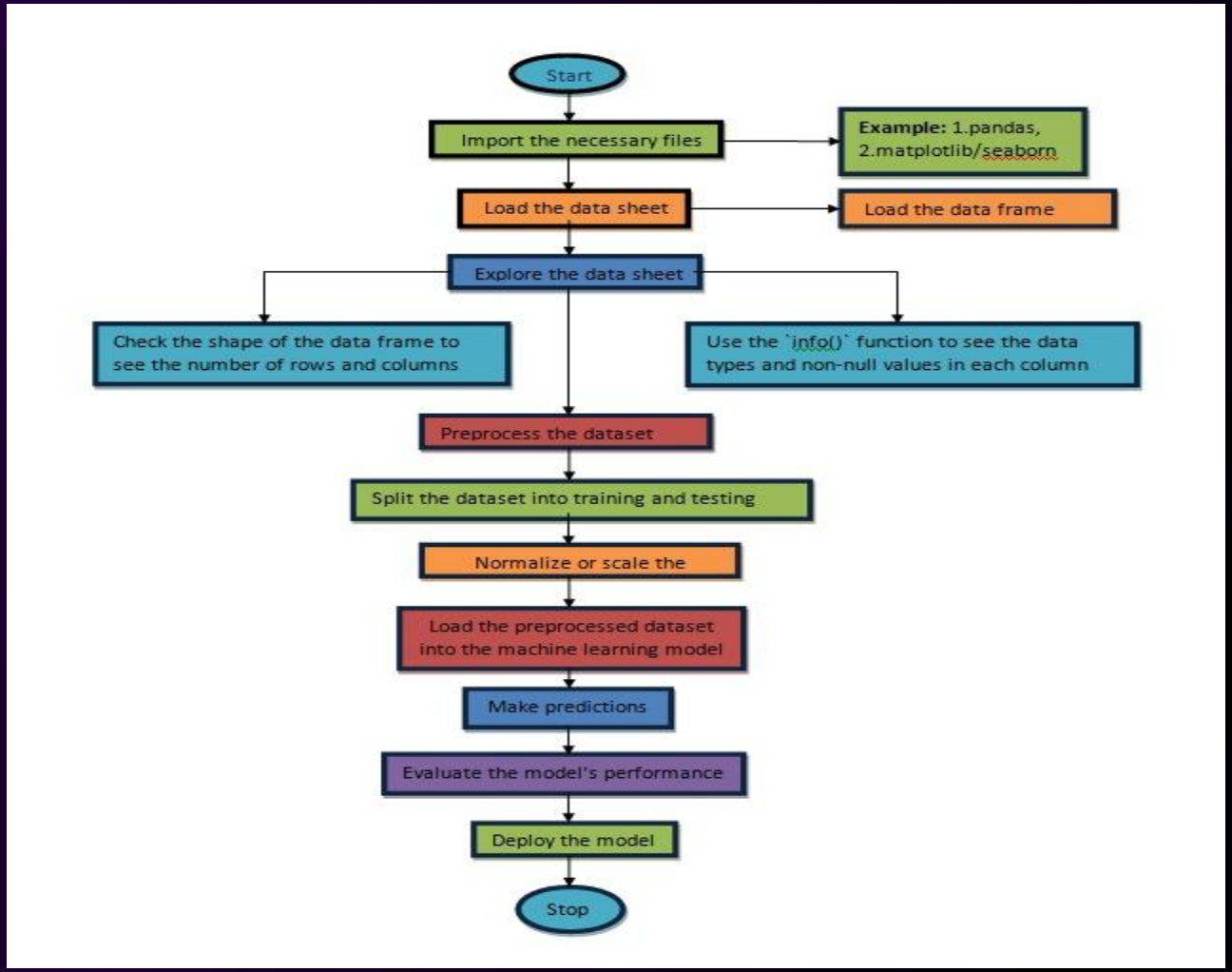
LOADING AND PREPROCESSING THE DATASET:-

- Earthquake prediction is a crucial task that requires preprocessing and loading the dataset to train machine learning models.
- The dataset can be downloaded from various sources such as Kaggle, GitHub, or other online repositories. Once the dataset is downloaded, the necessary libraries such as pandas, numpy, matplotlib, and seaborn need to be imported to preprocess and visualize the data.
- The dataset can be loaded into a pandas dataframe using the `read_csv()` function. Exploring the dataset using functions such as `head()`, `tail()`, `info()`, and `describe()` can provide a summary of the dataset.
  Cleaning the dataset by removing any missing or duplicate values, and converting the data types of the columns if necessary is essential.
- Splitting the dataset into training and testing sets using functions such as `train_test_split()` is necessary to train and evaluate the machine learning models.
- Normalizing the data using functions such as `StandardScaler()` or `MinMaxScaler()` ensures that all the features are on the same scale.
- Finally, the preprocessed data can be saved to a file for future use using functions such as `to_csv()`. Machine learning has the ability to advance our knowledge of earthquakes and enable more accurate forecasting and catastrophe response. By training the neural network on historical earthquake data, it can acquire the ability to identify precursor signals and patterns that indicate the probability of an upcoming earthquake.
- The random forest method performs best in classifying large earthquake occurrences, while the LSTM method provides a rough estimation of the earthquake.
- The best algorithm was taken into consideration after a review of a number of attributes. The model based on a graph convolutional neural network with batch normalization and attention mechanism techniques can successfully predict earthquakes from seismic data.

Steps to document the preprocessing and loading of the dataset for earthquake prediction using Python :-

- **Import the necessary libraries**: - Pandas: This library helps to load the data frame in a 2D array format and perform loading and preprocessing the dataset - Matplotlib/Seaborn: These libraries are used to draw dataset .
- **Load the dataset**: - Use the Pandas library to load the earthquake dataset into a data frame.
- **Explore the dataset**: - Check the shape of the data frame to see the number of rows and columns. - Use the `info()` function to see the data types and non-null values in each column. - Use the `describe()` function to get descriptive statistical measures of the dataset.
- **Preprocess the dataset**:- Handle missing values, outliers, and other data quality issues as needed. - Perform feature engineering, such as creating new variables or transforming existing ones, to improve the predictive power of the dataset.
- **Split the dataset into training and testing sets**:- Divide the dataset into a training set, which will be used to train the machine learning model, and a testing set, which will be used to evaluate the model's performance.
- **Normalize or scale the dataset**:- Depending on the machine learning algorithm used, it may be necessary to normalize or scale the dataset to ensure that all features have a similar range of values.
- **Load the preprocessed dataset into the machine learning model**:- Train the model using the training set and evaluate its performance using the testing set.
- **Make predictions**:- Use the trained model to make predictions on new, unseen data.

FLOW CHART:-


Start
Import the necessary files
Example: 1.pandas, 2.matplotlib/seaborn
Load the data sheet
Load the data frame
Explore the data sheet
Check the shape of the data frame to see the number of rows and columns
Use the `info()` function to see the data types and non-null values in each column
Preprocess the dataset
Split the dataset into training and testing
Normalize or scale the
Load the preprocessed dataset into the machine learning model
Make predictions
Evaluate the model's performance
Deploy the model
Stop

Python code for loading and preprocessing the dataset :-[sample code]

CODE :-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv('earthquake_data.csv')
# Remove duplicates
df.drop_duplicates(inplace=True)
# Handle missing values
df.fillna(method='ffill', inplace=True)
# Convert data types
df['Date'] = pd.to_datetime(df['Date'])
model_selection import train_test_split
X = df.drop('Magnitude', axis=1)
y = df['Magnitude']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
```

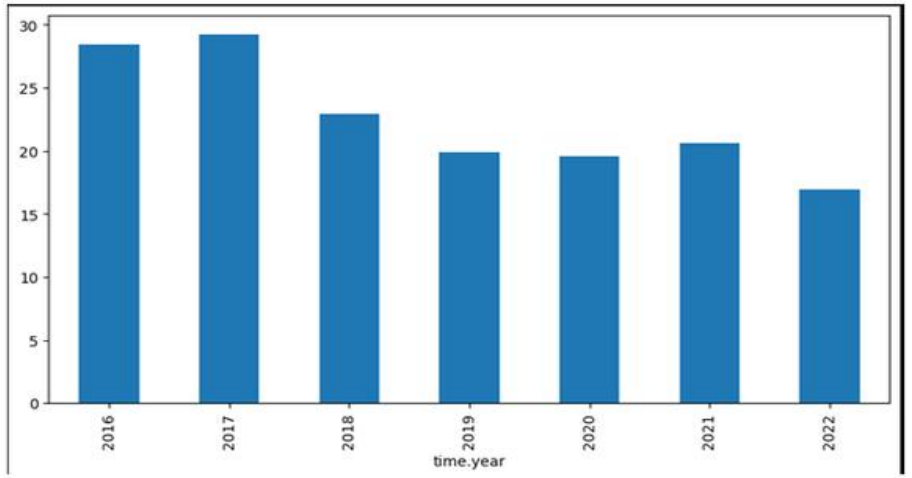
```
# Create models
lr = LinearRegression()
dt = DecisionTreeRegressor()
knn = KNeighborsRegressor()
# Train models
lr.fit(X_train, y_train)
dt.fit(X_train, y_train)
knn.fit(X_train, y_train)
from sklearn.metrics import mean_squared_error,
r2_score
# Make predictions
y_pred_lr = lr.predict(X_test)
y_pred_dt = dt.predict(X_test)
y_pred_knn = knn.predict(X_test)
# Evaluate performance
print('Linear Regression MSE:',
mean_squared_error(y_test, y_pred_lr))
print('Decision Tree MSE:',
mean_squared_error(y_test, y_pred_dt))
```

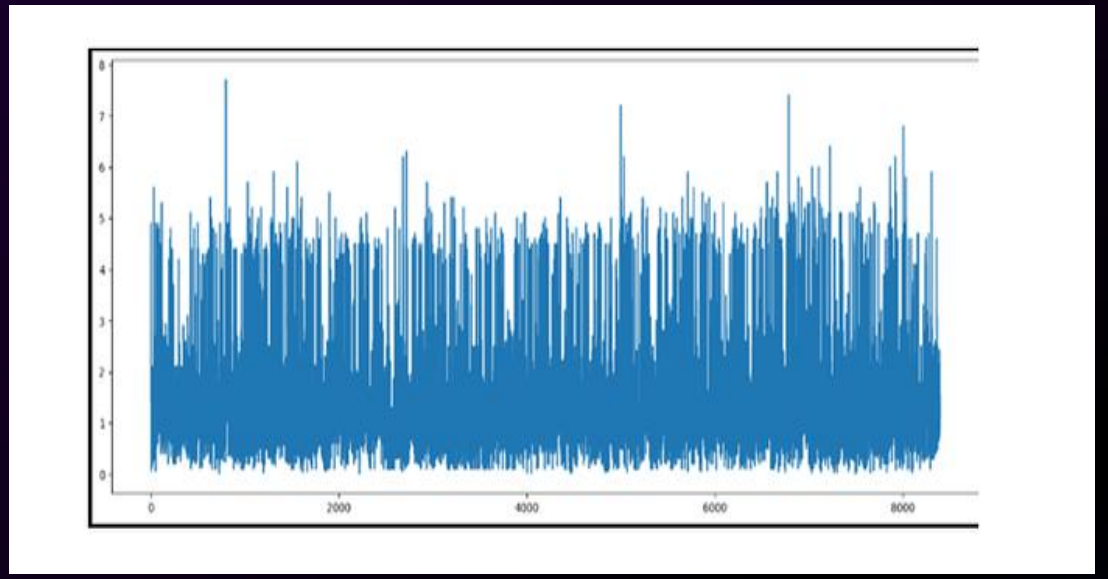
```
print('KNN MSE:', mean_squared_error(y_test, y_pred_knn))
print('Linear Regression R-squared:', r2_score(y_test, y_pred_lr))
print('Decision Tree R-squared:', r2_score(y_test, y_pred_dt))
print('KNN R-squared:', r2_score(y_test, y_pred_knn))
import seaborn as sns
# preprocessing data
sns.scatterplot(x='Latitude', y='Longitude', hue='Magnitude', data=df)
# preprocessing results
plt.plot(y_test, label='Actual')
plt.plot(y_pred_lr, label='Linear Regression')
plt.plot(y_pred_dt, label='Decision Tree')
plt.plot(y_pred_knn, label='KNN')
plt.legend()
plt.show()
```

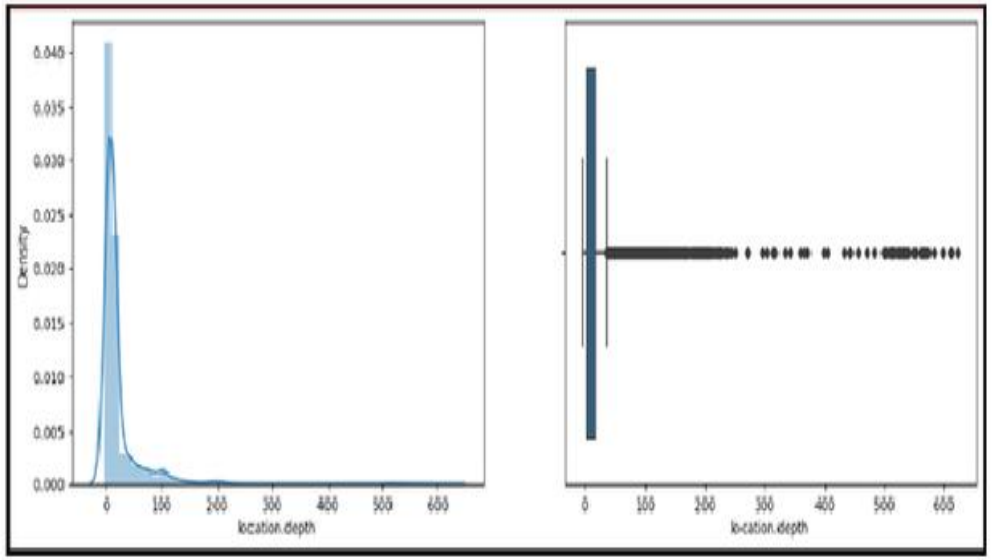
OUTPUT[SAMPLE] :-

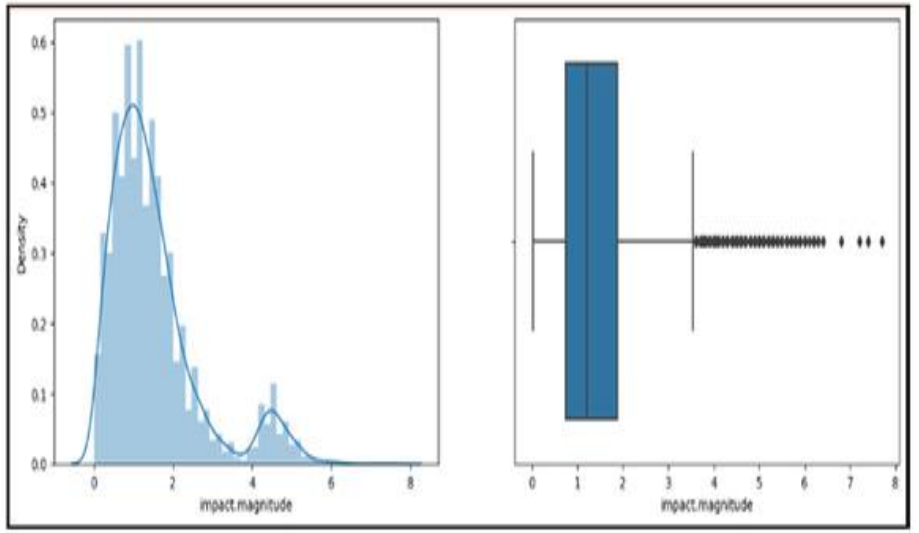
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8394 entries, 0 to 8393
Data columns (total 19 columns):
 #    Column                 Non-Null Count  Dtype
---   ------                 --------------  -----
 0    id                     8394 non-null   object
 1    impact.gap             8394 non-null   float64
 2    impact.magnitude       8394 non-null   float64
 3    impact.significance    8394 non-null   int64
 4    location.depth         8394 non-null   float64
 5    location.distance      8394 non-null   float64
 6    location.full          8394 non-null   object
 7    location.latitude      8394 non-null   float64
 8    location.longitude     8394 non-null   float64
 9    location.name          8394 non-null   object
 10   time.day               8394 non-null   int64
 11   time.epoch             8394 non-null   float64
 12   time.full              8394 non-null   object
 13   time.hour              8394 non-null   int64
 14   time.minute            8394 non-null   int64
 15   time.month             8394 non-null   int64
 16   time.second            8394 non-null   int64
 17   time.year              8394 non-null   int64
 18   location               8394 non-null   object
dtypes: float64(7), int64(7), object(5)
memory usage: 1.2+ MB
```

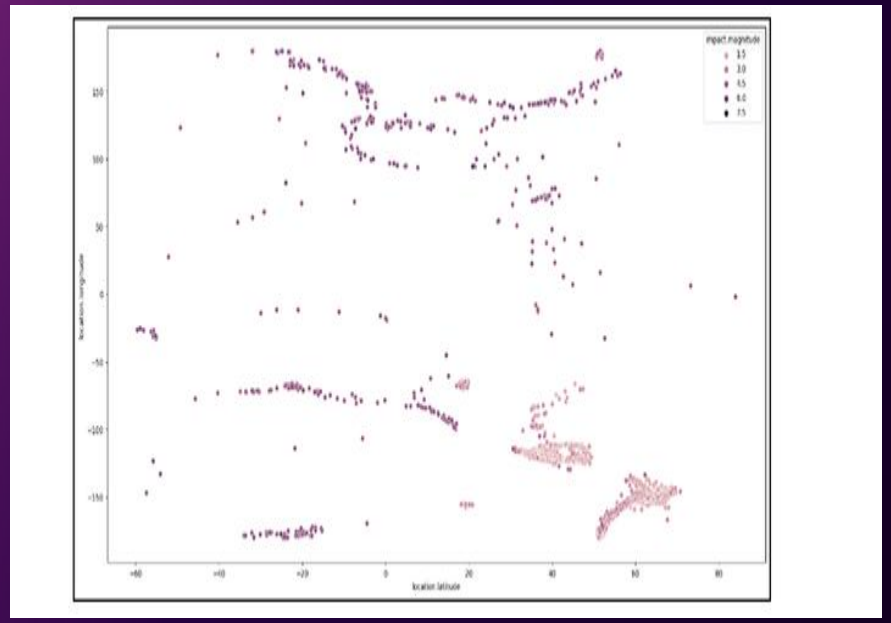
PIECHART[SAMPLE] :-

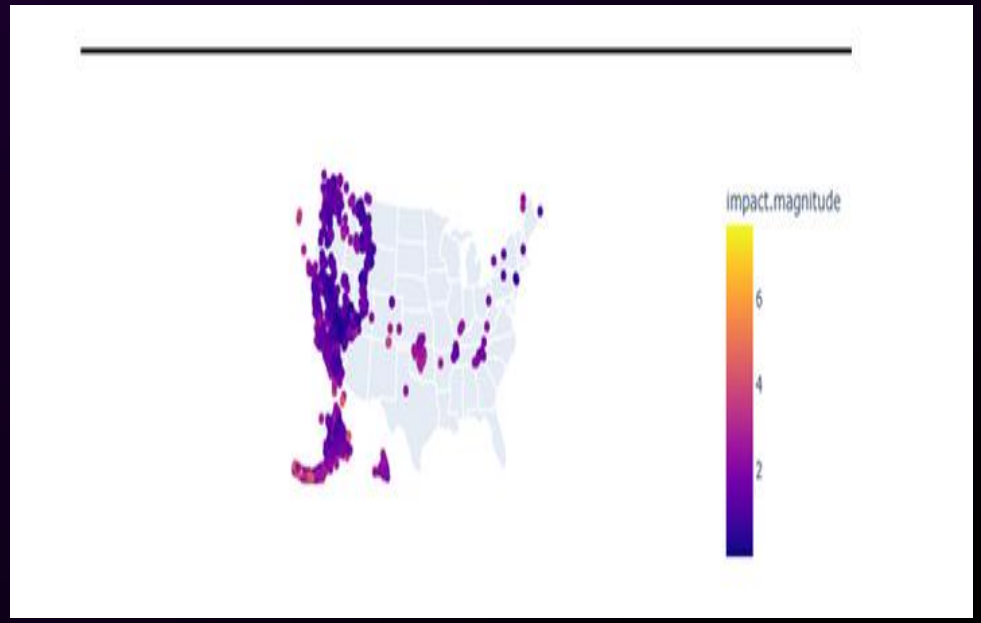


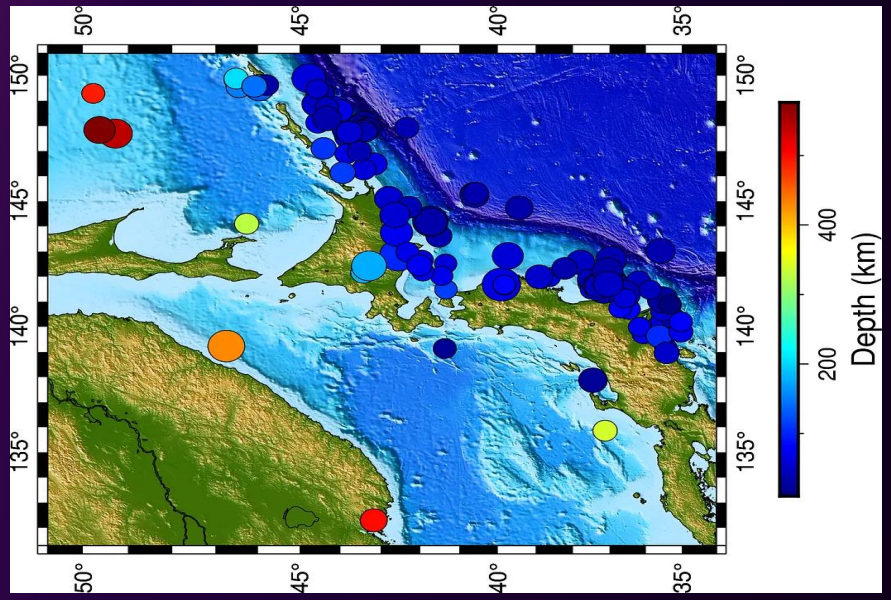


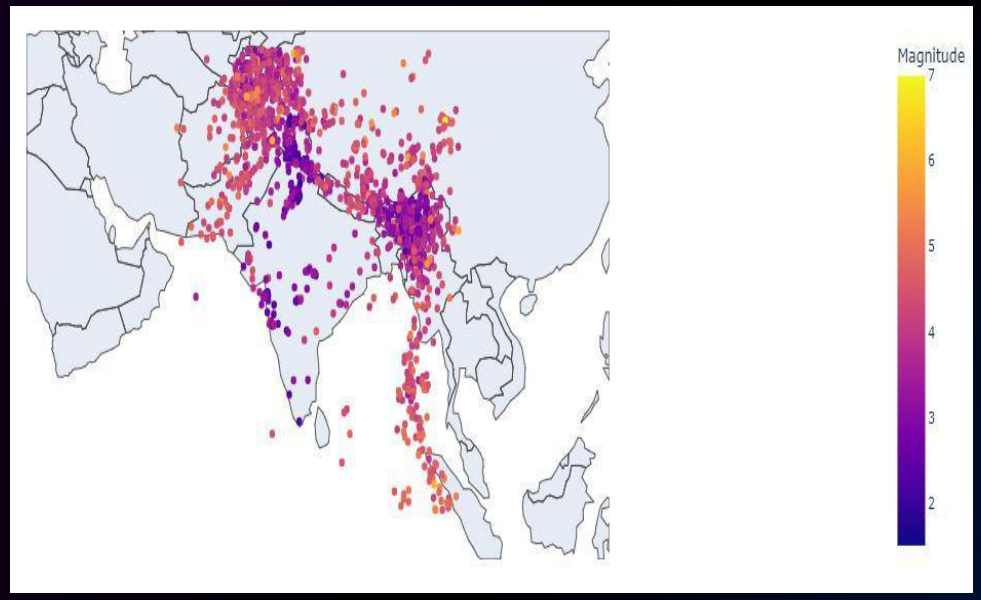




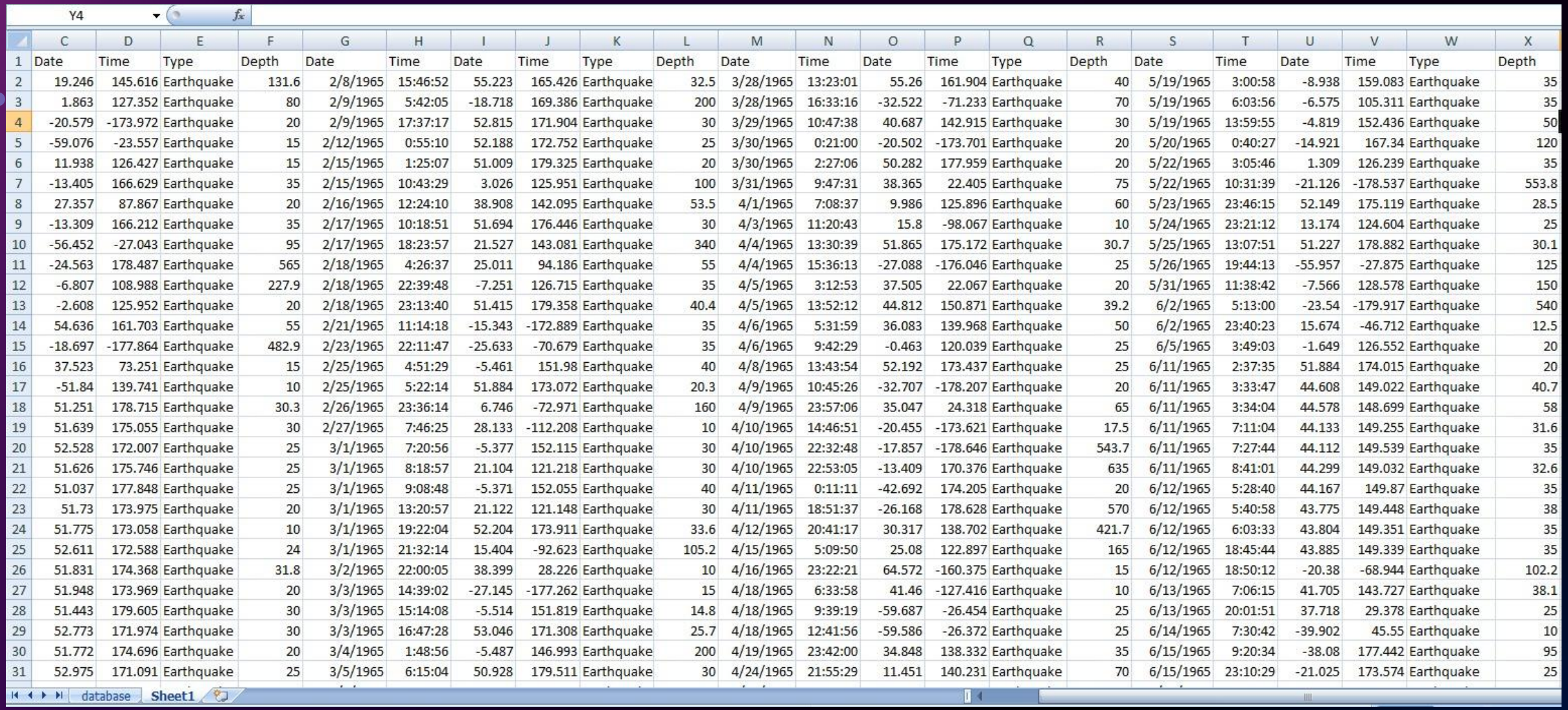



impact.magnitude
6
4
2


50°
45°
40°
35°
150°
145°
140°
135°
400
200
Depth (km)


Magnitude
7
6
5
4
3
2

DATASET:-



	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	Date	Time	Type	Depth	Date	Time	Date	Time	Type	Depth	Date	Time	Date	Time	Type	Depth	Date	Time	Date	Time	Type	Depth
2	19.246	145.616	Earthquake	131.6	2/8/1965	15:46:52	55.223	165.426	Earthquake	32.5	3/28/1965	13:23:01	55.26	161.904	Earthquake	40	5/19/1965	3:00:58	-8.938	159.083	Earthquake	35
3	1.863	127.352	Earthquake	80	2/9/1965	5:42:05	-18.718	169.386	Earthquake	200	3/28/1965	16:33:16	-32.522	-71.233	Earthquake	70	5/19/1965	6:03:56	-6.575	105.311	Earthquake	35
4	-20.579	-173.972	Earthquake	20	2/9/1965	17:37:17	52.815	171.904	Earthquake	30	3/29/1965	10:47:38	40.687	142.915	Earthquake	30	5/19/1965	13:59:55	-4.819	152.436	Earthquake	50
5	-59.076	-23.557	Earthquake	15	2/12/1965	0:55:10	52.188	172.752	Earthquake	25	3/30/1965	0:21:00	-20.502	-173.701	Earthquake	20	5/20/1965	0:40:27	-14.921	167.34	Earthquake	120
6	11.938	126.427	Earthquake	15	2/15/1965	1:25:07	51.009	179.325	Earthquake	20	3/30/1965	2:27:06	50.282	177.959	Earthquake	20	5/22/1965	3:05:46	1.309	126.239	Earthquake	35
7	-13.405	166.629	Earthquake	35	2/15/1965	10:43:29	3.026	125.951	Earthquake	100	3/31/1965	9:47:31	38.365	22.405	Earthquake	75	5/22/1965	10:31:39	-21.126	-178.537	Earthquake	553.8
8	27.357	87.867	Earthquake	20	2/16/1965	12:24:10	38.908	142.095	Earthquake	53.5	4/1/1965	7:08:37	9.986	125.896	Earthquake	60	5/23/1965	23:46:15	52.149	175.119	Earthquake	28.5
9	-13.309	166.212	Earthquake	35	2/17/1965	10:18:51	51.694	176.446	Earthquake	30	4/3/1965	11:20:43	15.8	-98.067	Earthquake	10	5/24/1965	23:21:12	13.174	124.604	Earthquake	25
10	-56.452	-27.043	Earthquake	95	2/17/1965	18:23:57	21.527	143.081	Earthquake	340	4/4/1965	13:30:39	51.865	175.172	Earthquake	30.7	5/25/1965	13:07:51	51.227	178.882	Earthquake	30.1
11	-24.563	178.487	Earthquake	565	2/18/1965	4:26:37	25.011	94.186	Earthquake	55	4/4/1965	15:36:13	-27.088	-176.046	Earthquake	25	5/26/1965	19:44:13	-55.957	-27.875	Earthquake	125
12	-6.807	108.988	Earthquake	227.9	2/18/1965	22:39:48	-7.251	126.715	Earthquake	35	4/5/1965	3:12:53	37.505	22.067	Earthquake	20	5/31/1965	11:38:42	-7.566	128.578	Earthquake	150
13	-2.608	125.952	Earthquake	20	2/18/1965	23:13:40	51.415	179.358	Earthquake	40.4	4/5/1965	13:52:12	44.812	150.871	Earthquake	39.2	6/2/1965	5:13:00	-23.54	-179.917	Earthquake	540
14	54.636	161.703	Earthquake	55	2/21/1965	11:14:18	-15.343	-172.889	Earthquake	35	4/6/1965	5:31:59	36.083	139.968	Earthquake	50	6/2/1965	23:40:23	15.674	-46.712	Earthquake	12.5
15	-18.697	-177.864	Earthquake	482.9	2/23/1965	22:11:47	-25.633	-70.679	Earthquake	35	4/6/1965	9:42:29	-0.463	120.039	Earthquake	25	6/5/1965	3:49:03	-1.649	126.552	Earthquake	20
16	37.523	73.251	Earthquake	15	2/25/1965	4:51:29	-5.461	151.98	Earthquake	40	4/8/1965	13:43:54	52.192	173.437	Earthquake	25	6/11/1965	2:37:35	51.884	174.015	Earthquake	20
17	-51.84	139.741	Earthquake	10	2/25/1965	5:22:14	51.884	173.072	Earthquake	20.3	4/9/1965	10:45:26	-32.707	-178.207	Earthquake	20	6/11/1965	3:33:47	44.608	149.022	Earthquake	40.7
18	51.251	178.715	Earthquake	30.3	2/26/1965	23:36:14	6.746	-72.971	Earthquake	160	4/9/1965	23:57:06	35.047	24.318	Earthquake	65	6/11/1965	3:34:04	44.578	148.699	Earthquake	58
19	51.639	175.055	Earthquake	30	2/27/1965	7:46:25	28.133	-112.208	Earthquake	10	4/10/1965	14:46:51	-20.455	-173.621	Earthquake	17.5	6/11/1965	7:11:04	44.133	149.255	Earthquake	31.6
20	52.528	172.007	Earthquake	25	3/1/1965	7:20:56	-5.377	152.115	Earthquake	30	4/10/1965	22:32:48	-17.857	-178.646	Earthquake	543.7	6/11/1965	7:27:44	44.112	149.539	Earthquake	35
21	51.626	175.746	Earthquake	25	3/1/1965	8:18:57	21.104	121.218	Earthquake	30	4/10/1965	22:53:05	-13.409	170.376	Earthquake	635	6/11/1965	8:41:01	44.299	149.032	Earthquake	32.6
22	51.037	177.848	Earthquake	25	3/1/1965	9:08:48	-5.371	152.055	Earthquake	40	4/11/1965	0:11:11	-42.692	174.205	Earthquake	20	6/12/1965	5:28:40	44.167	149.87	Earthquake	35
23	51.73	173.975	Earthquake	20	3/1/1965	13:20:57	21.122	121.148	Earthquake	30	4/11/1965	18:51:37	-26.168	178.628	Earthquake	570	6/12/1965	5:40:58	43.775	149.448	Earthquake	38
24	51.775	173.058	Earthquake	10	3/1/1965	19:22:04	52.204	173.911	Earthquake	33.6	4/12/1965	20:41:17	30.317	138.702	Earthquake	421.7	6/12/1965	6:03:33	43.804	149.351	Earthquake	35
25	52.611	172.588	Earthquake	24	3/1/1965	21:32:14	15.404	-92.623	Earthquake	105.2	4/15/1965	5:09:50	25.08	122.897	Earthquake	165	6/12/1965	18:45:44	43.885	149.339	Earthquake	35
26	51.831	174.368	Earthquake	31.8	3/2/1965	22:00:05	38.399	28.226	Earthquake	10	4/16/1965	23:22:21	64.572	-160.375	Earthquake	15	6/12/1965	18:50:12	-20.38	-68.944	Earthquake	102.2
27	51.948	173.969	Earthquake	20	3/3/1965	14:39:02	-27.145	-177.262	Earthquake	15	4/18/1965	6:33:58	41.46	-127.416	Earthquake	10	6/13/1965	7:06:15	41.705	143.727	Earthquake	38.1
28	51.443	179.605	Earthquake	30	3/3/1965	15:14:08	-5.514	151.819	Earthquake	14.8	4/18/1965	9:39:19	-59.687	-26.454	Earthquake	25	6/13/1965	20:01:51	37.718	29.378	Earthquake	25
29	52.773	171.974	Earthquake	30	3/3/1965	16:47:28	53.046	171.308	Earthquake	25.7	4/18/1965	12:41:56	-59.586	-26.372	Earthquake	25	6/14/1965	7:30:42	-39.902	45.55	Earthquake	10
30	51.772	174.696	Earthquake	20	3/4/1965	1:48:56	-5.487	146.993	Earthquake	200	4/19/1965	23:42:00	34.848	138.332	Earthquake	35	6/15/1965	9:20:34	-38.08	177.442	Earthquake	95
31	52.975	171.091	Earthquake	25	3/5/1965	6:15:04	50.928	179.511	Earthquake	30	4/24/1965	21:55:29	11.451	140.231	Earthquake	70	6/15/1965	23:10:29	-21.025	173.574	Earthquake	25

REFERENCE: https://www.kaggle.com/datasets/usgs/earthquake-database

CONCLUSION :-

- **Preprocessing the dataset:** This step is crucial for cleaning and transforming the raw data into a format suitable for analysis and modeling. Preprocessing techniques may include handling missing values, normalizing or scaling features, and encoding categorical variables

- **Splitting the dataset:** After preprocessing, the dataset is typically divided into training and testing sets. The training set is used to train the machine learning models, while the testing set is used to evaluate their performance.

- **Building machine learning models:** Various machine learning algorithms can be applied to earthquake prediction, such as linear regression, decision trees, and k-nearest neighbors. These models are trained on the training set and then used to make predictions on the testing set.

- **Further analysis:** Once the initial models are built and evaluated, further analysis can be performed to identify the main factors that contribute to the occurrence of earthquakes. This may involve feature selection techniques or more advanced machine learning algorithms.

- **Consider other research:** Other research papers and articles can provide insights into the latest advancements in earthquake prediction using machine learning.

THANK YOU