EARTHQUAKE PREDICTION MODEL USING PYTHON

PROJECT DONE BY

Team leader:

PRIYA GUPTA.K (922321106301)

Team members:

ABITHA.J (922321106003)
NITHYA.A (922321106020)
SHRISHTITHA.S (922321106034)
SUBHIKSHA.V (922321106038)
SWETHA.A.B (922321106040)

PHASE 3: DEVELOPMENT PART-1

TOPIC: DATA LOADING AND DATA PREPROCESSING

OBJECTIVE:

This introduction will guide you through the initial steps of the process. We'll explore how to import essential libraries, load the Earthquake dataset, and perform critical preprocessing steps. Collect and preprocess historical earthquake data, including information on location, depth, magnitude, and time. Gather relevant geological and geophysical data, such as fault lines, tectonic plate boundaries, and soil characteristics Data preprocessing is crucial as it helps clean, format, and prepare the data for further analysis. This includes handling missing values, encoding categorical variables, and ensuring that the data is appropriately scaled.

Dataset Link: **https://www.kaggle.com/datasets/usgs/earthquake-database**

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type
0	01/02/1965	13:44:18	19.2460	145.6160	Earthquake	131.60	NaN	NaN	6.0	MW
1	01/04/1965	11:29:49	1.8630	127.3520	Earthquake	80.00	NaN	NaN	5.8	MW
2	01/05/1965	18:05:58	-20.5790	-173.9720	Earthquake	20.00	NaN	NaN	6.2	MW
3	01/08/1965	18:49:43	-59.0760	-23.5570	Earthquake	15.00	NaN	NaN	5.8	MW
4	01/09/1965	13:32:50	11.9380	126.4270	Earthquake	15.00	NaN	NaN	5.8	MW
...	...	...	...	...	...	...	...	...	...	...
23407	12/28/2016	08:22:12	38.3917	-118.8941	Earthquake	12.30	1.2	40.0	5.6	ML
23408	12/28/2016	09:13:47	38.3777	-118.8957	Earthquake	8.80	2.0	33.0	5.5	ML
23409	12/28/2016	12:38:51	36.9179	140.4262	Earthquake	10.00	1.8	NaN	5.9	MWW
23410	12/29/2016	22:30:19	-9.0283	118.6639	Earthquake	79.00	1.8	NaN	6.3	MWW
23411	12/30/2016	20:08:28	37.3973	141.4103	Earthquake	11.94	2.2	NaN	5.5	MB

DATA LOADING:

Load your dataset into a Pandas DataFrame. You can typically find Earthquake datasets in CSV format, but you can adapt this code to other formats as needed.

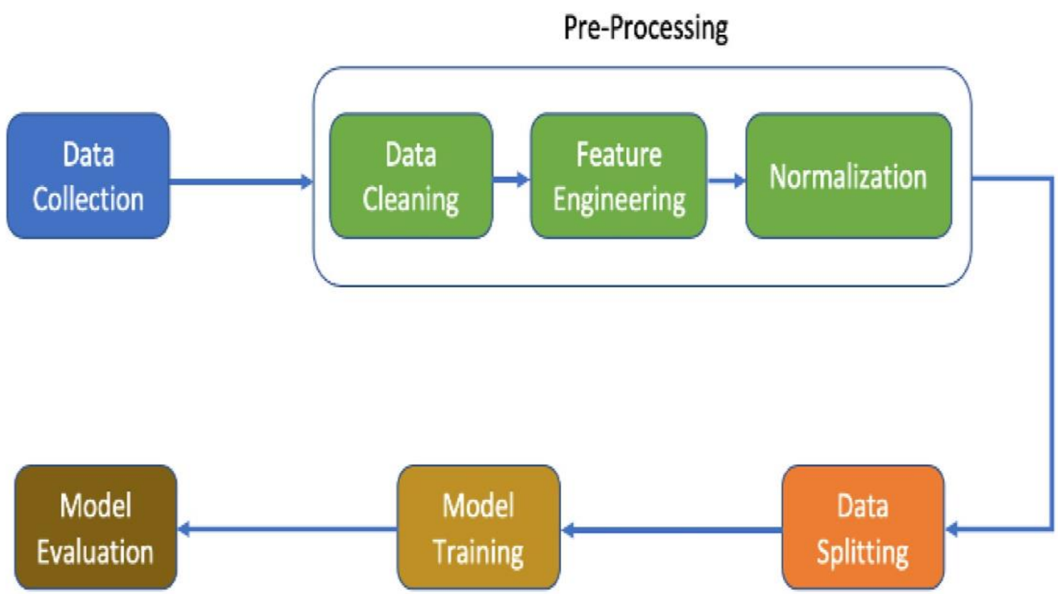
Here, basic example of how to load earthquake data:

```
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
# Import requests library to fetch earthquake data from USGS API
    import requests
# Define the API endpoint for USGS earthquake data
    usgs_api_url = "https://earthquake.usgs.gov/fdsnws/event/1/query"
# Define parameters for the earthquake query
    params = {
```

```
    "format": "geojson",
    "starttime": "2000-01-01",
    "endtime": "2021-12-31",
    "minmagnitude": 5.0,
    "orderby": "time",
  }
# Send a GET request to the USGS API and load the data into a Pandas DataFrame
  response = requests.get(usgs_api_url, params=params)
  data = response.json()
# Extract earthquake features and create a DataFrame
  earthquake_data = []
  for feature in data["features"]:
    properties = feature["properties"]
    coordinates = feature["geometry"]["coordinates"]
    earthquake_data.append({
      "Date": properties["time"],
      "Magnitude": properties["mag"],
      "Latitude": coordinates[1],
   "Longitude": coordinates[0],
      "Depth (km)": coordinates[2],
    })
  earthquake_df = pd.DataFrame(earthquake_data)
# Display the first few rows of the DataFrame
  print(earthquake_df.head())
# Visualize earthquake magnitudes
  plt.figure(figsize=(10, 6))
  plt.hist(earthquake_df["Magnitude"], bins=20, edgecolor="k")
  plt.title("Earthquake Magnitudes")
  plt.xlabel("Magnitude")
  plt.ylabel("Count")
  plt.show()
```

DATA PREPROCESSING:

Data preprocessing is a process of preparing the raw data and makingit suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.



It involves below steps:

I. Get the Dataset

- To create a machine learning model, the first thing we required is a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as the **dataset**.
- To use the dataset in our code, we usually put it into a CSV file. However, sometimes, we may also need to use an HTML or xlsx file.

II. Importing Libraries

In order to perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing,

which are:

- Numpy
- pandas
- matplotlib
- seaborn

III. Handling Missing data:

The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.

Ways to handle missing data:

- By deleting the particular row
- By calculating the mean

IV. Splitting the Dataset into the Training set and Test set

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of ourmachine learning model.

Training Set: A subset of dataset to train the machine learningmodel, and we already know the output.

Test set: A subset of dataset to test the machine learning model,and by using the test set, model predicts the output

V. Feature Scaling

- Feature scaling is the final step of data preprocessing in machinelearning.
- It is a technique to standardize the independent variables of thedataset in a specific range.
- In feature scaling, we put our variables in the same range and in thesame scale so that no any variable dominate the other variable.

VI. Visualization

- Visualization or visualisation (see spelling differences) is any technique for creating images, diagrams, or animations to communicate a message.
- Visualization through visual imagery has been an effective way to communicate both abstract and concrete ideas since the dawn of humanity.
- Visualization is a crucial aspect of earthquake prediction models. It helps you understand the data, discover patterns, and communicate your findings effectively. Here are some types of visualizations and their content that can be helpful in the context of earthquake prediction.

PYTHON PROGRAM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pltimport seaborn
as sns
from sklearn.preprocessing import StandardScaler from
sklearn.model_selection import train_test_split
import tensorflow as tf
data =pd.read_csv('C:/earthquake-database/database.csv')data
```

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	...
0	01/02/1965	13:44:18	19.2460	145.6160	Earthquake	131.60	NaN	NaN	6.0	MW	...
1	01/04/1965	11:29:49	1.8630	127.3520	Earthquake	80.00	NaN	NaN	5.8	MW	...
2	01/05/1965	18:05:58	-20.5790	-173.9720	Earthquake	20.00	NaN	NaN	6.2	MW	...
3	01/08/1965	18:49:43	-59.0760	-23.5570	Earthquake	15.00	NaN	NaN	5.8	MW	...
4	01/09/1965	13:32:50	11.9380	126.4270	Earthquake	15.00	NaN	NaN	5.8	MW	...
...	...	...	...	...	...	...	...	...	...	...	...
23407	12/28/2016	08:22:12	38.3917	-118.8941	Earthquake	12.30	1.2	40.0	5.6	ML	...
23408	12/28/2016	09:13:47	38.3777	-118.8957	Earthquake	8.80	2.0	33.0	5.5	ML	...
23409	12/28/2016	12:38:51	36.9179	140.4262	Earthquake	10.00	1.8	NaN	5.9	MWW	...
23410	12/29/2016	22:30:19	-9.0283	118.6639	Earthquake	79.00	1.8	NaN	6.3	MWW	...
23411	12/30/2016	20:08:28	37.3973	141.4103	Earthquake	11.94	2.2	NaN	5.5	MB	...

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23412 entries, 0 to 23411
Data columns (total 21 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Date                        23412 non-null  object
 1   Time                        23412 non-null  object
 2   Latitude                    23412 non-null  float64
 3   Longitude                   23412 non-null  float64
 4   Type                        23412 non-null  object
 5   Depth                       23412 non-null  float64
 6   Depth Error                 4461 non-null   float64
 7   Depth Seismic Stations      7097 non-null   float64
 8   Magnitude                   23412 non-null  float64
 9   Magnitude Type              23409 non-null  object
 10  Magnitude Error             327 non-null    float64
 11  Magnitude Seismic Stations  2564 non-null   float64
 12  Azimuthal Gap               7299 non-null   float64
 13  Horizontal Distance         1604 non-null   float64
 14  Horizontal Error            1156 non-null   float64
 15  Root Mean Square            17352 non-null  float64
 16  ID                          23412 non-null  object
 17  Source                      23412 non-null  object
 18  Location Source             23412 non-null  object
 19  Magnitude Source            23412 non-null  object
 20  Status                      23412 non-null  object
dtypes: float64(12), object(9)
memory usage: 3.8+ MB
```

```
data = data.drop('ID', axis=1)
```

```
data.isna().sum()
```

```
6]:
Date                              0
Time                              0
Latitude                          0
Longitude                         0
Type                              0
Depth                             0
Depth Error                   18951
Depth Seismic Stations        16315
Magnitude                         0
Magnitude Type                    3
Magnitude Error               23085
Magnitude Seismic Stations    20848
Azimuthal Gap                 16113
Horizontal Distance           21808
Horizontal Error              22256
Root Mean Square               6060
Source                            0
Location Source                   0
Magnitude Source                  0
Status                            0
dtype: int64
```

```
null_columns = data.loc[:, data.isna().sum() > 0.66 *data.shape[0]].columns
data.isna().sum()
```

```
Date                    0
Time                    0
Latitude                0
Longitude               0
Type                    0
Depth                   0
Magnitude               0
Magnitude Type          3
Root Mean Square     6060
Source                  0
Location Source         0
Magnitude Source        0
Status                  0
dtype: int64
```

```
data['Root Mean Square'] = data['Root Mean
Square'].fillna(data['Root Mean Square'].mean())data =
data.dropna(axis=0).reset_index(drop=True)data.isna().sum().sum()

0

y = data.loc[:, 'Status']

X = data.drop('Status', axis=1)


scaler = StandardScaler()

X = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y,train_size=0.7, random_state=56)

Split X.shape

(23406, 104)
```

```
y.mean()

0.88737930445185

inputs = tf.keras.Input(shape=(104,))

x = tf.keras.layers.Dense(64, activation='relu')(inputs)x =

tf.keras.layers.Dense(64, activation='relu')(x)

outputs = tf.keras.layers.Dense(1, activation='sigmoid')(x)model =

tf.keras.Model(inputs, outputs)

model.compile( optimizer='adam',

        loss='binary_crossentropy',

        metrics=[tf.keras.metrics.AUC(name='auc')]

)

batch_size = 32

epochs = 30

history = model.fit(

        X_train, y_train,

        validation_split=0.2, batch_size=batch_size,

        epochs=epochs,

        callbacks=[tf.keras.callbacks.ReduceLROnPlateau()],verbose=0

)
```

```
)

data['Month'] = data['Date'].apply(lambda x: x[0:2])

data['Year'] = data['Date'].apply(lambda x: x[-4:])data = data.drop('Date',

axis=1)

data['Month'] = data['Month'].astype(np.int)data[data['Year'].str.contains('Z')]
```

	Time	Latitude	Longitude	Type	Depth	Magnitude	Magnitude Type	Root Mean Square	Source	Location Source	Mag Sou
3378	1975-02-23T02:58:41.000Z	8.017	124.075	Earthquake	623.0	5.6	MB	1.022784	US	US	US
7510	1985-04-28T02:53:41.530Z	-32.998	-71.766	Earthquake	33.0	5.6	MW	1.300000	US	US	HR
20647	2011-03-13T02:23:34.520Z	36.344	142.344	Earthquake	10.1	5.8	MWC	1.060000	US	US	GC

```
invalid_year_indices =

data[data['Year'].str.contains('Z')].indexdata =

data.drop(invalid_year_indices,

axis=0).reset_index(drop=True) data['Year'] =

data['Year'].astype(np.int)

data['Hour'] = data['Time'].apply(lambda x: np.int(x[0:2]))data = data.drop('Time',

axis=1)

data
```

	Latitude	Longitude	Type	Depth	Magnitude	Magnitude Type	Root Mean Square	Source	Location Source	Magnitude Source	Status
0	19.2460	145.6160	Earthquake	131.60	6.0	MW	1.022784	ISCGEM	ISCGEM	ISCGEM	Autom
1	1.8630	127.3520	Earthquake	80.00	5.8	MW	1.022784	ISCGEM	ISCGEM	ISCGEM	Autom
2	-20.5790	-173.9720	Earthquake	20.00	6.2	MW	1.022784	ISCGEM	ISCGEM	ISCGEM	Autom
3	-59.0760	-23.5570	Earthquake	15.00	5.8	MW	1.022784	ISCGEM	ISCGEM	ISCGEM	Autom
4	11.9380	126.4270	Earthquake	15.00	5.8	MW	1.022784	ISCGEM	ISCGEM	ISCGEM	Autom
...	...	...	...	...	...	...	...	...	...	...	...
23401	38.3917	-118.8941	Earthquake	12.30	5.6	ML	0.189800	NN	NN	NN	Review
23402	38.3777	-118.8957	Earthquake	8.80	5.5	ML	0.218700	NN	NN	NN	Review
23403	36.9179	140.4262	Earthquake	10.00	5.9	MWW	1.520000	US	US	US	Review
23404	-9.0283	118.6639	Earthquake	79.00	6.3	MWW	1.430000	US	US	US	Review
23405	37.3973	141.4103	Earthquake	11.94	5.5	MB	0.910000	US	US	US	Review

23406 rows × 14 columns

data['Status'].unique()

array(['Automatic', 'Reviewed'], dtype=object) data['Status'] =

data['Status'].apply(lambda x: 1 if x =='Reviewed' else 0)

1

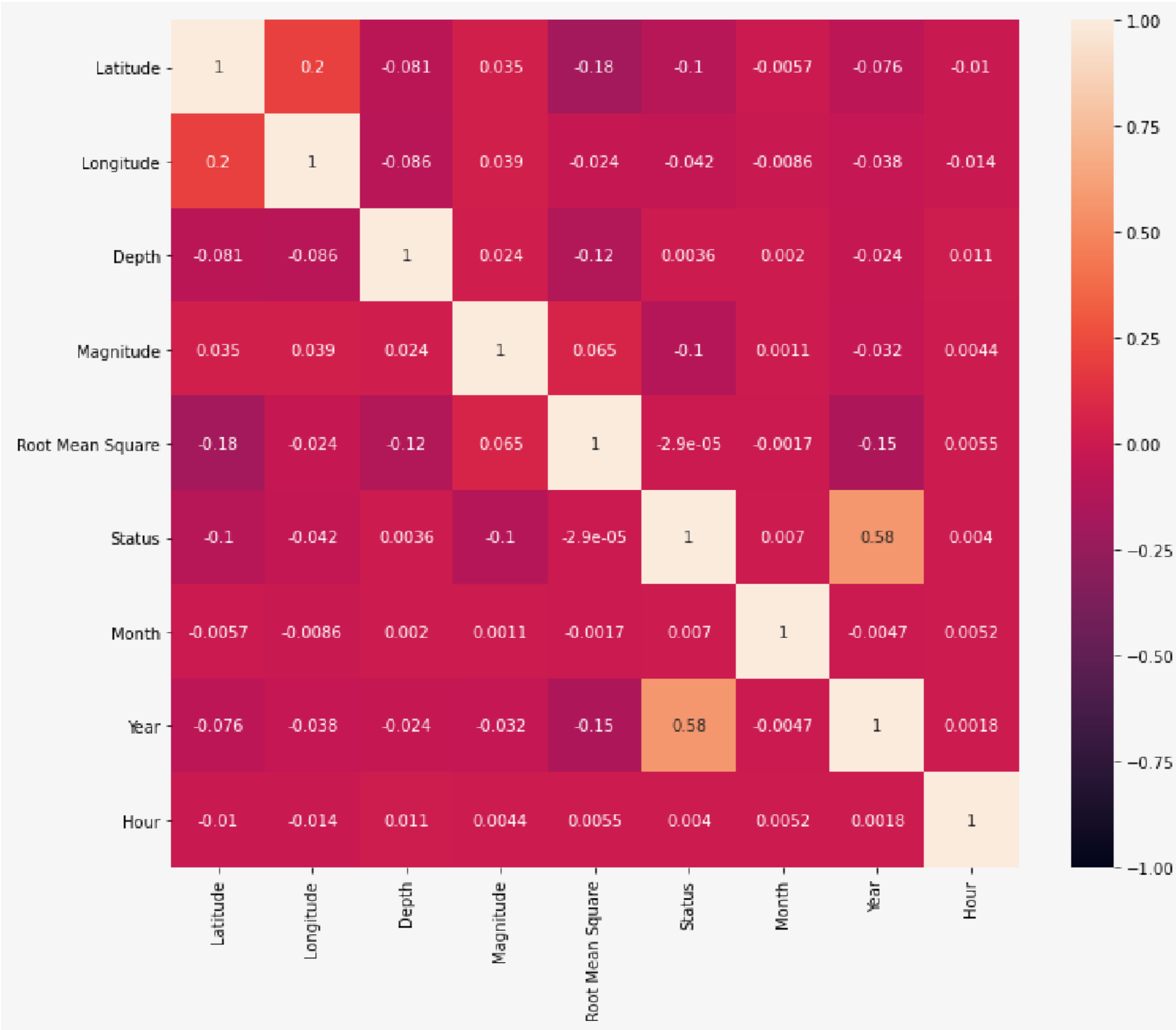
numeric_columns = [column for column **in** data.columns ifdata.dtypes[column]

!= 'object']

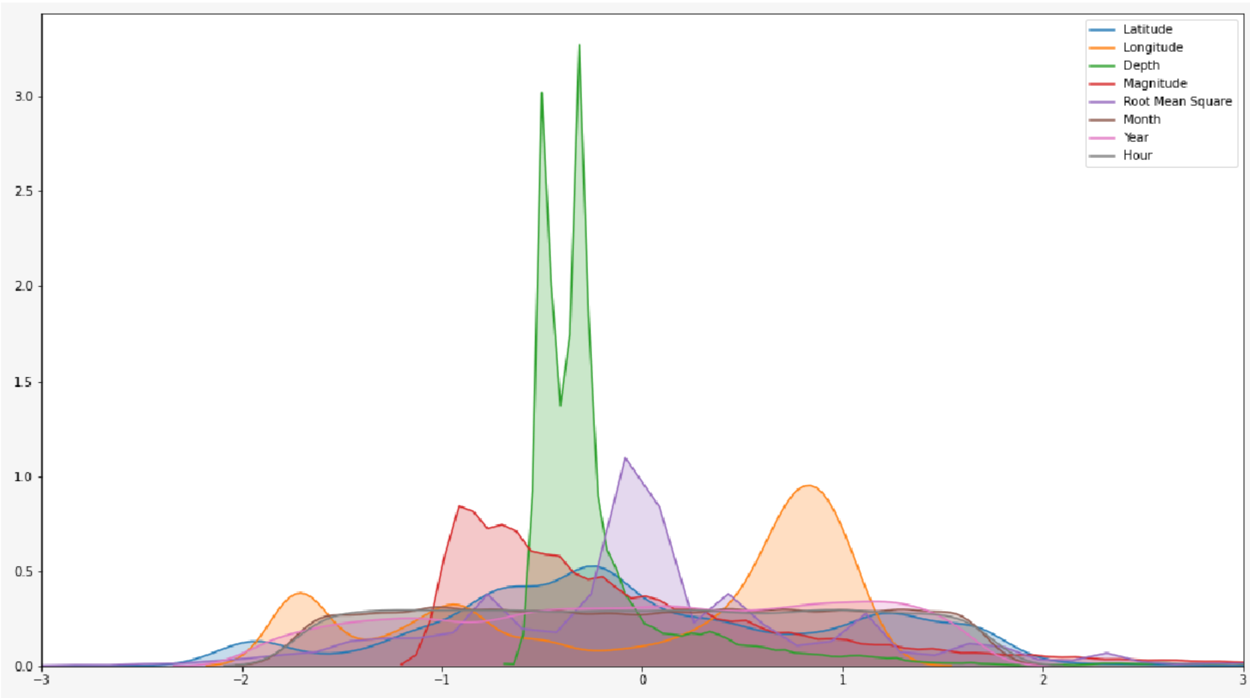
corr = data[numeric_columns].corr()plt.figure(figsize=(12, 10))

sns.heatmap(corr, annot=True, vmin=-1.0, vmax=1.0)

plt.show()




```
numeric_columns.remove('Status')scaler =
    StandardScaler()
standardized_df = pd.DataFrame(scaler.fit_transform(data[numeric_columns].copy()),
columns=numeric_columns)
plt.figure(figsize=(18, 10)) for column in
numeric_columns:
        sns.kdeplot(standardized_df[column], shade=True)
```



CONCLUSION:

- Data preprocessing emerged as a pivotal aspect of this process. Itinvolves cleaning, transforming, and refining the dataset to ensure_that it aligns with the requirements of machine learningalgorithms.
- With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a earthquake prediction model.