IBM NaanMudhalvan

ARTIFICIAL INTELLIGENCE

Project Title : Earthquake Prediction Using Python

Phase 3 : Development Part – 1

Begin building the earthquake prediction model by loading and preprocessing the dataset

Workbook Link : Google Colab

INTRODUCTION

This documentation is a guide to the preprocessing steps essential for constructing an earthquake prediction model. It covers data loading, cleaning, and exploratory analysis, providing transparency in the model-building process. The document emphasizes the rationale behind decisions, addressing challenges and nuances encountered. With a structured approach, it guides readers through feature engineering, transformations, and the crucial train-test split. Code snippets, visualizations, and examples facilitate understanding and reproducibility. Tailored for a diverse audience, from data scientists to enthusiasts, it highlights the significance of meticulous preprocessing in seismic prediction. The documentation's scope extends beyond replication, aiming to deepen comprehension of machine learning methodologies in earthquake forecasting. In 10 lines, it invites readers to explore the intricacies of preparing data for the vital task of earthquake prediction.

DATA LOADING

Data loading is the inaugural step in machine learning, essential for acquiring datasets that fuel model development. Identifying the data source, whether it be CSV files, databases, or APIs, dictates the loading approach. By integrating libraries like pandas, the process is streamlined, allowing users to efficiently manipulate and analyze data. The accompanying code snippets in the documentation showcase the programmatic loading of datasets, ensuring accessibility and ease of understanding. Versatility is emphasized, addressing various data formats such as CSV, Excel, JSON, or databases, providing adaptability to diverse structures. Robust data loading involves error handling, anticipating and managing issues like missing values or corrupted data.

The documentation also offers a glimpse of the loaded data, aiding users in comprehending its structure and content. Early data cleaning initiatives may be embedded during loading, tackling issues like missing values or inconsistent formatting. Emphasizing reproducibility, the documentation guides users on how to load the data with specific parameters for consistent results. Ultimately, data loading establishes the groundwork, connecting the acquired datasets to the subsequent stages of model training in the machine learning workflow.

PREPROCESSING

Preprocessing is a pivotal stage in machine learning workflows, acting as the foundation for robust model development. It encompasses several critical steps, beginning with the loading of raw data from diverse sources, such as CSV files or databases. The process involves thorough data cleaning, addressing issues like missing values, outliers, and duplicates to ensure the quality and reliability of the dataset. Exploratory Data Analysis (EDA) is employed to gain insights into the dataset's distribution, relationships, and potential patterns, guiding subsequent preprocessing decisions. Feature engineering follows, where new features are created or existing ones are transformed to enhance the model's understanding of underlying patterns. Data normalization and scaling are crucial for ensuring that features are on a consistent scale, preventing any particular feature from dominating the model training process. Categorical variables are appropriately encoded to numerical formats, facilitating their integration into machine learning models. The dataset is then split into training and testing sets to assess the model's generalization performance accurately. Throughout this process, documentation and inline comments are incorporated, ensuring transparency and reproducibility in the preprocessing pipeline. This meticulous preprocessing paves the way for effective model training, contributing significantly to the model's overall predictive accuracy.

PROGRAM :

```python
# Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

import tensorflow as tf

# Reading the dataset from the specified location
data = pd.read_csv('database.csv')

# Displaying the loaded dataset
data

# Providing information about the dataset,
including data types and missing values
data.info()
```

```
# Dropping the 'ID' column from the dataset
data = data.drop('ID', axis=1)

# Identifying and dropping columns with more than 66% missing values
null_columns = data.loc[:, data.isna().sum() > 0.66 * data.shape[0]].columns
data = data.drop(null_columns, axis=1)

# Displaying the count of missing values in each column
data.isna().sum()

# Filling missing values in the 'Root Mean Square' column with the mean value
data['Root Mean Square'] = data['Root Mean Square'].fillna(data['Root Mean Square'].mean())

# Dropping rows with any remaining missing values and resetting the index
data = data.dropna(axis=0).reset_index(drop=True)

# Confirming there are no more missing values in the dataset
data.isna().sum().sum()
```

```python
# Feature Engineering: Extracting 'Month', 'Year', and 'Hour' from 'Date' and 'Time'
data['Month'] = data['Date'].apply(lambda x: x[0:2])
data['Year'] = data['Date'].apply(lambda x: x[-4:])

# Converting 'Month' to integer type
data['Month'] = data['Month'].astype(np.int)

# Handling invalid 'Year' entries and converting to integer type
data[data['Year'].str.contains('Z')]
invalid_year_indices = data[data['Year'].str.contains('Z')].index
data = data.drop(invalid_year_indices, axis=0).reset_index(drop=True)
data['Year'] = data['Year'].astype(np.int)

# Extracting 'Hour' from 'Time' and displaying the modified dataset
data['Hour'] = data['Time'].apply(lambda x: np.int(x[0:2]))
data

# Displaying the shape and columns of the final dataset
data.shape
data.columns
```

```
# Selecting relevant columns and displaying the first few rows of the modified dataset
data = data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']]
data.head()


# Converting 'Date' and 'Time' to a timestamp in seconds
import datetime
import time


timestamp = []
for d, t in zip(data['Date'], data['Time']):
    try:
        ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S')
        timestamp.append(time.mktime(ts.timetuple()))
    except ValueError:
        # Handling cases where timestamp conversion fails
        timestamp.append('ValueError')


# Creating a new 'Timestamp' column in the dataset
timeStamp = pd.Series(timestamp)
data['Timestamp'] = timeStamp.values
```

 # Creating the final dataset by dropping 'Date' and 'Time' columns and removing rows with invalid timestamps

final_data = data.drop(['Date', 'Time'], axis=1)

final_data = final_data[final_data.Timestamp != 'ValueError']

final_data.head()

OUTPUT :

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns

    from sklearn.preprocessing import StandardScaler
    from sklearn.model_selection import train_test_split

    import tensorflow as tf
```

```
[ ] data = pd.read_csv('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
...	...	...	...	...	...	...	...	...	...	...

```
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
```

+ Code + Text

Feature Engineering ..

```
[ ] data
```

	Date	Time	Latitude	Longitude	Type	Depth	Magnitude	Magnitude Type	Root Mean Square	Source	Location Source
0	01/02/1965	13:44:18	19.2460	145.6160	Earthquake	131.60	6.0	MW	1.022784	ISCGEM	ISCGEM
1	01/04/1965	11:29:49	1.8630	127.3520	Earthquake	80.00	5.8	MW	1.022784	ISCGEM	ISCGEM
2	01/05/1965	18:05:58	-20.5790	-173.9720	Earthquake	20.00	6.2	MW	1.022784	ISCGEM	ISCGEM
3	01/08/1965	18:49:43	-59.0760	-23.5570	Earthquake	15.00	5.8	MW	1.022784	ISCGEM	ISCGEM
4	01/09/1965	13:32:50	11.9380	126.4270	Earthquake	15.00	5.8	MW	1.022784	ISCGEM	ISCGEM
...	...	...	...	...	...	...	...	...	...	...	...
23404	12/28/2016	08:22:12	38.3917	-118.8941	Earthquake	12.30	5.6	ML	0.189800	NN	NN
23405	12/28/2016	09:13:47	38.3777	-118.8957	Earthquake	8.80	5.5	ML	0.218700	NN	NN
23406	12/28/2016	12:38:51	36.9179	140.4262	Earthquake	10.00	5.9	MWW	1.520000	US	US
23407	12/29/2016	22:30:19	-9.0283	118.6639	Earthquake	79.00	6.3	MWW	1.430000	US	US
23408	12/30/2016	20:08:28	37.3973	141.4103	Earthquake	11.94	5.5	MB	0.910000	US	US

23409 rows × 13 columns

```
[ ] data['Month'] = data['Date'].apply(lambda x: x[0:2])
    data['Year'] = data['Date'].apply(lambda x: x[-4:])
```

```
[ ] data['Month'] = data['Month'].astype(np.int)

    <ipython-input-120-7b03c2eae7e8>:1: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To sile
    Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecat
      data['Month'] = data['Month'].astype(np.int)
```

```
[ ] data[data['Year'].str.contains('Z')]
```

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

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

```
[ ] data = data.drop(null_columns, axis=1)
```

```
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
```

```
[ ] data[data['Year'].str.contains('Z')]
```

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

```
[ ] invalid_year_indices = data[data['Year'].str.contains('Z')].index

    data = data.drop(invalid_year_indices, axis=0).reset_index(drop=True)
```

```
[ ] invalid_year = data[data['Year'].str.contains('Z')].index
```

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

```
<ipython-input-124-ca853ac0c7ce>:1: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To sile
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecat
  data['Year'] = data['Year'].astype(np.int)
```

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

```
<ipython-input-125-148729bf835d>:1: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To sile
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecat
  data['Hour'] = data['Time'].apply(lambda x: np.int(x[0:2]))
```

```
[ ] data
```

	Date	Time	Latitude	Longitude	Type	Depth	Magnitude	Magnitude Type	Root Mean Square	Source	Location Source
0	01/02/1965	13:44:18	19.2460	145.6160	Earthquake	131.60	6.0	MW	1.022784	ISCGEM	ISCGEM
1	01/04/1965	11:29:49	1.8630	127.3520	Earthquake	80.00	5.8	MW	1.022784	ISCGEM	ISCGEM
2	01/05/1965	18:05:58	-20.5790	-173.9720	Earthquake	20.00	6.2	MW	1.022784	ISCGEM	ISCGEM
3	01/08/1965	18:49:43	-59.0760	-23.5570	Earthquake	15.00	5.8	MW	1.022784	ISCGEM	ISCGEM
4	01/09/1965	13:32:50	11.9380	126.4270	Earthquake	15.00	5.8	MW	1.022784	ISCGEM	ISCGEM
...	...	...	...	...	...	...	...	...	...	...	...
23401	12/28/2016	08:22:12	38.3917	-118.8941	Earthquake	12.30	5.6	ML	0.189800	NN	NN
23402	12/28/2016	09:13:47	38.3777	-118.8957	Earthquake	8.80	5.5	ML	0.218700	NN	NN
23403	12/28/2016	12:38:51	36.9179	140.4262	Earthquake	10.00	5.9	MWW	1.520000	US	US
23404	12/29/2016	22:30:19	-9.0283	118.6639	Earthquake	79.00	6.3	MWW	1.430000	US	US
23405	12/30/2016	20:08:28	37.3973	141.4103	Earthquake	11.94	5.5	MB	0.910000	US	US

23406 rows × 16 columns

```
[ ] data.shape
```

```
(23406, 16)
```

```
[ ] data.columns
```

```
Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Magnitude',
       'Magnitude Type', 'Root Mean Square', 'Source', 'Location Source',
       'Magnitude Source', 'Status', 'Month', 'Year', 'Hour'],
      dtype='object')
```

```
[ ] data = data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']]
    data.head()
```

	Date	Time	Latitude	Longitude	Depth	Magnitude
0	01/02/1965	13:44:18	19.246	145.616	131.6	6.0
1	01/04/1965	11:29:49	1.863	127.352	80.0	5.8
2	01/05/1965	18:05:58	-20.579	-173.972	20.0	6.2
3	01/08/1965	18:49:43	-59.076	-23.557	15.0	5.8
4	01/09/1965	13:32:50	11.938	126.427	15.0	5.8

```
[ ] import datetime
    import time

    timestamp = []
    for d, t in zip(data['Date'], data['Time']):
        try:
            ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S')
            timestamp.append(time.mktime(ts.timetuple()))
        except ValueError:
            # print('ValueError')
            timestamp.append('ValueError')
```

```
[ ] timeStamp = pd.Series(timestamp)
    data['Timestamp'] = timeStamp.values
```

```
final_data = data.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'ValueError']
final_data.head()
```

1 to 5 of 5 entries Filter

index	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	-157630542.0
1	1.863	127.352	80.0	5.8	-157465811.0
2	-20.579	-173.972	20.0	6.2	-157355642.0
3	-59.076	-23.557	15.0	5.8	-157093817.0
4	11.938	126.427	15.0	5.8	-157026430.0

CONCLUSION

The loading and preprocessing of the earthquake dataset involved several key steps. The process began by loading the data and examining its structure, leading to the removal of the 'ID' column. Missing values were handled by dropping columns with a substantial amount of missing data and imputing the mean for the 'Root Mean Square' column. Feature engineering included extracting relevant information like 'Month', 'Year', and 'Hour' from 'Date' and 'Time'. Invalid entries in the 'Year' column were addressed. The dataset was further refined by selecting essential features and transforming 'Date' and 'Time' into a 'Timestamp' column. These steps ensure data integrity, enhance feature representation, and set the stage for constructing a robust earthquake prediction model, marking the dataset's readiness for subsequent analysis and model development.