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 LINK**]

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.

DATA LOADING

Data loading is the inaugural step in machine learning, essential foracquiring datasets that fuel model development. Identifying the datasource, 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.

PREPROCESSING

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, guidingsubsequent preprocessing decisions.

PROGRAM :

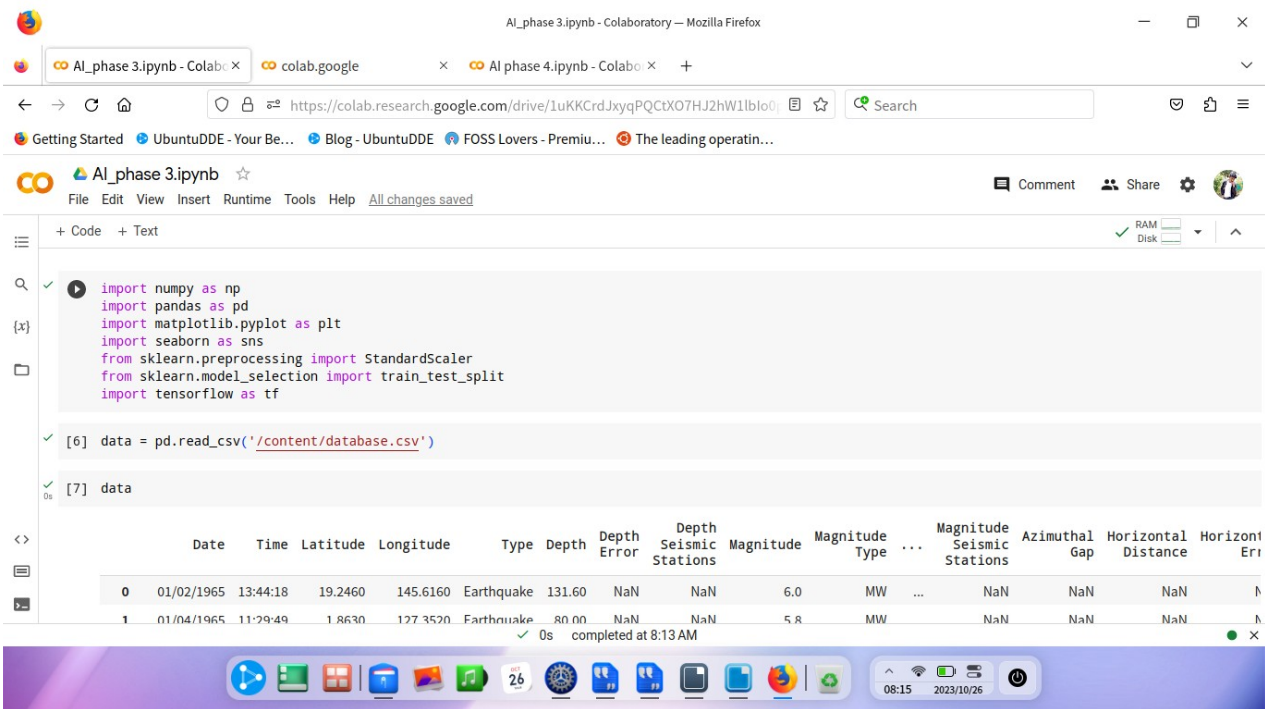
```
# Importing necessary libraries
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.preprocessing import StandardScaler
    from sklearn.model_selection import train_test_split
    import tensorflow as tf
# Reading the dataset 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()
# 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 :



```
[8] 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
```

```
 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)
```

```
[10] null_columns = data.loc[:, data.isna().sum() > 0.66 *
     data.shape[0]].columns
     data = data.drop(null_columns, axis=1)
```

```
[11] data.isna().sum()
```

AI_phase 3.ipynb - Colaboratory — Mozilla Firefox

```
[11] 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
```

```
[13] data['Root Mean Square'] = data['Root Mean Square'].fillna(data['Root Mean Square'].mean())
```

completed at 8:13 AM

AI_phase 3.ipynb - Colaboratory — Mozilla Firefox

```
[14] data = data.dropna(axis=0).reset_index(drop=True)
```

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

```
0
```

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

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

	Date	Time	Latitude	Longitude	Depth	Magnitude	Month	Year	Hour
0	01/02/1965	13:44:18	19.2460	145.6160	131.60	6.0	01	1965	13
1	01/04/1965	11:29:49	1.8630	127.3520	80.00	5.8	01	1965	11
2	01/05/1965	18:05:58	-20.5790	-173.9720	20.00	6.2	01	1965	18

completed at 8:13 AM

AI_phase 3.ipynb - Colaboratory — Mozilla Firefox

AI_phase 3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

[35]

	Date	Time	Latitude	Longitude	Depth	Magnitude	Month	Year	Hour
0	01/02/1965	13:44:18	19.2460	145.6160	131.60	6.0	01	1965	13
1	01/04/1965	11:29:49	1.8630	127.3520	80.00	5.8	01	1965	11
2	01/05/1965	18:05:58	-20.5790	-173.9720	20.00	6.2	01	1965	18
3	01/08/1965	18:49:43	-59.0760	-23.5570	15.00	5.8	01	1965	18
4	01/09/1965	13:32:50	11.9380	126.4270	15.00	5.8	01	1965	13
...	...	...	...	...	...	...	...	...	...
23401	12/28/2016	08:22:12	38.3917	-118.8941	12.30	5.6	12	2016	8
23402	12/28/2016	09:13:47	38.3777	-118.8957	8.80	5.5	12	2016	9
23403	12/28/2016	12:38:51	36.9179	140.4262	10.00	5.9	12	2016	12
23404	12/29/2016	22:30:19	-9.0283	118.6639	79.00	6.3	12	2016	22
23405	12/30/2016	20:08:28	37.3973	141.4103	11.94	5.5	12	2016	20

23406 rows × 9 columns

0s completed at 8:13 AM

AI_phase 3.ipynb - Colaboratory — Mozilla Firefox

AI_phase 3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

[36]

```
data.shape
data.columns
```

```
Index(['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude', 'Month',
       'Year', 'Hour'],
      dtype='object')
```

[24]

```
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

0s completed at 8:13 AM

AI_phase 3.ipynb - Colaboratory — Mozilla Firefox

AI_phase 3.ipynb

```
[47] data['Timestamp']
```

```
0       -1.576305e+08
1       -1.574658e+08
2       -1.573556e+08
3       -1.570938e+08
4       -1.570264e+08
             ...
23401    1.482913e+09
23402    1.482916e+09
23403    1.482929e+09
23404    1.483051e+09
23405    1.483129e+09
Name: Timestamp, Length: 23406, dtype: float64
```

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

Latitude Longitude Depth Magnitude Month Year Hour Timestamp

✓ 0s completed at 8:13 AM

AI_phase 3.ipynb - Colaboratory — Mozilla Firefox

AI_phase 3.ipynb

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

	Latitude	Longitude	Depth	Magnitude	Month	Year	Hour	Timestamp
0	19.246	145.616	131.6	6.0	01	1965	13	-157630542.0
1	1.863	127.352	80.0	5.8	01	1965	11	-157465811.0
2	-20.579	-173.972	20.0	6.2	01	1965	18	-157355642.0
3	-59.076	-23.557	15.0	5.8	01	1965	18	-157093817.0
4	11.938	126.427	15.0	5.8	01	1965	13	-157026430.0

```
[ ]
```

✓ 0s completed at 8:13 AM

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.