IBM NaanMudhalvan

ARTIFICIAL INTELLIGENCE

Project Title : Earthquake Prediction Using Python

Phase 2 : Importing the Dataset and Perform data
. Cleaning & Data Analysis.

Workbook Link : Google Colab

INTRODUCTION

In the realm of Earthquake Prediction using Machine Learning, the initial steps of importing the dataset and conducting meticulous data cleaning are pivotal. This project begins by acquiring seismic data, a critical precursor to predictive modeling. Rigorous data cleaning techniques are then employed to ensure the dataset's integrity and reliability. Subsequently, through advanced data analysis, we aim to unveil patterns and insights crucial for developing a robust ML model capable of predicting seismic activities. This introduction sets the stage for a comprehensive exploration of earthquake prediction, emphasizing the foundational role of data import and cleaning in the ML-driven analytical process.

WORKSPACE

We've worked on Google Colab for the intricate task of data cleaning and analysis in Earthquake Prediction using Python. Google Colab served as a powerful and accessible platform. Leveraging the collaborative and cloud-based features of Google Colab facilitated seamless collaboration and efficient processing of seismic datasets. The platform's integration with popular Python libraries streamlined coding and analysis workflows, enhancing productivity. For a detailed walkthrough of the data cleaning and analysis process, refer to the Notebook on Google Colab, Click Here.....

IMPORTING THE DATASET

Importing the dataset is the foundational step in our Earthquake Prediction using ML project. We seamlessly fetched seismic data from reliable sources, ensuring its accuracy and relevance. Leveraging the versatility of Python, we employed libraries like Pandas to efficiently read and organize the dataset for subsequent analysis. The chosen dataset encompasses essential seismic parameters, forming the basis for training and validating our machine learning model. The streamlined import process lays the groundwork for a comprehensive exploration into earthquake prediction methodologies.

PROGRAM :

Original file is located at
https://colab.research.google.com/drive/1IHe_-veRrUX6y4RuHVhBIvX-_oGMLYa_?usp=sharing

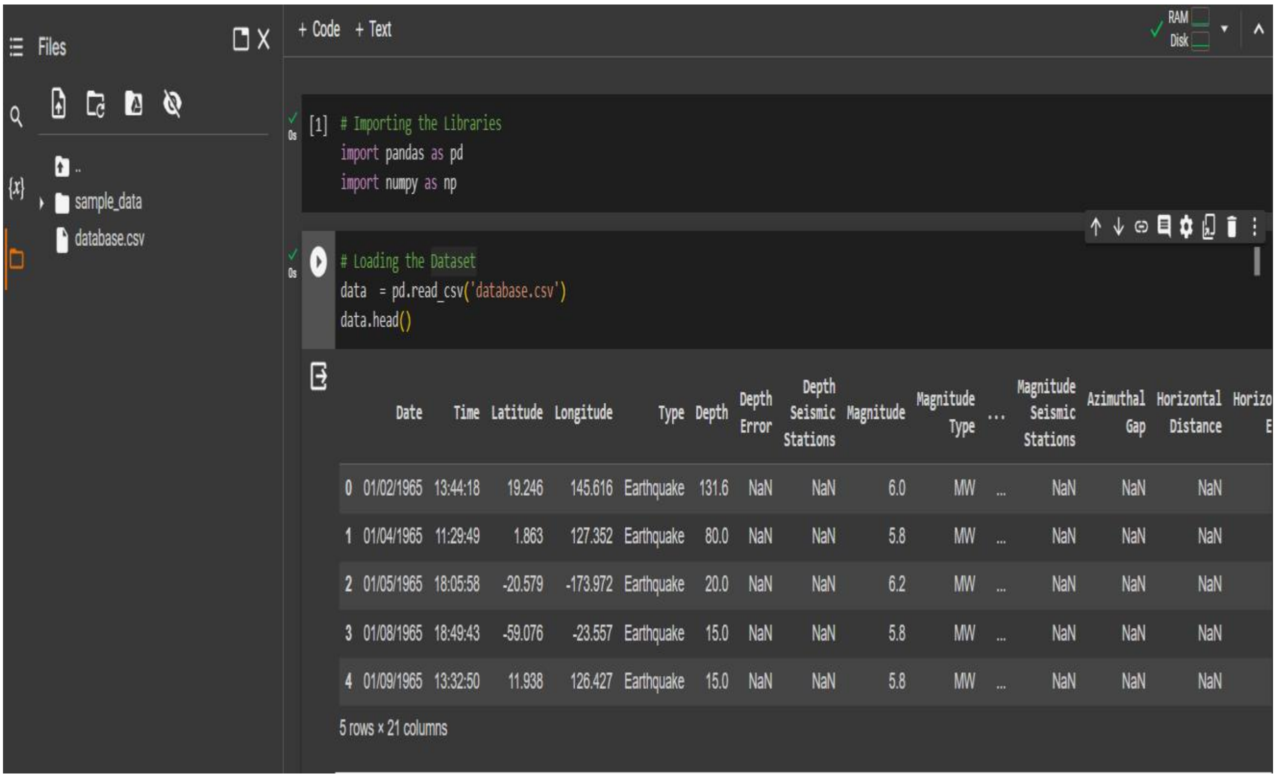
```
# Importing the Libraries
import pandas as pd
import numpy as np
```

 # Loading the Dataset

data = pd.read_csv('database.csv')

data.head()

OUTPUT :



	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	...	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance	Horizo E
0	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	...	NaN	NaN	NaN	
1	01/04/1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	
2	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	...	NaN	NaN	NaN	
3	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	
4	01/09/1965	13:32:50	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	

5 rows × 21 columns

DATA ANALYSIS

Data analysis in our Earthquake Prediction using ML project involves a meticulous exploration of seismic patterns and trends. Employing Python-based tools like NumPy and Pandas, we conducted descriptive statistics, revealing key insights into the dataset's characteristics. Visualization techniques, implemented with libraries such as Matplotlib and Seaborn, aided in uncovering spatial and temporal aspects of seismic activity. Correlation analysis provided a deeper understanding of feature relationships, guiding the model development process. The comprehensive data analysis phase contributes crucial inputs for building a robust machine learning model for earthquake prediction.

PROGRAM :

```
# Checking the Shape of the Dataset
data.shape

# Checking the Number of Entities
data.columns

# Checking Descriptive Structure of the data
data.describe()

# Checking Duplicated Rows.
```

data.duplicated()

 # Checking the Data Information

data.info()

df = pd.DataFrame(data)

 # Checking Categorical and Numerical Columns

 # Categorical columns

cat_col = [col for col in df.columns if df[col].dtype == 'object']

print('Categorical columns :',cat_col)

 # Numerical columns

num_col = [col for col in df.columns if df[col].dtype != 'object']

print('Numerical columns :',num_col)

 # Checking total number of Values in Categorical Columns

df[cat_col].nunique()

 # Checking total number of Values in Numerical Columns

df[num_col].nunique()

 # Checking the Missing Values Percentage

round((df.isnull().sum()/df.shape[0])*100,2)

OUTPUT :

```
# Checking the Shape of the Dataset
data.shape
```

```
(23412, 21)
```

```
[4] # Checking the Number of Entities
data.columns
```

```
Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth Error',
       'Depth Seismic Stations', 'Magnitude', 'Magnitude Type',
       'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
       'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',
       'Source', 'Location Source', 'Magnitude Source', 'Status'],
      dtype='object')
```

```
[5] # Checking Descriptive Structure of the data
data.describe()
```

	Latitude	Longitude	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Error	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance	Horizontal Error	Root Mean Square
count	23412.000000	23412.000000	23412.000000	4461.000000	7097.000000	23412.000000	327.000000	2564.000000	7299.000000	1604.000000	1156.000000	17352.000000
mean	1.679033	39.639961	70.767911	4.993115	275.364098	5.882531	0.071820	48.944618	44.163532	3.992660	7.662759	1.022784
std	30.113183	125.511959	122.651898	4.875184	162.141631	0.423066	0.051466	62.943106	32.141486	5.377262	10.430396	0.188545
min	-77.080000	-179.997000	-1.100000	0.000000	0.000000	5.500000	0.000000	0.000000	0.000000	0.004505	0.085000	0.000000
25%	-18.653000	-76.349750	14.522500	1.800000	146.000000	5.600000	0.046000	10.000000	24.100000	0.968750	5.300000	0.900000
50%	-3.568500	103.982000	33.000000	3.500000	255.000000	5.700000	0.059000	28.000000	36.000000	2.319500	6.700000	1.000000
75%	26.190750	145.026250	54.000000	6.300000	384.000000	6.000000	0.075500	66.000000	54.000000	4.724500	8.100000	1.130000
max	86.005000	179.998000	700.000000	91.295000	934.000000	9.100000	0.410000	821.000000	360.000000	37.874000	99.000000	3.440000

```
[6] # Checking Duplicated Rows.
    data.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
23407    False
23408    False
23409    False
23410    False
23411    False
Length: 23412, dtype: bool
```

```
# Checking the Data Information
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
```

```
[8] df = pd.DataFrame(data)
```

```
[9] # Checking Categorical and Numerical Columns
    # Categorical columns
    cat_col = [col for col in df.columns if df[col].dtype == 'object']
    print('Categorical columns :',cat_col)
    # Numerical columns
    num_col = [col for col in df.columns if df[col].dtype != 'object']
    print('Numerical columns :',num_col)
```

```
Categorical columns : ['Date', 'Time', 'Type', 'Magnitude Type', 'ID', 'Source', 'Location Source', 'Magnitude Source', 'Status']
Numerical columns : ['Latitude', 'Longitude', 'Depth', 'Depth Error', 'Depth Seismic Stations', 'Magnitude', 'Magnitude Error', 'Magnit
```

```
# Checking total number of Values in Categorical Columns
df[cat_col].nunique()
```

```
Date                12401
Time                20472
Type                    4
Magnitude Type         10
ID                  23412
Source                 13
Location Source        48
Magnitude Source       24
Status                  2
dtype: int64
```

```
[11] # Checking total number of Values in Numerical Columns
     df[num_col].nunique()
```

```
Latitude                      20676
Longitude                     21474
Depth                          3485
Depth Error                     297
Depth Seismic Stations          736
Magnitude                        64
Magnitude Error                 100
Magnitude Seismic Stations      246
Azimuthal Gap                  1109
Horizontal Distance            1448
Horizontal Error                186
Root Mean Square                190
dtype: int64
```

FEATURE ENGINEERING

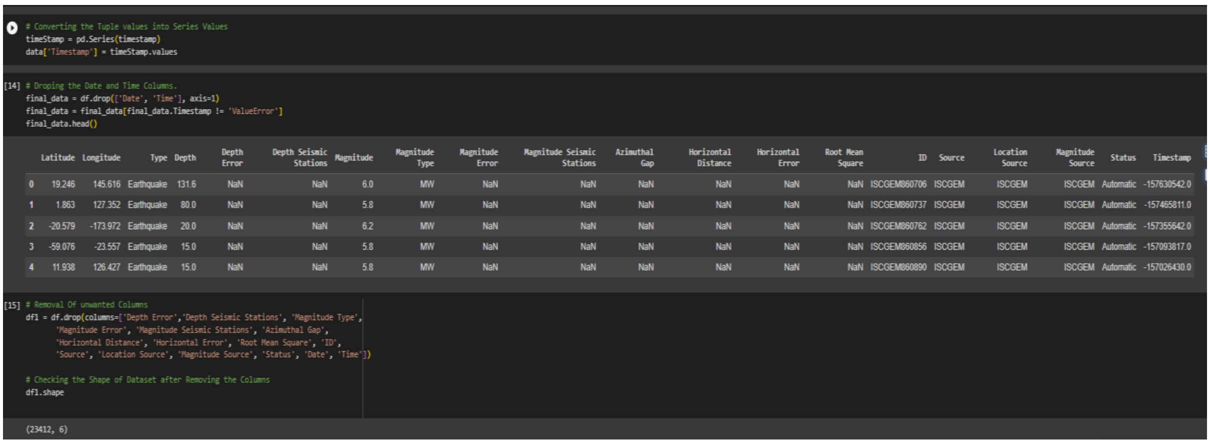
Feature engineering is a critical aspect of machine learning where raw data is transformed or new features are created to enhance model performance. It involves techniques like polynomial expansion, interaction terms, and domain-specific transformations to extract meaningful information. Dimensionality reduction methods, such as PCA, help manage high-dimensional data, preventing overfitting and improving model efficiency. Handling categorical variables through encoding methods ensures effective utilization of non-numeric data. Feature engineering is an iterative process, guided by continuous evaluation and refinement to build models that accurately capture underlying patterns in the data.

PROGRAM :

```
# Creating Timestamp Column from Data and Time Column
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')

# Converting the Tuple values into Series Values
timeStamp = pd.Series(timestamp)
data['Timestamp'] = timeStamp.values

# Droping the Date and Time Columns.
final_data = df.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'ValueError']
final_data.head()
```

OUTPUT :



```
# Converting the Tuple values into Series Values
timeStamp = pd.Series(timestamp)
data['Timestamp'] = timeStamp.values
```

```
[14] # Droping the Date and Time Columns.
final_data = df.drop(['Date', 'Time'], axis=1)
final_data = final_data[final_data.Timestamp != 'ValueError']
final_data.head()
```

	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	Magnitude Error	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance	Horizontal Error	Root Mean Square	ID	Source	Location Source	Magnitude Source	Status	Timestamp
0	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860706	ISCGEM	ISCGEM	ISCGEM	Automatic	-157630542.0
1	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860737	ISCGEM	ISCGEM	ISCGEM	Automatic	-157465811.0
2	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860762	ISCGEM	ISCGEM	ISCGEM	Automatic	-157355642.0
3	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860856	ISCGEM	ISCGEM	ISCGEM	Automatic	-157093817.0
4	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW	NaN	NaN	NaN	NaN	NaN	NaN	ISCGEM860890	ISCGEM	ISCGEM	ISCGEM	Automatic	-157026430.0

```
[15] # Removal Of unwanted Columns
df1 = df.drop(columns=['Depth Error','Depth Seismic Stations', 'Magnitude Type',
        'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
        'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',
        'Source', 'Location Source', 'Magnitude Source', 'Status', 'Date', 'Time'])

# Checking the Shape of Dataset after Removing the Columns
df1.shape
```

```
(23412, 6)
```

DATA CLEANING

PROGRAM:

```
# Removal Of Unwanted Columns
df1 = df.drop(columns=['Depth Error','Depth Seismic Stations', 'Magnitude Type',
       'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
       'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',
       'Source', 'Location Source', 'Magnitude Source', 'Status', 'Date', 'Time'])

# Checking the Shape of Dataset after Removing the Columns
df1.shape

df1.head(10)

# Checking Columns
df1.columns

# Checking the Missing Values Percentage
round((df1.isnull().sum()/df1.shape[0])*100,2)
```

```
# Checking the Data Information After droping the Unwanted Columns
df1.info()

# Checking the Descriptive Structure of the Data after the removal of Unwanted Columns
df1.describe()

# Checking Categorical and Numerical Columns
# Categorical columns
cat_col = [col for col in df1.columns if df1[col].dtype == 'object']
print('Categorical columns :',cat_col)

# Numerical columns
num_col = [col for col in df1.columns if df1[col].dtype != 'object']
print('Numerical columns :',num_col)

# Checking total number of Values in Categorical Columns
df1[cat_col].nunique()
```

 # Checking total number of Values in Numerical Columns

df[num_col].nunique()

 # Let's check the null values again

df1.isnull().sum()

OUTPUT:

```
[15] # Removal Of unwanted Columns
     df1 = df.drop(columns=['Depth Error','Depth Seismic Stations', 'Magnitude Type',
            'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
            'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',
            'Source', 'Location Source', 'Magnitude Source', 'Status', 'Date', 'Time'])

     # Checking the Shape of Dataset after Removing the Columns
     df1.shape

     (23412, 6)
```

```
df1.head(10)
```

	Latitude	Longitude	Type	Depth	Magnitude	Timestamp
0	19.246	145.616	Earthquake	131.6	6.0	-157630542.0
1	1.863	127.352	Earthquake	80.0	5.8	-157465811.0
2	-20.579	-173.972	Earthquake	20.0	6.2	-157355642.0
3	-59.076	-23.557	Earthquake	15.0	5.8	-157093817.0
4	11.938	126.427	Earthquake	15.0	5.8	-157026430.0
5	-13.405	166.629	Earthquake	35.0	6.7	-156939808.0
6	27.357	87.867	Earthquake	20.0	5.9	-156767255.0
7	-13.309	166.212	Earthquake	35.0	6.0	-156472938.0
8	-56.452	-27.043	Earthquake	95.0	6.0	-156428843.0
9	-24.563	178.487	Earthquake	565.0	5.8	-156345403.0

```
[17] # Checking Columns
     df1.columns

     Index(['Latitude', 'Longitude', 'Type', 'Depth', 'Magnitude', 'Timestamp'], dtype='object')
```

```
[18] # Checking the Missing Values Percentage
     round((df1.isnull().sum()/df1.shape[0])*100,2)
```

```
Latitude     0.0
Longitude    0.0
Type         0.0
Depth        0.0
Magnitude    0.0
Timestamp    0.0
dtype: float64
```

```
# Checking the Data Information After droping the Unwanted Columns
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23412 entries, 0 to 23411
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Latitude   23412 non-null  float64
 1   Longitude  23412 non-null  float64
 2   Type       23412 non-null  object
 3   Depth      23412 non-null  float64
 4   Magnitude  23412 non-null  float64
 5   Timestamp  23412 non-null  object
dtypes: float64(4), object(2)
memory usage: 1.1+ MB
```

```python
# Checking the Descriptive Structure of the Data after the removal of Unwanted Columns
df1.describe()
```

	Latitude	Longitude	Depth	Magnitude
count	23412.000000	23412.000000	23412.000000	23412.000000
mean	1.679033	39.639961	70.767911	5.882531
std	30.113183	125.511959	122.651898	0.423066
min	-77.080000	-179.997000	-1.100000	5.500000
25%	-18.653000	-76.349750	14.522500	5.600000
50%	-3.568500	103.982000	33.000000	5.700000
75%	26.190750	145.026250	54.000000	6.000000
max	86.005000	179.998000	700.000000	9.100000

[21]
```python
# Checking Categorical and Numerical Columns
# Categorical columns
cat_col = [col for col in df1.columns if df1[col].dtype == 'object']
print('Categorical columns :',cat_col)
# Numerical columns
num_col = [col for col in df1.columns if df1[col].dtype != 'object']
print('Numerical columns :',num_col)
```

```
Categorical columns : ['Type', 'Timestamp']
Numerical columns : ['Latitude', 'Longitude', 'Depth', 'Magnitude']
```

[22]
```python
# Checking total number of Values in Categorical Columns
df1[cat_col].nunique()
```

```
Type             4
Timestamp    23391
dtype: int64
```

[22]
```python
# Checking total number of Values in Categorical Columns
df1[cat_col].nunique()
```

```
Type             4
Timestamp    23391
dtype: int64
```


[23]
```python
# Checking total number of Values in Numerical Columns
df[num_col].nunique()
```

```
Latitude     20676
Longitude    21474
Depth         3485
Magnitude       64
dtype: int64
```

[24]
```python
# Let's check the null values again
df1.isnull().sum()
```

```
Latitude     0
Longitude    0
Type         0
Depth        0
Magnitude    0
Timestamp    0
dtype: int64
```

CONCLUSION

The process of earthquake prediction using machine learning involves meticulous data cleaning to ensure dataset reliability. Data importing combines seismic, geological, and environmental data for a comprehensive analysis. Feature engineering enhances the dataset, optimizing models for pattern recognition. Iterative refinement based on model performance fosters nuanced earthquake prediction insights. Overall, this approach, encompassing data cleaning, importing, and analysis, advances our ability to develop accurate machine learning models for mitigating the impact of seismic events.