EARTHQUAKE PREDICTION MODEL USING PYTHON

BATCH MEMBER 950621104079 M.SANDHIYA

Phase 3 Submission Document

**Project Title:** Earthquake Prediction Model Using Python

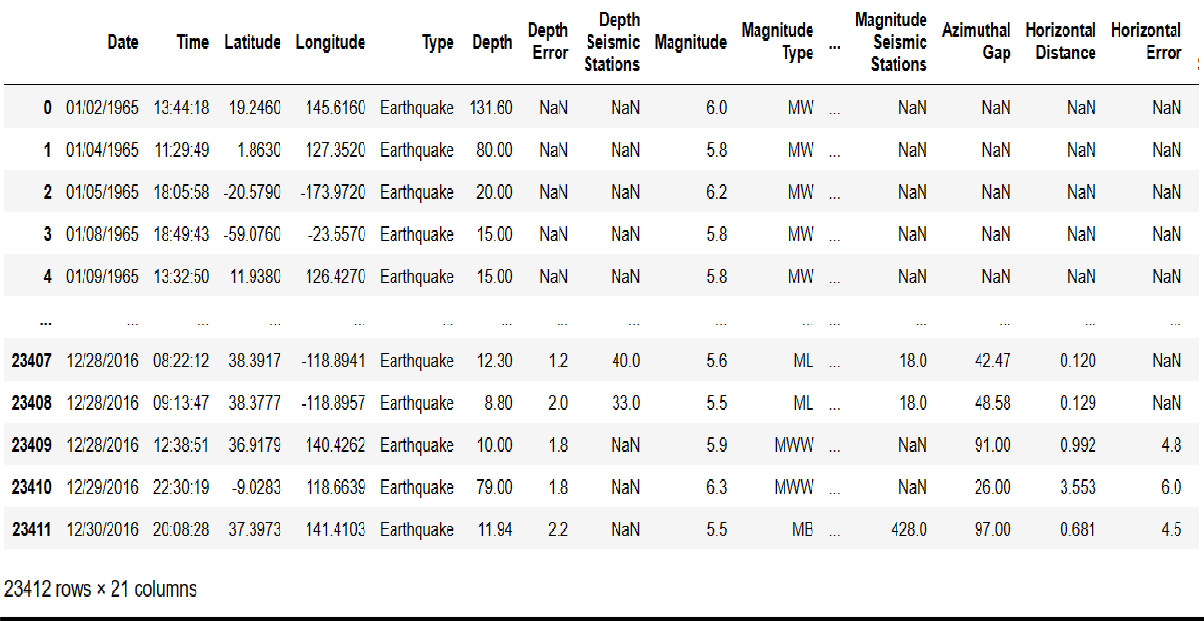
**Phase 3:** Development Part 1

**Topic:** Start building the earthquake prediction model by loading and pre-processing the dataset.

# Earthquake Prediction Model Using Python Introduction:

* Whether you’re using earthquake prediction app then the value of the data will be accurate and give you the notification about it.
* The data scientist aiming to build a predictive model, the foundation of this endeavour lies in loading and preprocessing the dataset.
* This introduction will guide you through the initial steps of the process.
* Data preprocessing is crucial as it helps clean, format, and prepare the data for further analysis.
* This include handling missing values, encoding categorical variables, and ensuring that the data is appropriately scaled.

# Given Dataset:



**Necessary Steps to follow:**

# Import Libraries:

Start by importing the necessary libraries:

# Program:

Import pandas as pd Import numpy as np

From sklearn.model\_selection import train\_test\_split From sklearn.preprocessing import StandardScaler

# Load the Dataset:

Load the dataset into the Pandas dataframe. We can typically find earthquake prediction dataset in CSV format, but we can adapt this code to other formats as needed.

# Program:

df = pd.read\_csv('C:/Users/barat/Downloads/archive/database.csv') df

# Exploratory Data Analysis(EDA):

Perform EDA to understand the data better.

# Programs:

# Check for missing values Print(df.isnull().sum()) #Explore statistics Print(df.describe()) #Visulaize the data

(eg., histogram, scatter plots, etc.)

# Split the Data:

Split the dataset into training and testing sets. This helps us to evaluate the model’s performance later.

# Program:

X = df.drop('Depth',axis=1) y=df['Depth']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Feature Scaling:

Apply feature scaling to normalize the data, ensuring that all features have similar scales.

# Program:

scaler = StandardScaler()

X\_train= scaler.fit\_transform(X\_train) X\_test = scaler.transform(X\_test);

# Challenges :

There are a number of challenges involved in loading and preprocessing a earthquake prediction dataset, including:

# Handling missing values:

Earthquake prediction dataset often contain missing values, which can be due to a variety of factors, such as human error or incomplete data collection.

# Splitting the dataset into training and testing sets:

Once the data has been pre-processed, we need to split the dataset into training and testing sets. The training set will be used to train the model, and the testing set will be used to evaluate the performance of the model on unseen data.

# Program:

1. **Importing the needed libraries:**

import pandas as pd import numpy as np import seaborn as sns

import matplotlib.pyplot as plt

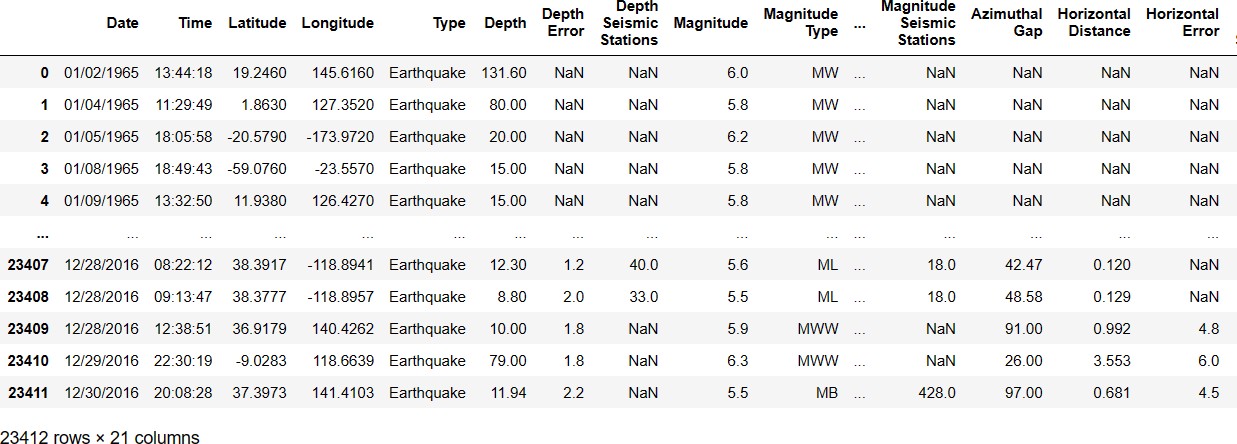
from sklearn.model\_selection import train\_test\_split from sklearn.compose import ColumnTransformer from sklearn.preprocessing import StandardScaler from sklearn.preprocessing import OneHotEncoder from sklearn.pipeline import Pipeline

*Loading Dataset:*

df = pd.read\_csv('C:/Users/barat/Downloads/archive/database.csv')

*Data Exploration:*

Dataset:



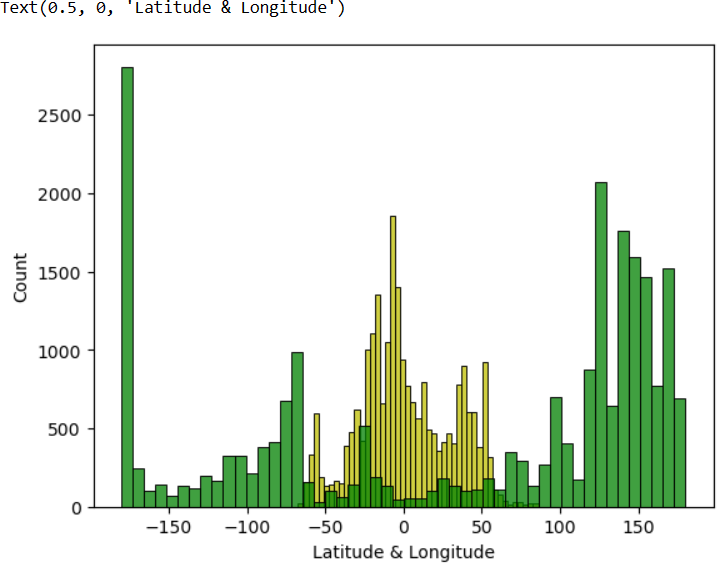
# Preprocessing :

*Visualisation and pre-processing of data:*

In[1]:

sns.histplot(df,x = 'Latitude', bins=50, color='y') sns.histplot(df,x = 'Longitude', bins=50, color='g') plt.xlabel('Latitude & Longitude')

Out[1]:

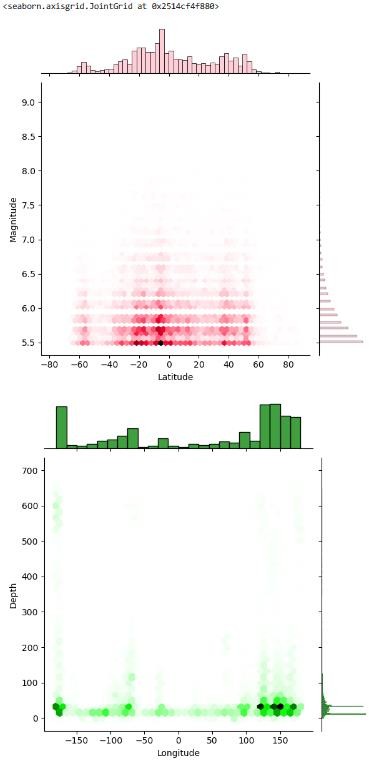


Out[2]:

In[2]:

sns.jointplot(df,x='Latitude', y='Magnitude', kind='hex', color='pink') sns.jointplot(df,x='Longitude', y='Depth', kind='hex', color='green')

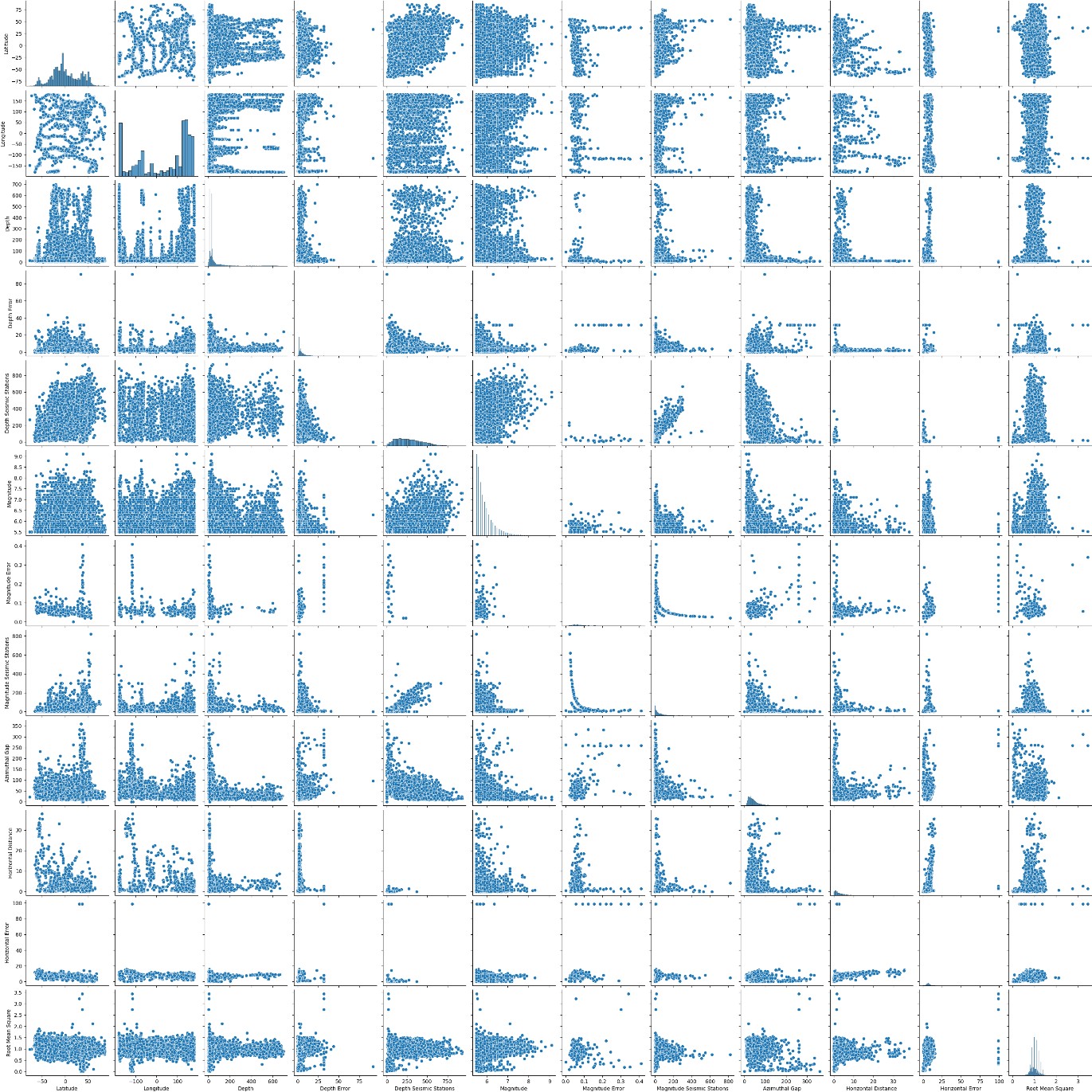
Out[2]:



In[3]:

plt.figure(fig size=(6,8)) sns.pairplot(df) plt.show()

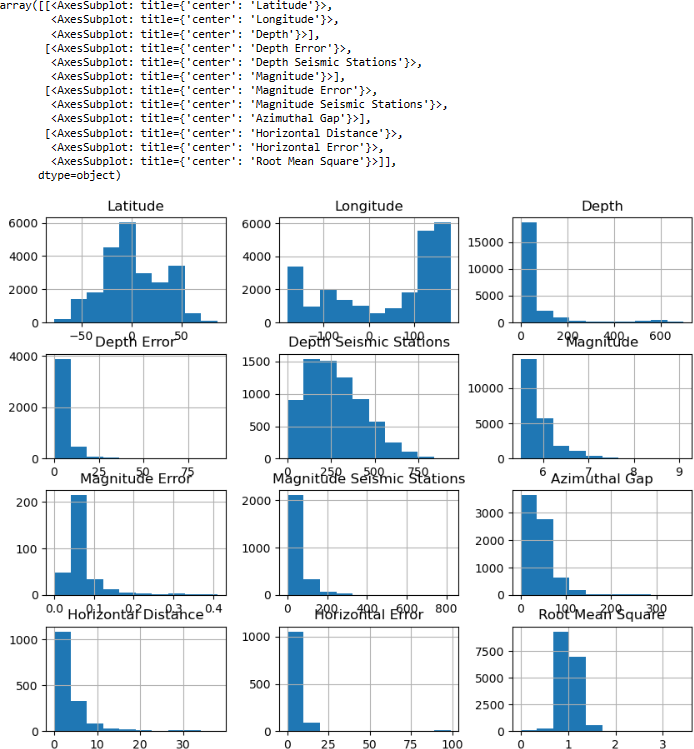
Out[3]:



In[4]:

df.hist(figsize=(10,8))

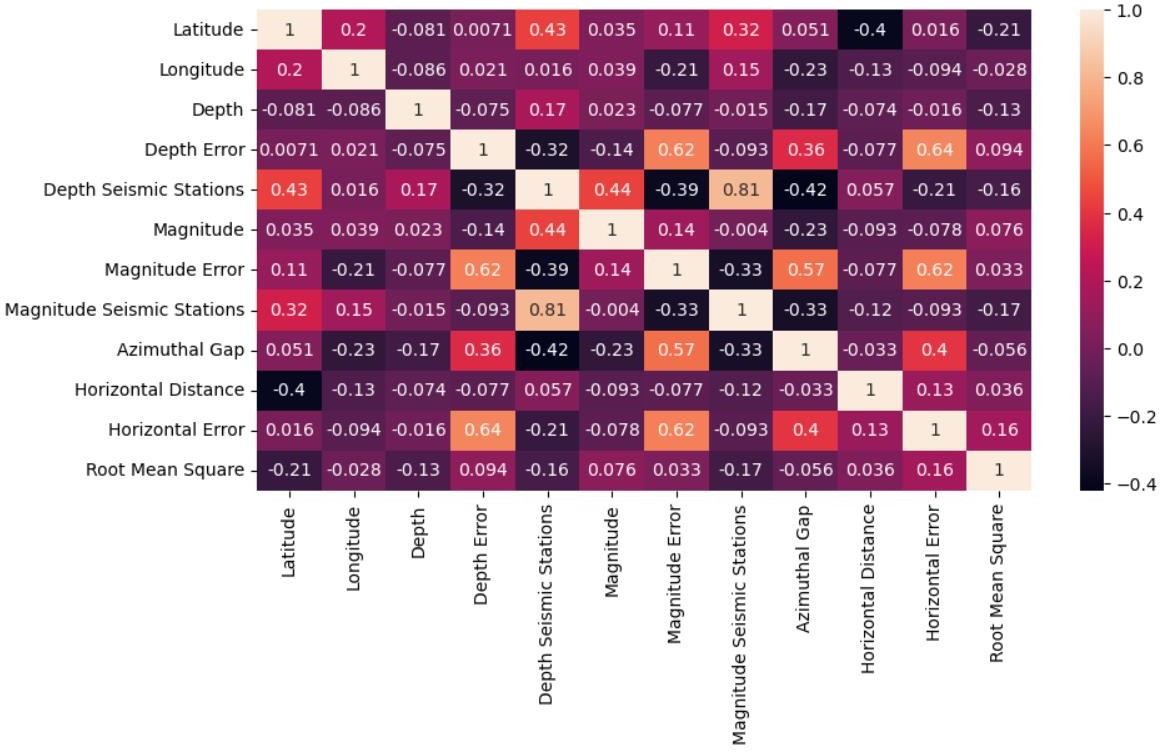
Out[4]:



In[5]:

plt.figure(figsize=(10,5)) sns.heatmap(df.corr(numeric\_only= True),annot = True) plt.show()

Out[5]:



# Some common data preprocessing tasks include:

# Program:

import pandas as pd import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split from sklearn.compose import ColumnTransformer from sklearn.preprocessing import StandardScaler from sklearn.preprocessing import OneHotEncoder from sklearn.pipeline import Pipeline

*Load the dataset:*

In[1]:

df = pd.read\_csv('C:/Users/barat/Downloads/archive/database.csv')

*Exploratory Data Analysis:*

In[2]:

missing\_values = df.isnull().sum() print(missing\_values)

*Feature Engineering:*

In[3]:

description = df.describe() print(description)

*Data Splitting:*

In[4]:

X = df.drop('Depth',axis=1) y=df['Depth']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

categorical\_cols = ['Longitude'] transformers=[

('num', StandardScaler(),['Magnitude','Date']), ('cat',OneHotEncoder(),categorical\_cols)

]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print(f"X\_train shape:{X\_train.shape}") print(f"X\_test shape:{X\_test.shape}") print(f"y\_train shape:{y\_train.shape}") print(f"y\_test shape:{y\_test.shape}")

*Preprocessing and feature sealing using pipeline:*

model = Pipeline( [

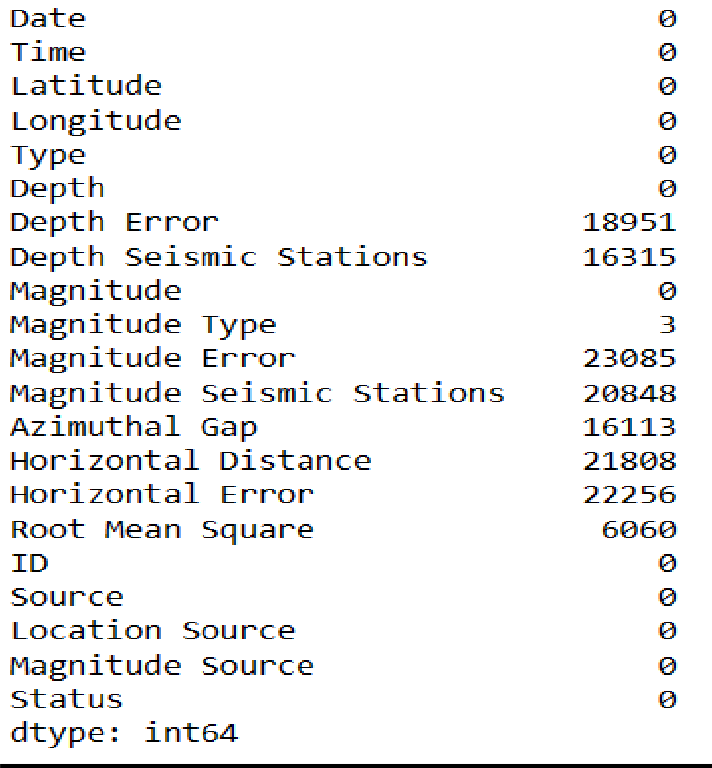
('preprocessor',preprocessor),

])

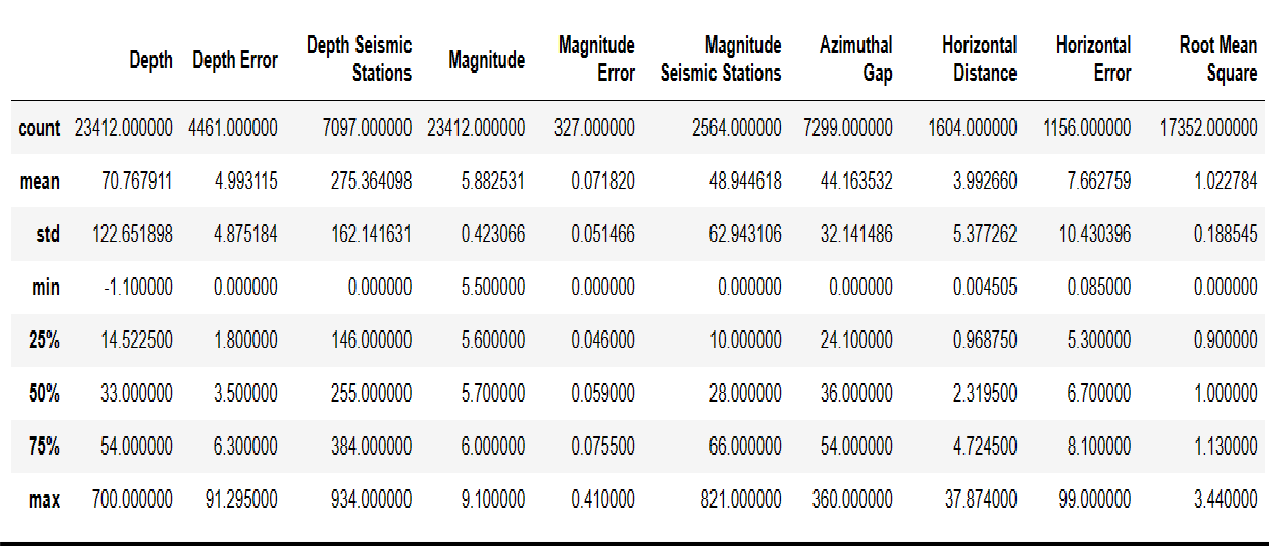
X\_train = model.fit\_transform(X\_train) X\_test = model.transform(X\_test) Print(“PreProcessing Complete!!!”)

# Output:

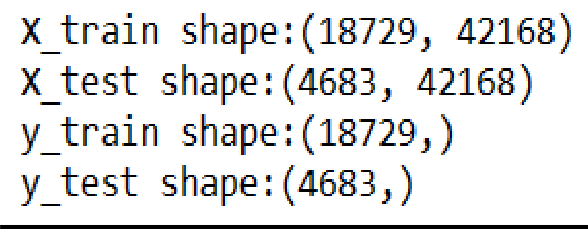
1. **Checking for missing values:**



# Descriptive Statistics:



1. **Data Splitting:**

PreProcessing Complete!!!

# Conclusion:

* + Data preprocessing emerged as a pivot aspect of this process. It involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirements of machine learning algorithms.
  + With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a house price prediction model.