## Earthquake Prediction Model with python

It is well known that if a disaster occurs in one region, it is likely to happen again. Some regions have frequent earthquakes, but this is only a comparative amount compared to other regions.

So, predicting the earthquake with date and time, latitude and longitude from previous data is not a trend that follows like other things, it happens naturally.

 will start this task to create a model for earthquake prediction by importing the necessary python libraries:

Now let’s load and read the dataset. The dataset that I am using here can be easily downloaded [**here**](https://github.com/amankharwal/Website-data/blob/master/database.csv):

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')

Now let’s see the main characteristics of earthquake data and create an object of these characteristics, namely, date, time, latitude, longitude, depth, magnitude:

|  | **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 |

Since the data is random, so we need to scale it based on the model inputs. In this, we convert the given date and time to Unix time which is in seconds and a number. This can be easily used as an entry for the network we have built:

Now, to create the earthquake prediction model, we need to divide the data into Xs and ys which respectively will be entered into the model as inputs to receive the output from the model.

Here the inputs are TImestamp, Latitude and Longitude and the outputs are Magnitude and Depth. I’m going to split the xs and ys into train and test with validation. The training set contains 80% and the test set contains 20%:

**(18727, 3) (4682, 3) (18727, 2) (4682, 3)**

## Neural Network for Earthquake Prediction

Now I will create a neural network to fit the data from the training set. Our neural network will consist of three dense layers each with 16, 16, 2 nodes and reread. Relu and softmax will be used as activation functiOn.

* the earth’s level the earthquake started.
* The magnitude of the earthquake
* Location

Python3

|  |
| --- |
| df.shape |

**Output:**

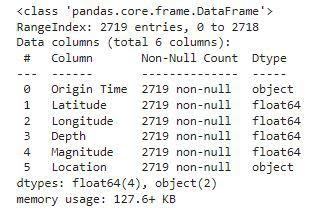
(2719, 6)

Now let’s see which data is present in which type of data format.

Python3

|  |
| --- |
| df.info() |

**Output:**

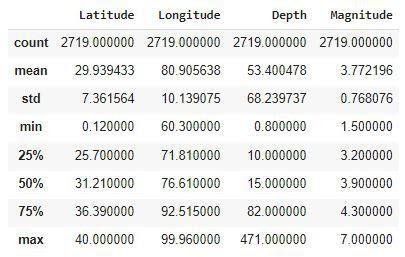


Looking at the descriptive statistical measures also gives us some idea regarding the distribution of the data.

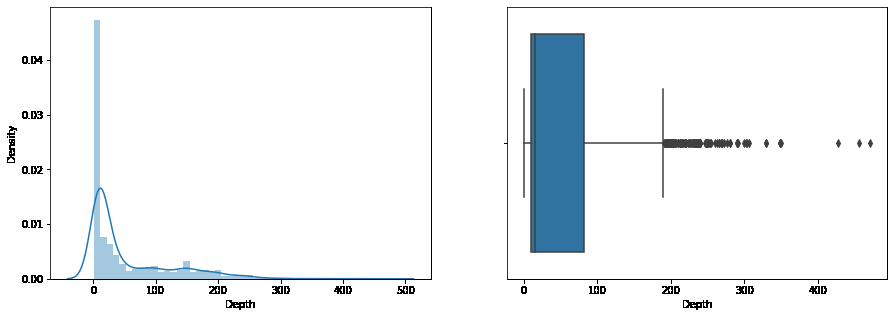
Python3

|  |
| --- |
| df.describe() |

**Output:**



**Output:**

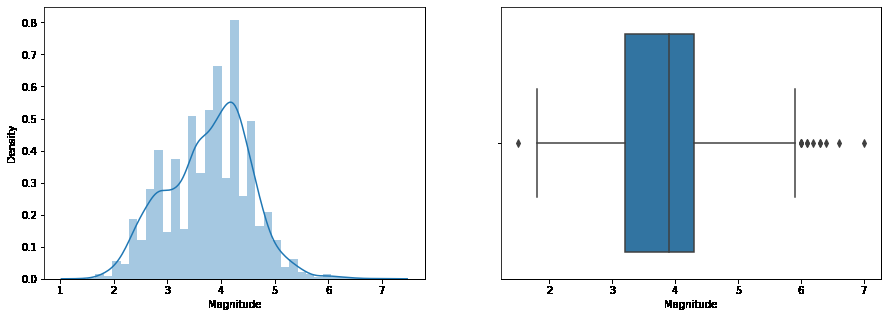


From the distribution graph, it is visible that there are some [outliers](https://www.geeksforgeeks.org/machine-learning-outlier/amp/)that can be confirmed by using the [boxplot](https://www.geeksforgeeks.org/boxplot-using-seaborn-in-python/amp/). But the main point to observe here is that the distribution of the depth at which the earthquake rises is left-skewed.

Python3

|  |
| --- |
| plt.subplots(figsize**=**(15, 5))    plt.subplot(1, 2, 1)  sb.distplot(df['Magnitude'])    plt.subplot(1, 2, 2)  sb.boxplot(df['Magnitude'])    plt.show() |

**Output:**



As we know that many natural phenomena follow a [normal distribution](https://www.geeksforgeeks.org/python-normal-distribution-in-statistics/amp/)and here we can observe that the magnitude of the earthquake also follows a normal distribution.

Python3

|  |
| --- |
| plt.figure(figsize**=**(10, 8))  sb.scatterplot(data**=**df,                 x**=**'Latitude',                 y**=**'Longitude',                 hue**=**'Magnitude')  plt.show() |