# Loading and preprocessing the dataset in Earthquake Prediction model using Python

Loading and preprocessing the dataset in Earthquake Prediction model using Python. Begin building the earthquake prediction model by loading and preprocessing the dataset. This Process involves several key steps to prepare the data for prediction and analysis. Let’s break down the steps outlined in the article.

## Understanding the Dataset:

* Earthquake Prediction data set is an ordered collection of data.
* As we know, a collection of information obtained through observations, measurements, study, or analysis is referred to as data.
* It could include information such as facts, numbers, figures, names, or even basic descriptions of objects.

## Importing Libraries:

* Import the necessary libraries required for buidling the model and data analysis of the earthquakes.

import numpy as np

import pandas as pd

import geopandas as gpd

import matplotlib.pyplot as plt

import seaborn as sns

import folium

from folium import Choropleth

from folium.plugins import HeatMap

import datetime

import os

print(os.listdir("../input"))

## Read the dataset:

* Read the data from csv and also columns which are necessary for the model and the column which needs to be predicted.
* Use the open() function(opens a file and returns a file object as a result) to open the . data file in read-binary mode by passing the file name, and mode 'rb' as arguments to it.
* Use the read() function(reads the specified number of bytes from the file and returns them.

data = pd.read\_csv("../input/database.csv")

data.head()

## Visualization:

* Here, all the earthquakes from the database in visualized on to the world map which shows clear representation of the locations where frequency of the earthquake will be more.
* Analyzing earthquake data using the matplotlib library of Python can provide valuable insights into the frequency, magnitude, and location of earthquakes, which can help in predicting and mitigating their impacts.

from mpl\_toolkits.basemap import Basemap

m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=180,lat\_ts=20,resolution='c')

longitudes = data["Longitude"].tolist()

latitudes = data["Latitude"].tolist()

*#m = Basemap(width=12000000,height=9000000,projection='lcc',*

*#resolution=None,lat\_1=80.,lat\_2=55,lat\_0=80,lon\_0=-107.)*

x,y = m(longitudes,latitudes)

# Feature Engineering:

* Feature Engineeringhelps to derive some valuable features from the existing ones.
* These extra features sometimes help in increasing the performance of the model significantly and certainly help to gain deeper insights into the data.

## Splitting the Data:

* Firstly, split the data into Xs and ys which are input to the model and output of the model respectively.
* Here, inputs are TImestamp, Latitude and Longitude and outputs are Magnitude and Depth. Split the Xs and ys into train and test with validation.
* Training dataset contains 80% and Test dataset contains 20%.

X = final\_data[['Timestamp', 'Latitude', 'Longitude']]

y = final\_data[['Magnitude', 'Depth']]

* Here, we used the RandomForestRegressor model to predict the outputs, we see the strange prediction from this with score above 80% which can be assumed to be best fit but not due to its predicted values.

from sklearn.ensemble import RandomForestRegressor

reg = RandomForestRegressor(random\_state=42)

reg.fit(X\_train, y\_train)

reg.predict(X\_test)

## Neural Network model:

* We build the neural network to fit the data for training set.
* Neural Network consists of three Dense layer with each 16, 16, 2 nodes and relu, relu and softmax as activation function.

from keras.models import Sequential

from keras.layers import Dense

def create\_model(neurons, activation, optimizer, loss):

model = Sequential()

model.add(Dense(neurons, activation=activation, input\_shape=(3,)))

model.add(Dense(neurons, activation=activation))

model.add(Dense(2, activation='softmax'))

model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])

return model

# Seismic Analysis with Python:

## Data Preprocessing:

* **Date parsing:** Parsing date to dtype datetime64(ns)
* **Time Parsing:** Parsing time to dtype timedelta64
* **Adding Attributes:** ” Date\_Time ” and ” Days “

lengths = data["Date"].str.len()

lengths.value\_counts()

Data Analysis:

* Data analysts can also use Python libraries to structure large datasets and make mathematical operations more manageable.
* Pandas, a Python library, offers a data structure called data frame to effectively work with large tables of data.

Time\_series=sns.<a onclick="parent.postMessage({'referent':'.seaborn.lineplot'}, '\*')">lineplot(x=data['Date'].dt.year,y="Magnitude",data=data, color="#ffa600")

Time\_series.set\_title("Time Series Of Earthquakes Over Years", color="#58508d")

Time\_series.set\_ylabel("Magnitude", color="#58508d")

Time\_series.set\_xlabel("Date", color="#58508d")

## Geospatial Analysis:

* Earth’s crust is divided into numerous segments called the tectonic plates.
* These plates are sometimes moved in little or huge amounts due to the geothermal energy generated in the Mantle of the Earth.
* These movements are mostly seen in the tectonic boundaries and are the major cause of natural Earthquakes.

plates = list(tectonic\_plates["plate"].unique())

for plate in plates:

plate\_vals = tectonic\_plates[tectonic\_plates["plate"] == plate]

lats = plate\_vals["lat"].values

lons = plate\_vals["lon"].values

points = list(zip(lats, lons))

indexes = [None] + [<a onclick="parent.postMessage({'referent':'.kaggle.usercode.13873697.58987243.[5527,5530].i'}, '\*')">i + 1 for <a onclick="parent.postMessage({'referent':'.kaggle.usercode.13873697.58987243.[5527,5530].i'}, '\*')">i, <a onclick="parent.postMessage({'referent':'.kaggle.usercode.13873697.58987243.[5527,5530].x'}, '\*')">x in enumerate(points) if <a onclick="parent.postMessage({'referent':'.kaggle.usercode.13873697.58987243.[5527,5530].i'}, '\*')">i < len(points) - 1 and abs(<a onclick="parent.postMessage({'referent':'.kaggle.usercode.13873697.58987243.[5527,5530].x'}, '\*')">x[1] - points[<a onclick="parent.postMessage({'referent':'.kaggle.usercode.13873697.58987243.[5527,5530].i'}, '\*')">i + 1][1]) > 300] + [None]

for i in range(len(indexes) - 1):

folium.<a onclick="parent.postMessage({'referent':'.folium.vector\_layers'}, '\*')">vector\_layers.PolyLine(points[indexes[i]:indexes[i+1]], popup=plate, color="#58508d", fill=False, ).add\_to(tectonic)

The above preprocessing steps are used to predict the earthquake model using python by loading the given dataset are loaded successfully.

## Conclusion:

## In this project we discussed how to load and preprocesses the dataset in Earthquake Prediction Model Using Python. After loaded the dataset, preprocessing is the crucial step in Python, as it help to prepare the text for Analysis. By following the steps outlined in this article, you can preprocess the dataset and apply various analysis methods to detect the Earthquake Prediction.