# EARTHQUAKE PREDICTION MODEL USING PYTHON

TEAM MEMBER

# Project:

950621104079 : M.SANDHIYA

# Phase 2 Submission Document

Earthquake Prediction Model Using Python

# Introduction:

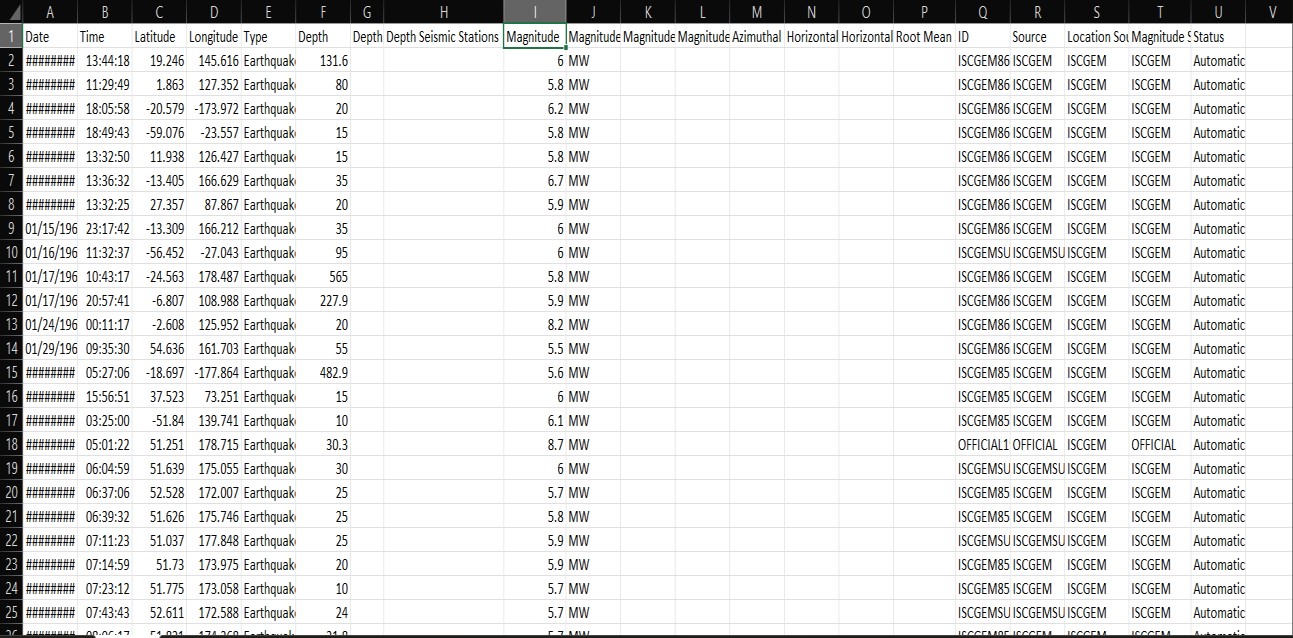
* Create a world map visualization to display earthquake frequency distribution.
* Using tensorflow, keras for library for the earthquake prediction.
* Advanced techniques of the hyperparameter tuning such as GridSearchCV is used for the earthquake prediction.

# Content for Project Phase 2:

Consider advanced techniques such as hyperparameter tuning and feature engineering to improve the prediction model's performance.

**Data Source :**

A good data source for earthquake prediction using machine learning should be accurate time, location, depth, magnitude.

Dataset Link: ( [https://www.kaggle.com/datasets/usgs/earthquake-](https://www.kaggle.com/datasets/usgs/earthquake-database) [database](https://www.kaggle.com/datasets/usgs/earthquake-database)

# Feature Engineering:

* Create new features or transform existing one to capture valuable information.
* Emphasize the impact of engineered features on model performance.
* Explain the process of creating new features or transforming existing ones.

# Advanced Technique:

* Tensorflow : It is a multidimensional array that represents all types of data.
* Sklearn : It supports many supervised and unsupervised learning method such as support vector machine, random foresets, k-means and gradient boosting.
* Keras : It supports various tools and algorithms for data analysis such as classification, regression and clustering.
* GridSearchCV : It help you improve your model’s performance by finding the best hyperparameters for your problem.

# Program:

**Earthquake prediction model**

## Importing Dependencies

import pandas as pd import numpy as np

import matplotlib.pyplot as plt import datetime

import time import sklearn

from sklearn.model\_selection import train\_test\_split, GridSearchCV import calendar

from keras.models import Sequential from keras.layers import Dense import tensorflow as tf

from tensorflow import keras

from keras.wrappers.scikit\_learn import KerasClassifier from mpl\_toolkits.basemap import Basemap

## Loading Dataset

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

## Data Cleaning In [1]:

df.duplicated()

## Out [1]:

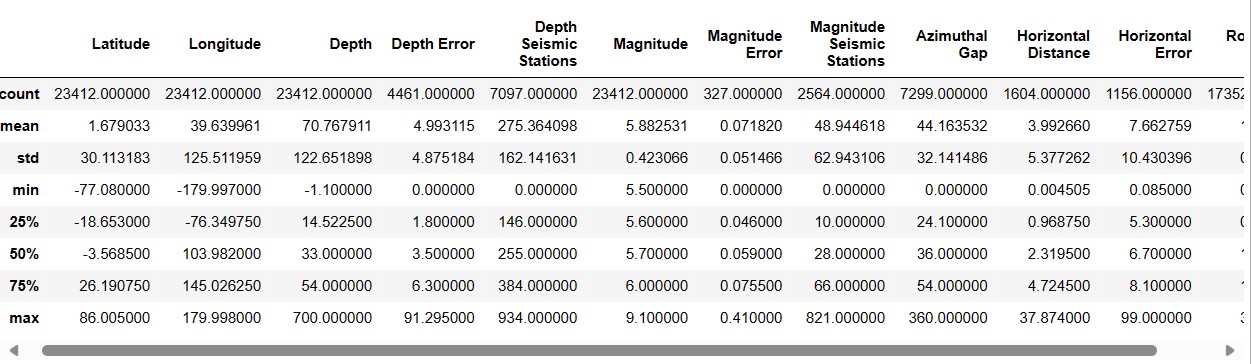
1. False
2. False
3. False

|  |  |
| --- | --- |
| 3 | False |
| 4  ... | False |
| 23407 | False |
| 23408 | False |
| 23409 | False |
| 23410 | False |
| 23411 | False |

## In[2]:

**Out[2]:**

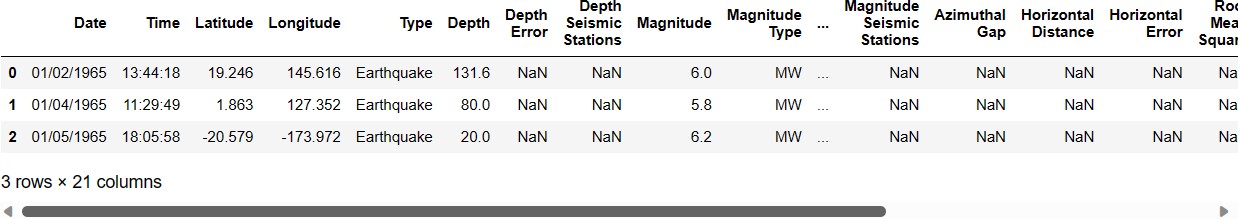
Length: 23409, dtype: bool df.describe()



## In[3]:

df.head(3)

## Out[3]:



**keeping the important columns In[4]:**

df.columns

## Out[4]:

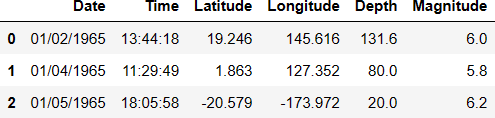
Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth Erro r', 'Depth Seismic Stations', 'Magnitude', 'Magnitude Type', 'Magnitude Error', ' Magnitude Seismic Stations', 'Azimuthal Gap', 'Horizontal Distance', 'Horizonta l Error', 'Root Mean Square', 'ID', 'Source', 'Location Source', 'Magnitude

Source', 'Status'], dtype='object')

## In[5]:

**Out[5]:**

df = df[['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']] df.head(3)



## In[6]:

timestamp = []

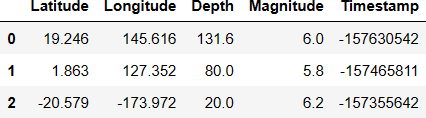
for d, t in zip(df['Date'], df['Time']):

try:

ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S') timestamp.append(calendar.timegm(ts.timetuple()))

except ValueError: timestamp.append('ValueError') timeStamp = pd.Series(timestamp) df['Timestamp'] = timeStamp.values df = df.drop(['Date', 'Time'], axis=1) df = df[df.Timestamp != 'ValueError'] df.head(3)

## Out[6]:



**Splitting the data: In[7]:**

df.columns

## Out[7]:

Index(['Latitude', 'Longitude', 'Depth', 'Magnitude', 'Timestamp'], dtype

='object')

## In[8]:

X = df[['Latitude', 'Longitude', 'Timestamp']] y = df[['Depth', 'Magnitude']]

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

print(X\_train.shape, X\_test.shape, y\_train.shape, X\_test.shape)

## Out[8]:

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

## In[9]:

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=['accur acy'])

return model

## Hyperparameter Tuning:

**In[10]:**

import tensorflow as tf

from tensorflow import keras

from keras.wrappers.scikit\_learn import KerasClassifier model = KerasClassifier(build\_fn=create\_model, verbose=0) param\_grid = {

"neurons": [16],

"batch\_size": [10, 20],

"epochs": [10],

"activation": ['sigmoid', 'relu'],

}

## In[11]:

"optimizer": ['SGD', 'Adadelta'], "loss": ['squared\_hinge']

X\_train = np.asarray(X\_train).astype(np.float32) y\_train = np.asarray(y\_train).astype(np.float32) X\_test = np.asarray(X\_test).astype(np.float32)

y\_test = np.asarray(y\_test).astype(np.float32)

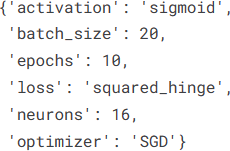
## In[12]:

grid = GridSearchCV(estimator=model, param\_grid=param\_grid,

\_jobs=-1)

\_result = grid.fit(X\_train, y\_train) best\_params = grid\_result.best\_params\_ best\_params

## Out[12]:



**In[13]:**

from mpl\_toolkits.basemap import Basemap

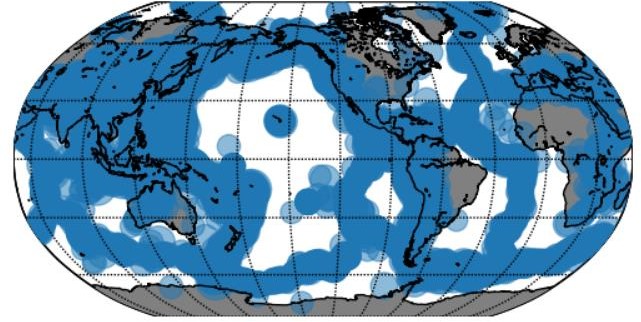
m = Basemap(projection='robin', resolution = 'l', lat\_0=0, lon\_0=-130) m.drawcoastlines()

m.fillcontinents(color = 'gray') m.drawmapboundary() m.drawmeridians(np.arange(0, 360, 30))

m.drawparallels(np.arange(-90, 90, 30))

x,y = m(df['Longitude'].values, df['Latitude'].values) m.scatter(x,y,s=df['Magnitude']\*\*3,alpha=0.5) ax.set\_title('Earthquakes around the world') plt.show()

**Out[13]:**



# Conclusion:

# We will reiterate the impact of these techniques on improving the accuracy and robustness of earthquake prediction.