EARTHQUAKE PREDICTION MODEL USING PYTHON

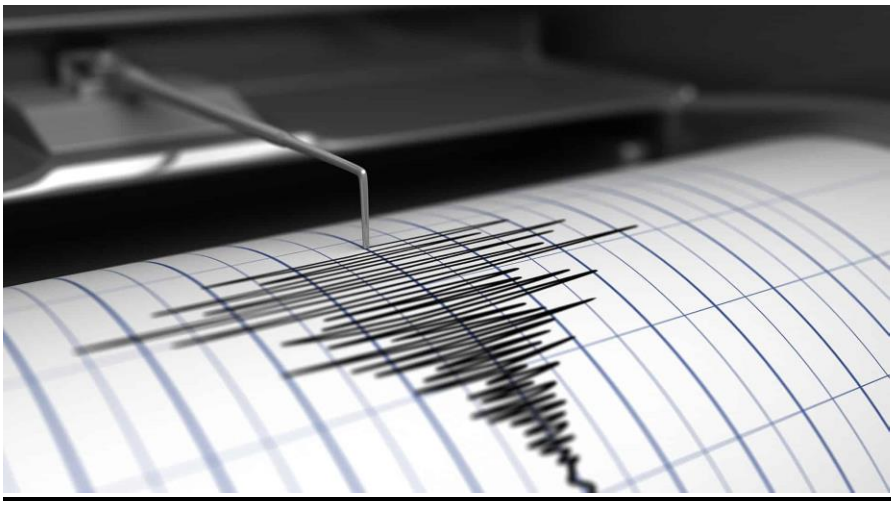
TEAM_MEMBER

950621104097 : S.SOBIKA

Phase 2 Submission Document

Project:

Earthquake Prediction Model Using Python



Introduction:

- Create a world map visualization to display earthquake frequency distribution.
- Split the dataset into a training set and a test set for model validation.
- 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-database)

1	Date	Time	Latitude	Longitude	Type	Depth	Depth	Depth Seismic Stations	Magnitude	Magnitude	Magnitude	Magnitude	Azimuthal	Horizontal	Horizontal	Root Mean	ID	Source	Location Sou	Magnitude S	Status
2	########	13:44:18	19.246	145.616	Earthquak	131.6			6	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
3	########	11:29:49	1.863	127.352	Earthquak	80			5.8	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
4	########	18:05:58	-20.579	-173.972	Earthquak	20			6.2	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
5	########	18:49:43	-59.076	-23.557	Earthquak	15			5.8	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
6	########	13:32:50	11.938	126.427	Earthquak	15			5.8	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
7	########	13:36:32	-13.405	166.629	Earthquak	35			6.7	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
8	########	13:32:25	27.357	87.867	Earthquak	20			5.9	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
9	01/15/196	23:17:42	-13.309	166.212	Earthquak	35			6	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
10	01/16/196	11:32:37	-56.452	-27.043	Earthquak	95			6	MW							ISCGEMSU	ISCGEMSU	ISCGEM	ISCGEM	Automatic
11	01/17/196	10:43:17	-24.563	178.487	Earthquak	565			5.8	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
12	01/17/196	20:57:41	-6.807	108.988	Earthquak	227.9			5.9	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
13	01/24/196	00:11:17	-2.608	125.952	Earthquak	20			8.2	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
14	01/29/196	09:35:30	54.636	161.703	Earthquak	55			5.5	MW							ISCGEM86	ISCGEM	ISCGEM	ISCGEM	Automatic
15	########	05:27:06	-18.697	-177.864	Earthquak	482.9			5.6	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Automatic
16	########	15:56:51	37.523	73.251	Earthquak	15			6	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Automatic
17	########	03:25:00	-51.84	139.741	Earthquak	10			6.1	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Automatic
18	########	05:01:22	51.251	178.715	Earthquak	30.3			8.7	MW							OFFICIAL1	OFFICIAL	ISCGEM	OFFICIAL	Automatic
19	########	06:04:59	51.639	175.055	Earthquak	30			6	MW							ISCGEMSU	ISCGEMSU	ISCGEM	ISCGEM	Automatic
20	########	06:37:06	52.528	172.007	Earthquak	25			5.7	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Automatic
21	########	06:39:32	51.626	175.746	Earthquak	25			5.8	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Automatic
22	########	07:11:23	51.037	177.848	Earthquak	25			5.9	MW							ISCGEMSU	ISCGEMSU	ISCGEM	ISCGEM	Automatic
23	########	07:14:59	51.73	173.975	Earthquak	20			5.9	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Automatic
24	########	07:23:12	51.775	173.058	Earthquak	10			5.7	MW							ISCGEM85	ISCGEM	ISCGEM	ISCGEM	Automatic
25	########	07:43:43	52.611	172.588	Earthquak	24			5.7	MW							ISCGEMSU	ISCGEMSU	ISCGEM	ISCGEM	Automatic

Data Collection and Preprocessing:

- Importing the dataset : Obtain a comprehensive dataset containing relevant features such as time, location, depth.
- Data Preprocessing : Clean the data by handling missing values and outliers.

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]:

```
0       False
1       False
2       False
```

3 False
4 False
...
23407 False
23408 False
23409 False
23410 False
23411 False
Length: 23409, dtype: bool

In[2]:

```
df.describe()
```

Out[2]:

	Latitude	Longitude	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Error	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance	Horizontal Error	Ro
count	23412.000000	23412.000000	23412.000000	4461.000000	7097.000000	23412.000000	327.000000	2564.000000	7299.000000	1604.000000	1156.000000	1735
mean	1.679033	39.639961	70.767911	4.993115	275.364098	5.882531	0.071820	48.944618	44.163532	3.992660	7.662759	
std	30.113183	125.511959	122.651898	4.875184	162.141631	0.423066	0.051466	62.943106	32.141486	5.377262	10.430396	(
min	-77.080000	-179.997000	-1.100000	0.000000	0.000000	5.500000	0.000000	0.000000	0.000000	0.004505	0.085000	(
25%	-18.653000	-76.349750	14.522500	1.800000	146.000000	5.600000	0.046000	10.000000	24.100000	0.968750	5.300000	(
50%	-3.568500	103.982000	33.000000	3.500000	255.000000	5.700000	0.059000	28.000000	36.000000	2.319500	6.700000	
75%	26.190750	145.026250	54.000000	6.300000	384.000000	6.000000	0.075500	66.000000	54.000000	4.724500	8.100000	
max	86.005000	179.998000	700.000000	91.295000	934.000000	9.100000	0.410000	821.000000	360.000000	37.874000	99.000000	3

In[3]:

```
df.head(3)
```

Out[3]:

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	...	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance	Horizontal Error	Roo Mea Squar
0	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	...	NaN	NaN	NaN	NaN	Na
1	01/04/1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	...	NaN	NaN	NaN	NaN	Na
2	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	...	NaN	NaN	NaN	NaN	Na

3 rows × 21 columns

keeping the important columns

In[4]:

```
df.columns
```

Out[4]:

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')

In[5]:

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

Out[5]:

	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

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]:

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	-157630542
1	1.863	127.352	80.0	5.8	-157465811
2	-20.579	-173.972	20.0	6.2	-157355642

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'],
```

```
        "optimizer": ['SGD', 'Adadelta'],
         "loss": ['squared_hinge']
}
```

In[11]:

```
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]:

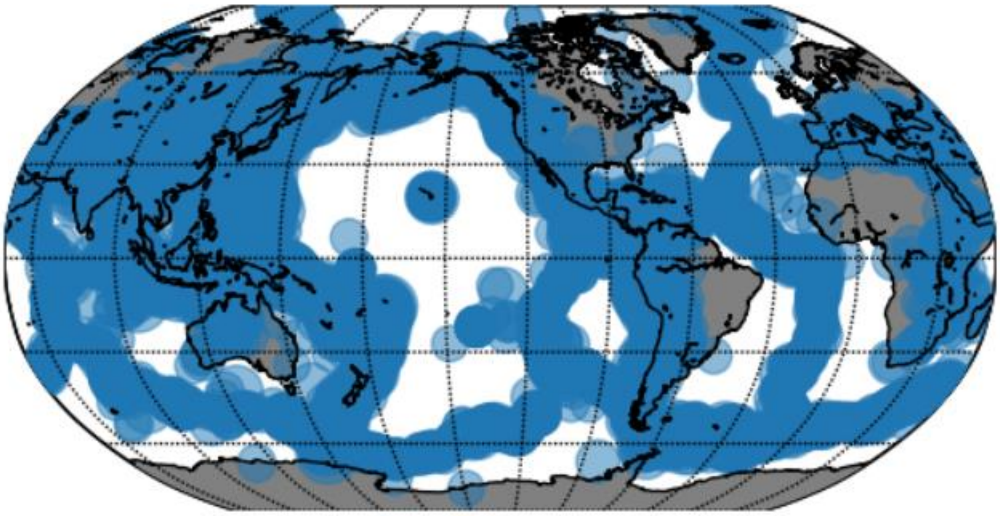
```
{'activation': 'sigmoid',
 'batch_size': 20,
 'epochs': 10,
 'loss': 'squared_hinge',
 'neurons': 16,
 'optimizer': 'SGD'}
```

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:

In the phase 2 conclusion, we will summarize the key findings and insights from the advanced regression techniques. We will reiterate the impact of these techniques on improving the accuracy and robustness of earthquake prediction.