<u>Project title - Earthquake Prediction Model using Python</u> <u>Phase 2 – Innovation</u>

Introduction:

- The data cleaning and data analysis mentioned in the phase 1 design thinking steps are implemented in this phase 2 innovation file.
- The workspace used is "Google colab".

Colab Notebook link: Notebook Link

Importing the dataset:

df.head(5)											
	Date	Time	Latitude	Longitude	Туре	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	
0	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	
1	01/04/1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	
2	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	
3	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	
4	01/09/1965	13:32:50	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW	
5 rows × 21 columns											

Feature Engineering:

Data cleaning and analysis:

Firstly the data is analysed before applying feature engineering.

Statistical information about the dataset is given by

df.describe()

#Printing the columns of the dataset

df.columns

#Printing the shape of the dataset

df.shape

#Checking for duplicated values

df.duplicated()

#Printing the numerical and categorical features from the dataset

```
[71] #Printing the numerical and categorical features
    # Categorical columns
    cat_col = [col for col in df.columns if df[col].dtype == 'object']
    print('Categorical columns :',cat_col)
    # Numerical columns
    num_col = [col for col in df.columns if df[col].dtype != 'object']
    print('Numerical columns :',num_col)

Categorical columns : ['Date', 'Time', 'Type', 'Magnitude Type', 'ID', 'Source', 'Location Source', 'Magnitude Source', 'Status']
    Numerical columns : ['Latitude', 'Longitude', 'Depth', 'Depth Error', 'Depth Seismic Stations', 'Magnitude', 'Magnitude Error', 'Magnitude
```

#Printing the number of unique values in the numerical data values

```
0
    df[num_col].nunique()

→ Latitude

                                 20676
    Longitude
                                 21474
    Depth
                                  3485
    Depth Error
                                   297
    Depth Seismic Stations
                                  736
    Magnitude
                                   64
    Magnitude Error
                                  100
    Magnitude Seismic Stations
                                  246
    Azimuthal Gap
                                 1109
                                 1448
    Horizontal Distance
    Horizontal Error
                                   186
                                   190
    Root Mean Square
    dtype: int64
```

Printing the number of unique values in the categorical data values

```
[73] #Checking number of unique values in categorical columns
     df[cat col].nunique()
     Date
                         12401
     Time
                         20472
     Type
     Magnitude Type
                            10
     ID
                         23412
                           13
     Location Source
                            48
     Magnitude Source
                            24
     Status
                             2
     dtype: int64
```

Handling missing values:

#Finding missing values in our dataset

```
[74] #Finding number of missing values in each column
         print(df.isnull().sum())
        Date
                                                                0
        Time
        Latitude
        Longitude
                                                               0
                                                              0
        Type
        Depth
                                                               0
        Depth Error
                                                       18951
       Depth Error 18951
Depth Seismic Stations 16315
Magnitude 0
Magnitude Type 3
Magnitude Error 23085
Magnitude Seismic Stations 20848
Azimuthal Gap 16113
Horizontal Distance 21808
Horizontal Error 22256
Root Mean Square 6060
        Root Mean Square
                                                        6060
                                                             0
        Source
                                                             0
        Location Source
                                                             0
                                                               0
        Magnitude Source
                                                               0
        Status
         dtype: int64
```

#Finding the percentage of missing values in each column.

```
_{	t 0s} [75] #Finding the percentage of missing values in each column
          miss_percent = (df.isnull().sum()/df.shape[0])*100
          print(round(miss_percent,2))
         Date
                                                 0.00
          Time
                                                 0.00
          Latitude
                                                 0.00
          Longitude
                                                 0.00
          Туре
                                                 0.00
          Depth
                                                 0.00
          Depth Error
         Depth Seismic Stations 69.69
Magnitude 0.00
Magnitude Type 0.01
Magnitude Error 98.60
         Magnitude Seismic Stations 89.05
         Azimuthal Gap 68.82
Horizontal Distance 93.15
Horizontal Error 95.06
Root Mean Square 25.88
          TD
                                                 0.00
          Source
                                                 0.00
         Location Source
Magnitude Source
                                                 0.00
                                                 0.00
          Status
                                                 0.00
          dtype: float64
```

Feature selection:

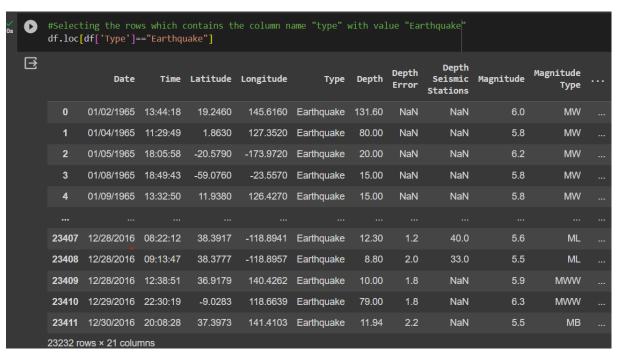
Since we are interested only in predicting earthquake ,we restrict our instances to contain only the type attribute with earthquake value only.

```
#Checking the number of instances in each class of the type attribute

df['Type'].value_counts()

Earthquake 23232
Nuclear Explosion 175
Explosion 4
Rock Burst 1
Name: Type, dtype: int64
```

Applying feature engineering:



Scaling down the features by dropping unwanted columns:

Independent features and the target feature:

```
#Significant columns
df.columns

Index(['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude'], dtype='object')
```

Checking for null values again:

```
#Checking for null values after feature engineering df.isnull().sum()

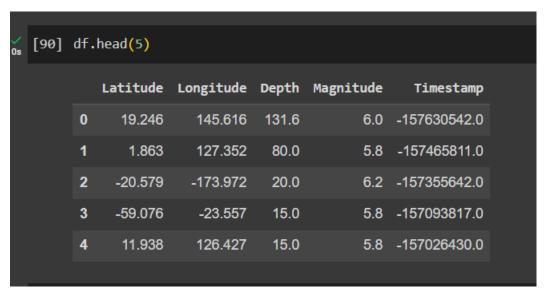
Date 0
Time 0
Latitude 0
Longitude 0
Depth 0
Magnitude 0
dtype: int64
```

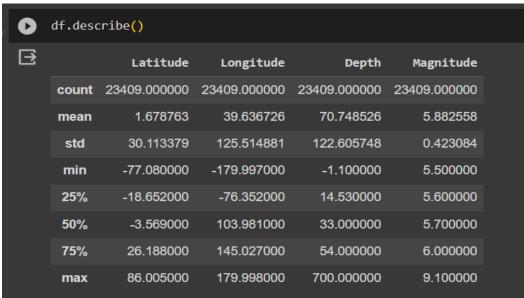
Feature Creation technique:

- We convert given Date and Time to Unix time which is in seconds and a numeral.
- This can be easily used as input for the network we built.

```
# We convert given Date and Time to Unix time which is in seconds and a numeral.
        import datetime
        import time
        timestamp = []
        for d, t in zip(df['Date'], df['Time']):
                ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S')
                timestamp.append(time.mktime(ts.timetuple()))
            except ValueError:
                timestamp.append('ValueError')
        timeStamp = pd.Series(timestamp)
        df['Timestamp'] = timeStamp.values
[88] df.drop(['Date', 'Time'], axis=1,inplace=True)
        df = df[df.Timestamp != 'ValueError']
        print(df.head(5))
           Latitude Longitude Depth Magnitude
                                                    Timestamp
                     145.616 131.6 6.0 -157630542.0
            19.246
                    127.352 80.0
-173.972 20.0
-23.557 15.0
                                            5.8 -157465811.0
             1.863
            -20.579
                                            6.2 -157355642.0
           -59.076
                                            5.8 -157093817.0
            11.938
                      126.427 15.0
                                             5.8 -157026430.0
```

Data Analysis after feature engineering:







Hyperparameter Tuning:

- Number of neurons
- Activation function
- Number of hidden layers
- Batch size
- Epochs
- Optimizers
- Learning rate
- Callbacks
- Dropout layer
- ➤ The hyperparameters to tune are the number of neurons, activation function, optimizer, learning rate, batch size, and epochs.
- The second step is to tune the number of layers.
- The first hyperparameter to tune is the number of neurons in each hidden layer.
- ➤ The number of neurons should be adjusted to the solution complexity. For example, range is set to be from 10 to 100.
- ➤ The activation function decides how to compute the input values of a layer into output values.
- ➤ Optimizer is very important to achieve the possible highest accuracy or minimum loss. There are 7 optimizers to choose from.
- ➤ One of the hyperparameters in the optimizer is the learning rate.
- ➤ Learning rate controls the step size for a model to reach the minimum loss function.
- ➤ The number of times a whole dataset is passed through the neural network model is called an epoch.
- ➤ Suitable number of epochs between 20 to 100. An example is as follows:

```
params_nn ={
    'neurons': (10, 100),
    'activation':(0, 9),
    'optimizer':(0,7),
    'learning_rate':(0.01, 1),
    'batch_size':(200, 1000),
    'epochs':(20, 100)
}
```

Tuning the layers using regularization technique:

- ➤ Inserting regularization layers in a neural network can help prevent overfitting.
- The dropout layer, as its name suggests, randomly drops a certain number of neurons in a layer. The dropped neurons are not used anymore. The rate of how much percentage of neurons to drop is set in the dropout rate.

An example

```
params_nn2 ={
    'neurons': (10, 100),
    'activation':(0, 9),
    'optimizer':(0,7),
    'learning_rate':(0.01, 1),
    'batch_size':(200, 1000),
    'epochs':(20, 100),
    'layers1':(1,3),
    'normalization':(0,1),
    'dropout':(0,1),
    'dropout_rate':(0,0.3)
}
```

Ensembling Techniques:

- Neural network models are nonlinear and have a high variance, which can be frustrating when preparing a final model for making predictions.
- Ensemble learning combines the predictions from multiple neural network models to reduce the variance of predictions and reduce generalization error.
- ➤ K-Fold cross validation ensembling is best suited for the neural networks model. Various other techniques are as follows
 - Varying Training Data
 - k-fold Cross-Validation Ensemble
 - · Bootstrap Aggregation (bagging) Ensemble
 - Random Training Subset Ensemble
 - Varying Models
 - · Multiple Training Run Ensemble
 - Hyperparameter Tuning Ensemble
 - Snapshot Ensemble
 - Horizontal Epochs Ensemble
 - Vertical Representational Ensemble
 - · Varying Combinations
 - Model Averaging Ensemble
 - Weighted Average Ensemble
 - · Stacked Generalization (stacking) Ensemble
 - Boosting Ensemble
 - Model Weight Averaging Ensemble