Prediction Model for Earthquake

By

Bhashwitha Kolluri – 11526264

Requirements:

The goal of this project is to use historical data to estimate the magnitude of an earthquake for a region specified by the user.   Earthquake prediction can be done in a variety of methods, according to geologists. The results have so far proved successful in predicting where an earthquake is more likely to occur, but when it will occur is still being investigated. This program will prompt the user to enter the latitude and longitude of the region in which they wish to know if an earthquake is likely to occur. It will use the data set and neural networks to correlate data and create predictions based on these coordinates and visual techniques to display the exact or approximate place of earthquakes.

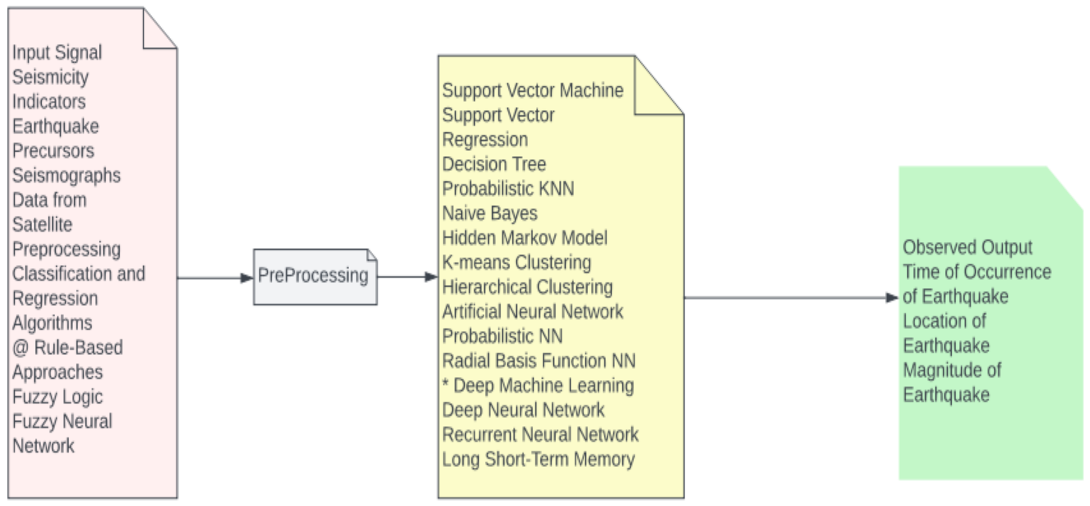
A dataset which can be used for training and the testing the model.

In terms of software and packages:

- Jupyter Notebook
- Python 3.6 or above
- Pandas for creating dataframes and data analysis
- Numpy for n-dimensional array
- Seaborn for visualization
- Sklearn for ML libraries
- Matplotlib for visualization
- Keras for creating deep models
- Support vector regressor and hybrid neural networks

STATE DIAGRAMS AND FLOW CHARTS OF THE ALGORITHMS USED IN PREDICTING EARTHQAUAKE MODEL USING AI.

A general earthquake prediction model



The seismicity indicators, which are derived from the earthquake database, may be these characteristics. There are certain earthquake precursors that occurred a few days prior to the actual earthquake. However, no earthquake is ever confirmed by these antecedents. Some of the indicators of an impending earthquake include radon gas concentration, changes in soil temperature, and unusual cloud formation. P-waves and S-waves from the seismograph can be identified, allowing earthquake prediction. Some nations utilize specialized satellites to track characteristics related to earthquakes, which aids in the discovery of earthquake precursors. The prediction model uses this data as an input signal. The missing values are then removed from the data, and it is then transformed into a format compatible with the classification and regression techniques.

AI FOR EARTHQUAKE PREDICTION

FUZZY NEURAL NETWORK (FNN):

We refer to a system as a neuro-fuzzy system when fuzzy networks are represented as artificial neural networks so they can be tuned using backpropagation or genetic algorithm (GA) (NFS). The Mamdani methodology, developed by Ebhasim Mamdani, is one method for putting this theory into practice. This method requires that the system's input and output both be fuzzy quantities. It is a great model for human inference systems since it has a straightforward min-max operations structure. Humans can grasp this model, but as the number of input rules rises, so does its complexity. This model employs five prediction layers, listed as follows:

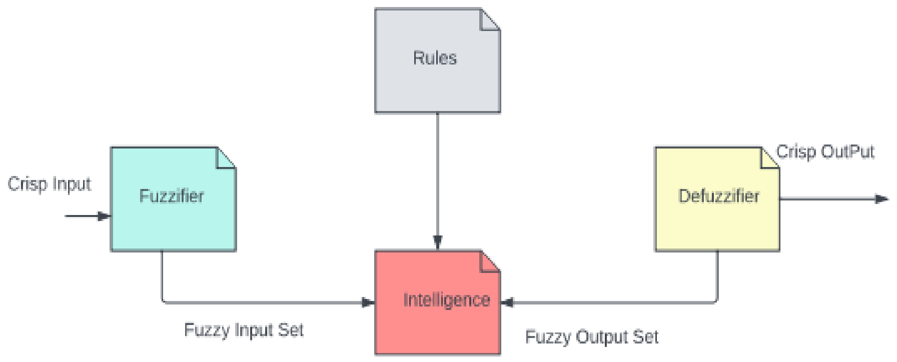
1) **Fuzzification layer:** The membership value of the input vector made up of features is determined in the fuzzification layer. The Gaussian function is typically chosen to calculate the membership value.

2) **Fuzzy inference layer:** By multiplying the membership values, the fuzzy rules in the inference layer are triggered based on the input vector.

3) **Implication layer**: Consequent membership functions are computed in the implication layer based on their strength.

4) **Aggregation layer**: The firing strength multiplied with the ensuing parameters are added up in the aggregation layer.

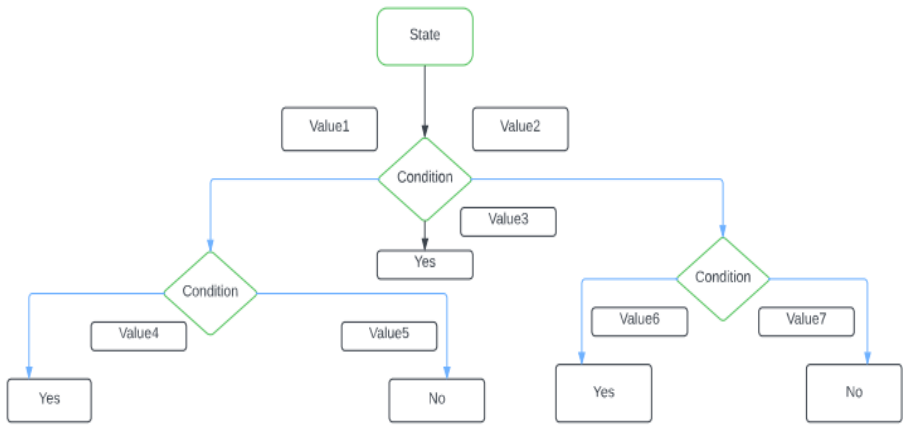
5) **Defuzzification layer**: Defuzzification, which employs the center of the area method, produces the final crisp result.



Fuzzy Logic architecture.

In Fuzzy logic, the crisp input is fuzzified and compared with the rules to create a crisp output.

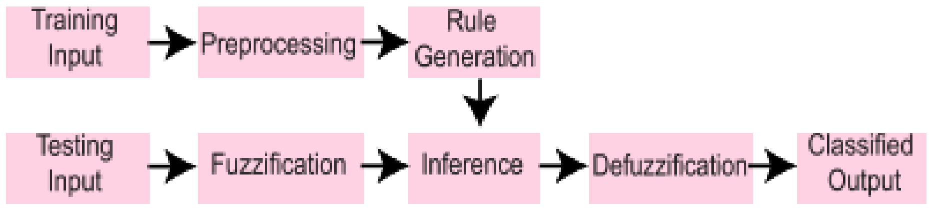
Classification process of the SVM algorithm and Decision-making process of the Decision Tree



This statistical classifier creates decision trees based on information gain, with the highest normalized gain chosen as the splitting criterion. This algorithm, which can handle continuous variables, discrete variables, and missing values, is an upgraded version of the Iterative Dichotomiser 3 algorithm. By examining the training set to categorize the test set, it creates a classifier. To combat overfitting, this algorithm supports pruning trees. Here, a leaf node can take the place of a subtree. Small adjustments to a dataset can cause the decision tree to shift in the meantime. Even with a noisy dataset, this approach is simple to use and performs well. Figure 1 depicts how DT algorithms draw conclusions from the data based on conditions.

RULE-BASED APPROACHES

Some rules are defined in rule-based approaches to earthquake prediction from the knowledge base or from expert opinion. In order to compare the input signals with the rules, some membership functions fuzzify the signals. To obtain the true output from this comparison, the output is defuzzified. This procedure is shown in Fig. 8, where the training and testing data flow in a separate direction to obtain an earthquake prediction. The studies are classified into two groups: studies on the features of earthquakes for rule-based techniques, and studies on the prediction of earthquakes and after shocks for rule-based approach.

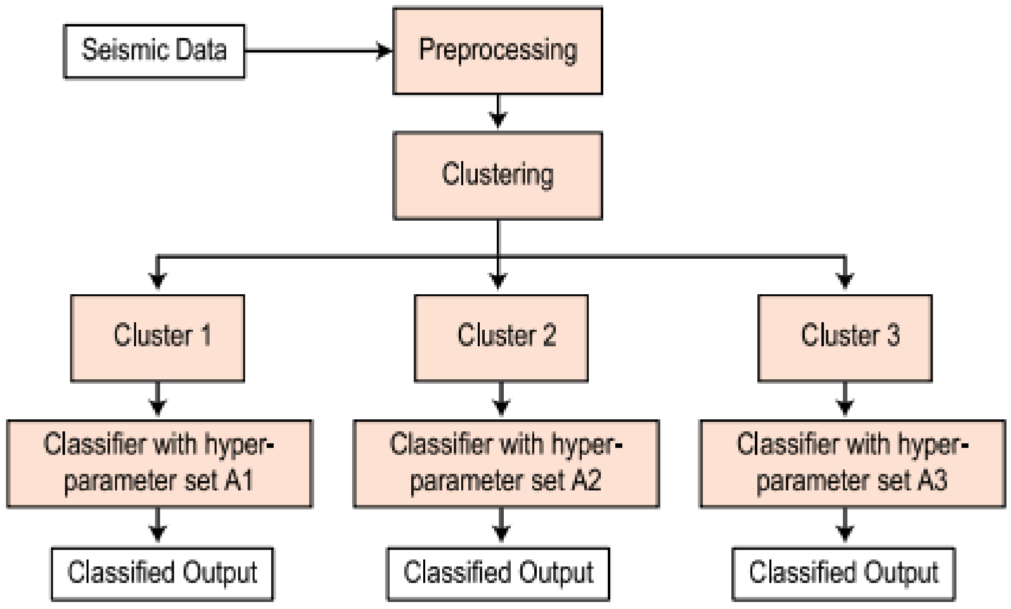


Prediction process of rule-based approaches.

Usually, the dataset is divided into training and testing samples. Based on the training data, rules are generated. The testing samples are fuzzified and compared with the rules to infer an output.

Clustering Models

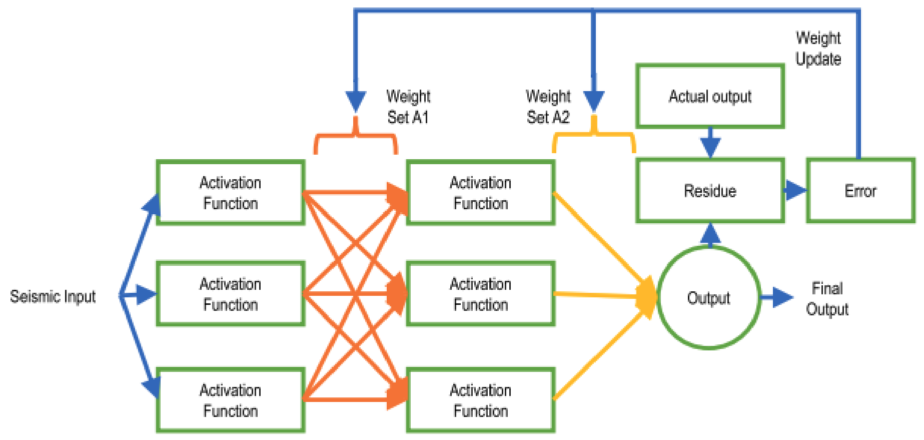
The clustering layer receives the processed seismic data as input and splits it into several groupings. Here are three clusters that were determined using the seismic data: Cluster 1, Cluster 2, and Cluster 3. In order to create the required result, each cluster is classed using several iterations of the classification algorithm based on its properties.


Seismic Data
Preprocessing
Clustering
Cluster 1
Cluster 2
Cluster 3
Classifier with hyper-parameter set A1
Classifier with hyper-parameter set A2
Classifier with hyper-parameter set A3
Classified Output
Classified Output
Classified Output

Earthquake prediction process using clustering approaches.

Artificial Neural Network

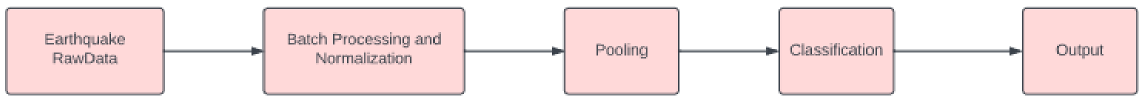
The network's seismic inputs are processed by activation functions to provide an output that is multiplied by weights. The error, which is used to modify the weights so that earthquake predictions are more accurate, is the difference between the final output and the actual output.



Earthquake prediction process of a Artificial Neural Network-based model

Deep Learning

The Deep Neural Networks uses multiple pooling, batch processing, dropout layers to produce features. It can generate sophisticated features which are very difficult to calculate by hand. Based on these features, an earthquake is predicted.



Prediction process of an earthquake using Deep Learning-based approaches.

PSEUDOCODE:

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.

In [8]:

```
from mpl_toolkits.basemap import Basemap

m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=180,lat_ts=2
0,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)
```

In [9]:

```
fig = plt.figure(figsize=(12,10))
plt.title("All affected areas")
m.plot(x, y, "o", markersize = 2, color = 'blue')
m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
m.drawmapboundary()
m.drawcountries()
plt.show()
```

Neural Network model

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

In [16]:

```
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
```

Using TensorFlow backend.

In this, we define the hyperparameters with two or more options to find the best fit.

In [17]:

```
from keras.wrappers.scikit_learn import KerasClassifier

model = KerasClassifier(build_fn=create_model, verbose=0)

# neurons = [16, 64, 128, 256]
neurons = [16]
# batch_size = [10, 20, 50, 100]
batch_size = [10]
epochs = [10]
# activation = ['relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear', 'exponential']
activation = ['sigmoid', 'relu']
# optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
optimizer = ['SGD', 'Adadelta']
loss = ['squared_hinge']

param_grid = dict(neurons=neurons, batch_size=batch_size, epochs=epochs, activation=activatio
n, optimizer=optimizer, loss=loss)
```

Here, we find the best fit of the above model and get the mean test score and standard deviation of the best fit model.

In [18]:

```
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)

grid_result = grid.fit(X_train, y_train)


print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))

means = grid_result.cv_results_['mean_test_score']

stds = grid_result.cv_results_['std_test_score']

params = grid_result.cv_results_['params']

for mean, stdev, param in zip(means, stds, params):

    print("%f (%f) with: %r" % (mean, stdev, param))
```

References:

1. https://thecleverprogrammer.com/2020/11/12/earthquake-prediction-model-with-machine-learning/

2. https://scholarworks.calstate.edu/downloads/fj236210v

3. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9218936

4. https://www.quantamagazine.org/artificial-intelligence-takes-on-earthquake-prediction-20190919/