**EARTHQUAKE PREDICTION MODEL USING PYTHON**

Name: Yokesh.S

Reg.No: 511721104015

**Introduction:**

* It is well known that if a disaster occurs in one region, it is likely to happen again. Some regions have frequent earthquakes, but this is only a comparative amount compared to other regions.

* So, predicting the earthquake with date and time, latitude and longitude from previous data is not a trend that follows like other things, it happens naturally.



So we will be predicting the earthquake from Date and Time, Latitude, and Longitude from previous data is not a trend that follows like other things. It is naturally occurring.

**Importing Libraries:**

1. **import** numpy as np
2. **import** pandas as pd
3. **import** matplotlib.pyplot as plt
5. **import** os
6. print(os.listdir("../input"))

**Output:**

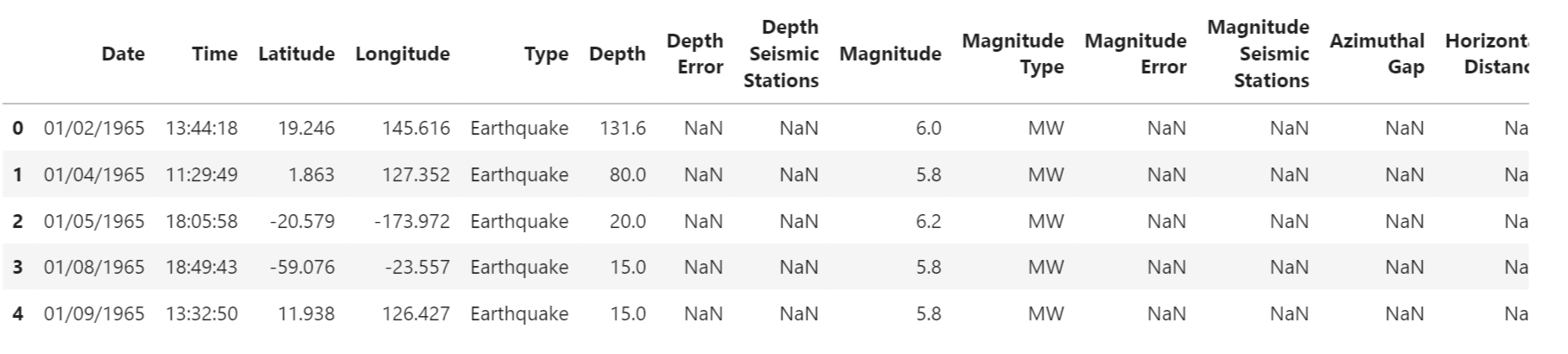
Earthquake Prediction Using Machine Learning

**Read the Dataset:**

Now we will read the dataset and look for the various features in the dataset.

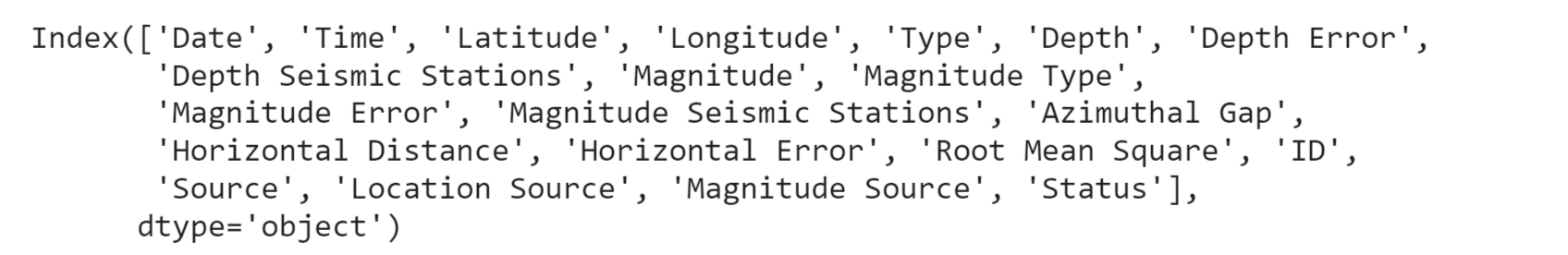
1. data = pd.read\_csv("../input/database.csv")
2. data.head()

**Output:**



1. Data.columns

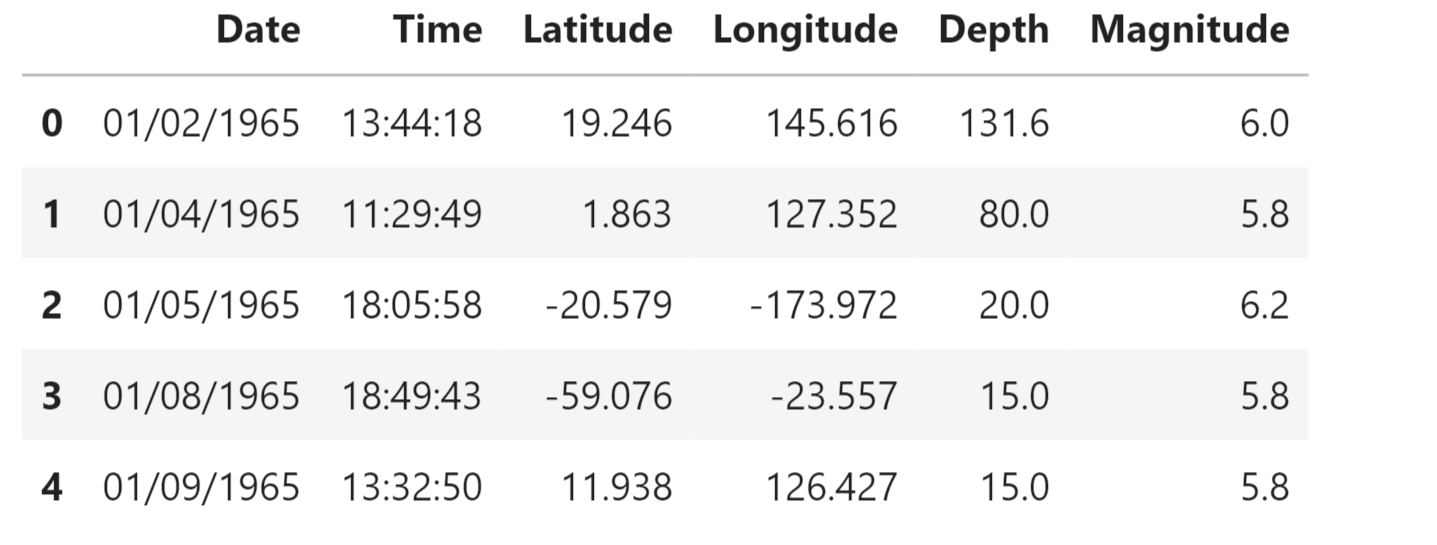
**Output:**



We need to select the features that will be useful for our prediction.

1. data = data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']]
2. data.head()

**Output:**

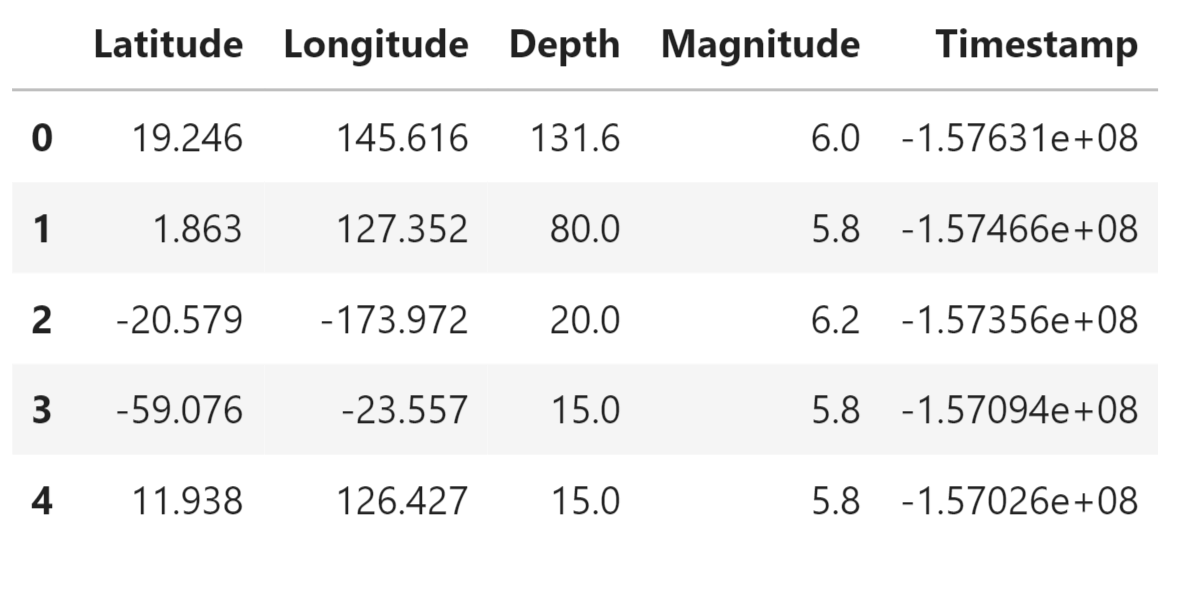


**We will try to frame the time and place of the earthquake that has happened in the past on the world map.**

1. **import** datetime
2. **import** time
4. timestamp = []
5. **for** d, t in zip(data['Date'], data['Time']):
6. **try**:
7. ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S')
8. timestamp.append(time.mktime(ts.timetuple()))
9. except ValueError:
10. # print('ValueError')
11. timestamp.append('ValueError')

1. timeStamp = pd.Series(timestamp)
2. data['Timestamp'] = timeStamp.values
3. final\_data = data.drop(['Date', 'Time'], axis=1)
4. final\_data = final\_data[final\_data.Timestamp != 'ValueError']
5. final\_data.head()

**Output:**

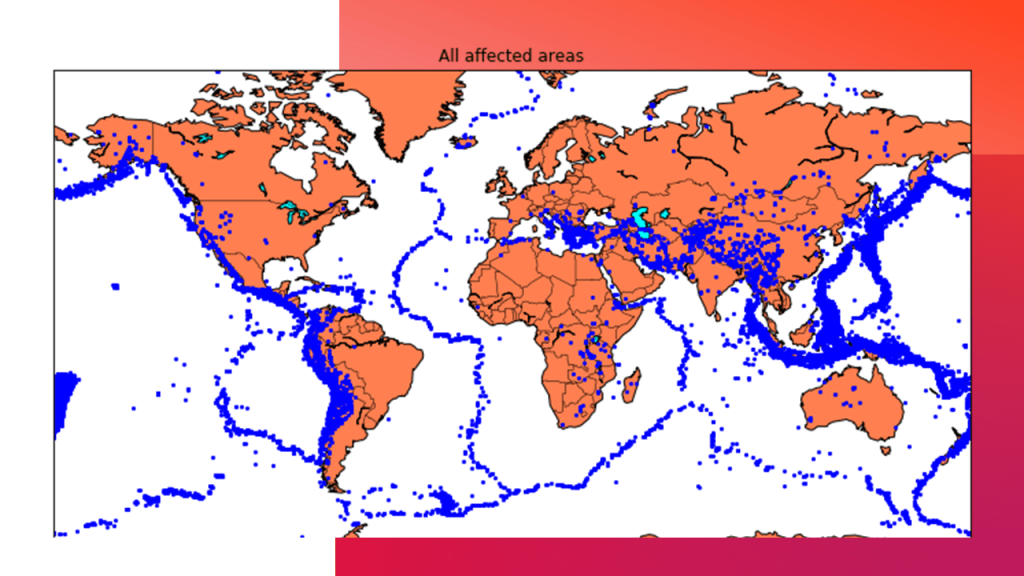


**Data Visualization:**

Now, before we create the earthquake prediction model, let’s visualize the data on a world map that shows a clear representation of where the earthquake frequency will be more:

**Code:**

|  |  |
| --- | --- |
|  | 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) |
|  |  |
|  | 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() |



**Splitting the Dataset:**

Now, to create the earthquake prediction model, we need to divide the data into Xs and ys which respectively will be entered into the model as inputs to receive the output from the model.

Here the inputs are TImestamp, Latitude and Longitude and the outputs are Magnitude and Depth. I’m going to split the xs and ys into train and test with validation. The training set contains 80% and the test set contains 20%:

**Code:**

|  |  |
| --- | --- |
|  | X = final\_data[['Timestamp', 'Latitude', 'Longitude']] |
|  | y = final\_data[['Magnitude', 'Depth']] |
|  | from sklearn.cross\_validation import train\_test\_split |
|  |  |
|  | X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) |
|  | *print(X\_train.shape, X\_test.shape, y\_train.shape, X\_test.shape)* Neural Network for Earthquake Prediction: A neural network model can be employed to forecast earthquakes by examining  diverse elements and trends in seismic data. This model harnesses the capabilities  of neural networks, which draw inspiration from the neural connections of the  human brain, to analyze intricate data and reveal hidden relationships and patterns.  By training the neural network on historical earthquake data,  it can acquire the ability to identify precursor signals  and patterns that indicate the probability of an upcoming earthquake.  **Code:**  from keras.models **import** Sequential   1. from keras.layers **import** Dense 3. def create\_model(neurons, activation, optimizer, loss): 4. model = Sequential() 5. model.add(Dense(neurons, activation=activation, input\_shape=(3,))) 6. model.add(Dense(neurons, activation=activation)) 7. model.add(Dense(2, activation='softmax')) 9. model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy']) 11. **return** model 12. from keras.wrappers.scikit\_learn **import** KerasClassifier 14. model = KerasClassifier(build\_fn=create\_model, verbose=0) 16. # neurons = [16, 64, 128, 256] 17. neurons = [16] 18. # batch\_size = [10, 20, 50, 100] 19. batch\_size = [10] 20. epochs = [10] 21. # activation = ['relu', 'tanh', 'sigmoid', 'hard\_sigmoid', 'linear', 'exponential'] 22. activation = ['sigmoid', 'relu'] 23. # optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam'] 24. optimizer = ['SGD', 'Adadelta'] 25. loss = ['squared\_hinge'] 27. param\_grid = dict(neurons=neurons, batch\_size=batch\_size, epochs=epochs, 28. activation=activation, optimizer=optimizer, loss=loss) 29. grid = GridSearchCV(estimator=model, param\_grid=param\_grid, n\_jobs=-1) 30. grid\_result = grid.fit(X\_train, y\_train) 32. print("Best: %f using %s" % (grid\_result.best\_score\_, grid\_result.best\_params\_)) 33. means = grid\_result.cv\_results\_['mean\_test\_score'] 34. stds = grid\_result.cv\_results\_['std\_test\_score'] 35. params = grid\_result.cv\_results\_['params'] 36. **for** mean, stdev, param in zip(means, stds, params): 37. print("%f (%f) with: %r" % (mean, stdev, param))   **Output:**  *Earthquake Prediction Using Machine Learning*   1. model = Sequential() 2. model.add(Dense(16, activation='relu', input\_shape=(3,))) 3. model.add(Dense(16, activation='relu')) 4. model.add(Dense(2, activation='softmax')) 6. model.compile(optimizer='SGD', loss='squared\_hinge', metrics=['accuracy']) 7. model.fit(X\_train, y\_train, batch\_size=10, epochs=20, verbose=1, 8. validation\_data=(X\_test, y\_test))   **Output:**  *Earthquake Prediction Using Machine Learning*   1. [test\_loss, test\_acc] = model.evaluate(X\_test, y\_test) 2. print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test\_loss, test\_acc))   **Output:**  *Earthquake Prediction Using Machine Learning*  Isn't it amazing that we got an accuracy of 92%.  We can say the neural network is one of the best models to predict earthquakes  that can be used in future.  **Conclusion:**  Understanding earthquakes and effectively responding to them remains a complex  and challenging task, even with the latest technological advancements. However,  leveraging the capabilities of machine learning can greatly enhance our  comprehension of seismic events. By employing machine learning techniques to  analyze seismic data, we can uncover valuable insights and patterns that contribute  to a deeper understanding of earthquakes. These insights can subsequently inform  more effective strategies for mitigating risks and responding to seismic events.  As we head towards the future, we might see new technologies that will precisely  predict the place and time of the earthquake that will happen. |