|  |  |
| --- | --- |
|  |  |

Bottom of Form

|  |
| --- |
| **Earthquake Prediction Using Machine Learning** Machine learning has the ability to advance our knowledge of earthquakes and enable more accurate forecasting and catastrophe response. It's crucial to remember that developing accurate and dependable prediction models for earthquakes still needs more study as it is a complicated and difficult topic.  In order to anticipate earthquakes, machine learning may be used to examine seismic data trends. Seismometers capture seismic data, which may be used to spot changes to the earth's surface, like seismic waves brought on by earthquakes. Machine learning algorithms may utilize these patterns to forecast the risk of an earthquake happening in a certain region by studying these patterns and learning to recognize key traits that are linked to seismic activity.  So we will be predicting the earthquake fromDate and Time, Latitude, and Longitude from previous data is not a trend that follows like other things. It is naturally occurring.  **Code:** **Importing Libraries** **import** numpy as np  **import** pandas as pd  **import** matplotlib.pyplot as plt **Read the Dataset** Now we will read the dataset and look for the various features in the dataset.   1. data = pd.read\_csv(“database.csv") 2. data.head()   **Output:**  Earthquake Prediction Using Machine Learning   1. data.columns   **Output:**  Earthquake Prediction Using Machine Learning  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:**  Earthquake Prediction Using Machine Learning  We will try to frame the time and place of the earthquake that has happened in the past on the world map.  **import** datetime  **import** time  timestamp = []  **for** d, t in zip(data['Date'], data['Time']):  **try**:          ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S')          timestamp.append(time.mktime(ts.timetuple()))      except ValueError:   print('ValueError')  timestamp.append('ValueError')  timeStamp = pd.Series(timestamp)  data['Timestamp'] = timeStamp.values  final\_data = data.drop(['Date', 'Time'], axis=1)  final\_data = final\_data[final\_data.Timestamp != 'ValueError']  final\_data.head()  **Output:**  Earthquake Prediction Using Machine Learning **Visualization** Here, we will visualize the earthquakes that have occurred all around the world.   1. from mpl\_toolkits.basemap **import** Basemap 3. m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=180,lat\_ts=20,resolution='c') 5. longitudes = data["Longitude"].tolist() 6. latitudes = data["Latitude"].tolist() 7. #m = Basemap(width=12000000,height=9000000,projection='lcc', 8. #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')  .drawmapboundary()  m.drawcountries()  plt.show()  **Output:**  Earthquake Prediction Using Machine Learning  We have located on the world map where earthquakes happened in the last few years. **Splitting The Dataset** Now we will split the dataset into a training and testing set.  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)  **Output:**  Earthquake Prediction Using Machine Learning  We will be using the RandomForestRegressor model to predict the earthquake, here will look for its accuracy.  reg = RandomForestRegressor(random\_state=42)  reg.fit(X\_train, y\_train)  reg.predict(X\_test)  **Output:**  Earthquake Prediction Using Machine Learning reg.score(X\_test, y\_test)  reg.score(X\_test, y\_test)  **Output:**  Earthquake Prediction Using Machine Learning  86% of accuracy is quite high.  Now we will shift to GridSearch.  from sklearn.model\_selection **import** GridSearchCV  parameters = {'n\_estimators':[10, 20, 50, 100, 200, 500]}  grid\_obj = GridSearchCV(reg, parameters)  grid\_fit = grid\_obj.fit(X\_train, y\_train)  best\_fit = grid\_fit.best\_estimator\_  best\_fit.predict(X\_test)  **Output:**  Earthquake Prediction Using Machine Learning   1. best\_fit.score(X\_test, y\_test)   **Output:**  Earthquake Prediction Using Machine Learning  Considering it's a natural phenomenon, we have got a high accuracy number.  We will employ a neural network for predicting the earthquake. **Neural Network Model** 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.  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  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=activation, optimizer=optimizer, loss=loss)  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))  **Output:**  Earthquake Prediction Using Machine Learning  model = Sequential()  model.add(Dense(16, activation='relu', input\_shape=(3,)))  model.add(Dense(16, activation='relu'))  model.add(Dense(2, activation='softmax'))  model.compile(optimizer='SGD', loss='squared\_hinge', metrics=['accuracy'])  model.fit(X\_train, y\_train, batch\_size=10, epochs=20, verbose=1, validation\_data=(X\_test, y\_test))  **Output:**  Earthquake Prediction Using Machine Learning  [test\_loss, test\_acc] = model.evaluate(X\_test, y\_test)  print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test\_loss, test\_acc))  **Output:**  Earthquake Prediction Using Machine Learning |