**Naan Mudhalvan – Artificial Intelligence -Group 1**

**Earthquake Prediction Model Using Python**

**TEAM MEMBER**

**211121104037 : NIVETHA.V**

**Phase 5 Final submission document**



**Introduction:**

This project aims to predict the magnitude and probability of Earthquake occurring in a particular region using the historic data .The realm of data -driven issue-solving, an organized methodology that includes phased development, design thinking, and problem identification must be strictly adhered to. This approach guarantees that we not only understand the issue at hand but also develop creative solutions that are supported by facts and put through a thorough testing process. This document explores the entire process of solving a particular problem in this setting, going over every step from conception to deployment.

**Design Thinking process:**

**1.Problem Definition:**

* The problem at hand is to develop an earthquake prediction model utilizing a dataset from Kaggle. The primary objective is to comprehensively understand the earthquake data, extract key features, visualize them on a global scale, split the data for training and testing purposes, and construct a neural network model for predicting earthquake magnitudes based on the provided features.

**2.Data** **Source**:

* Kaggle Dataset Selection
* Begin by selecting a Kaggle dataset that contains relevant earthquake data. The dataset should ideally include features such as date, time, latitude, longitude, depth, and magnitude. Evaluate the dataset for data quality, completeness, and consistency.

**3.Feature** **Exploration**:

* **Analyze Key Features**:

Examine the chosen dataset to understand the distribution, correlations, and characteristics of the essential features (date, time, latitude, longitude, depth, and magnitude). Address missing or inconsistent data if necessary.

**4.Visualization**:

* **Global** **Overview**:

Create a world map visualization that depicts the distribution of earthquake occurrences based on the provided latitude and longitude information. Utilize color coding or markers to represent earthquake magnitude.

**5.Data** **Splitting**:

**Training and Testing Sets:**

* Split the dataset into two subsets: a training set and a test set. The training set will be used to train the neural network model, while the test set will serve for model evaluation and validation. Ensure that the data splitting process maintains the temporal order of earthquake occurrences.

**6.Model** **Development**:

**Neural** **Network** **Architecture**:

* Develop a neural network model suitable for earthquake magnitude prediction.
* Design the architecture, including input layers (features), hidden layers, and output layer.
* Consider hyperparameter tuning and regularization techniques to optimize model performance.
* Implement a loss function suitable for regression tasks.

**7.Training** **and** **Evaluation**:

**Model** **Training**:

* Train the neural network model on the training set. Monitor training metrics and loss to assess convergence.

**Model** **Evaluation**:

* Evaluate the model’s performance on the test set using appropriate evaluation metrics (e.g., Mean Absolute Error, Root Mean Squared Error). Visualize and interpret the results to gauge the model’s accuracy and reliability. Consider techniques like cross-validation for a robust assessment.

**8.Iteration** **and** **Improvement**:

* **Analyze** **Results**:

Reflect on the model’s performance and identify areas for improvement.

* **Iterate**:

If necessary, iterate through the feature engineering, model architecture, and hyperparameter tuning stages to enhance prediction accuracy.

**9.Documentation** **and** **Reporting**:

* Create a comprehensive report or presentation summarizing the entire process, including dataset details, feature exploration, visualization, model development, training, and evaluation results. Share insights and findings from the project, along with recommendations

**Data Preprossing Steps**

**Source program:**

1. **Import Libraries:**

* Start by importing the necessary libraries.

import numpyas npimportpandasas pd

importmatplotlib.pyplotasplt

import osprint(os.listdir("../input"))

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

data.columns

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

.

import numpyas npimportpandasas pd

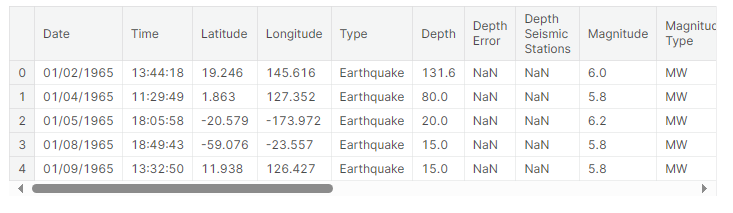
importmatplotlib.pyplotasplt

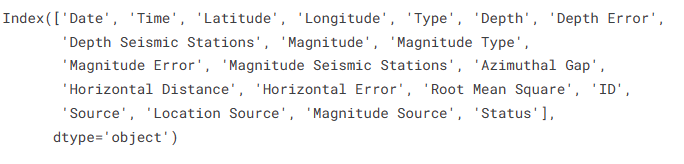
import osprint(os.listdir("../input"))

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

data.columns

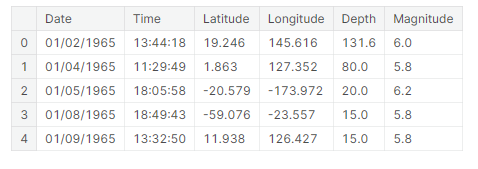
data=data[['Date','Time','Latitude','Longitude','Depth','Magnitude']]data.head()

data.columns



data=data[['Date','Time','Latitude','Longitude','Depth','Magnitude']]

data.head()



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()))timestamp.append('ValueError')

timeStamp= pd.Series(timestamp)

data['Timestamp']=timeStamp.values

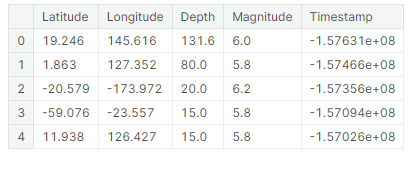
# 2. DataPreparation:

* UsingPandas,youreadseismicdatafromaCSVfile.
* Youhandledatetimeparsingissuesandextractspecifiedcolumns(suchas"Timestamp,""Latitude,""Longitude,""Magnitude,"and"Depth").
* Rowswith"ValueError"inthe"Timestamp"columnareremove

final\_data=data.drop(['Date','Time'],axis=1)

final\_data=final\_data[final\_data.Timestamp!='ValueError']

final\_data.head()



frommpl\_toolkits.basemapimportBasemap

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")

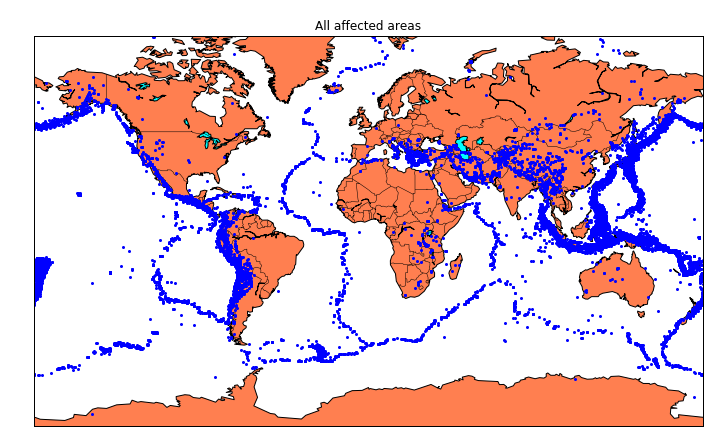
m.plot(x,y,"o",markersize=2,color='blue')

m.drawcoastlines()

m.fillcontinents(color='coral',lake\_color='aqua')

m.drawmapboundary()

m.drawcountries()

plt.show()

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()

timeStamp= pd.Series(timestamp)data['Timestamp']=timeStamp.value

final\_data= data.drop(['Date', 'Time'], axis=1)

final\_data= final\_data[final\_data.Timestamp!= 'ValueError']final\_data.head

# 3. DataVisualization:

* + Tovisualizeearthquakelocationsonamap,constructaBasemap.

frommpl\_toolkits.basemapimportBasemap

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()

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(

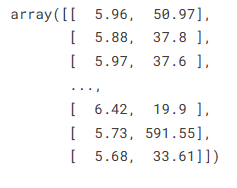
)

fromsklearn.ensembleimportRandomForestRegressor

reg=RandomForestRegressor(random\_state=42)

reg.fit(X\_train,y\_train)

reg.predict(X\_test)



# DataSplitting:

* Usingtrain\_test\_split,dividethedataintotrainingandtestingsets.
* YoutrainaRandomForestRegressorusingthetrainingsetofdata.
* Thescore()methodisusedtoassessthemodel.

fromkeras.modelsimportSequential

fromkeras.layersimportDense

defcreate\_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'])

returnmodel

fromkeras.wrappers.scikit\_learnimportKerasClassifier

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)

# Hyper parameterTuning(GridSearch):

* GridSearchCVisusedtoadjustthehyperparametersoftheRandomForestRegressor.

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\_))

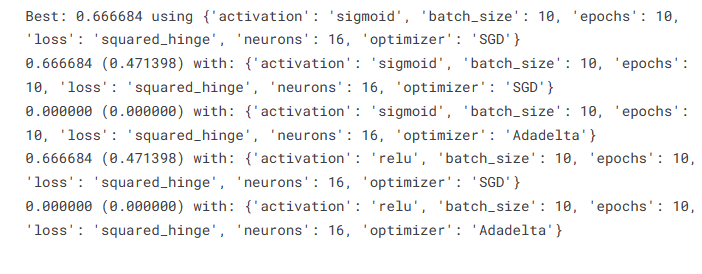
means=grid\_result.cv\_results\_['mean\_test\_score']

stds=grid\_result.cv\_results\_['std\_test\_score']

params=grid\_result.cv\_results\_['params']

formean,stdev,param**in**zip(means,stds,params):

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



model=Sequential()

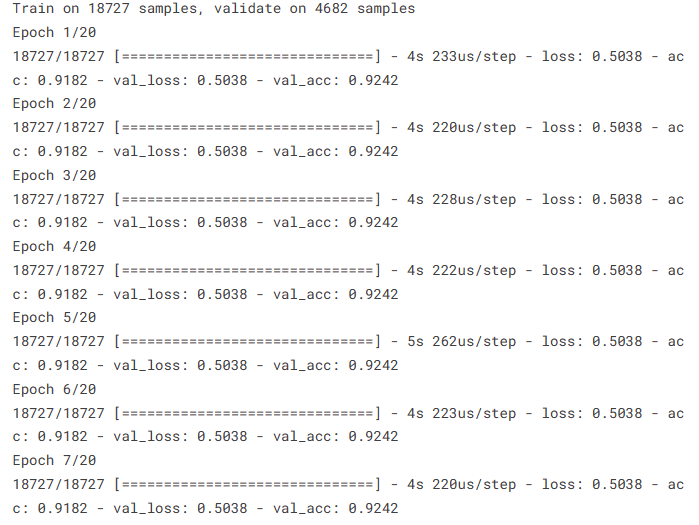
model.add(Dense(16,activation='relu',input\_shape=(3,)))

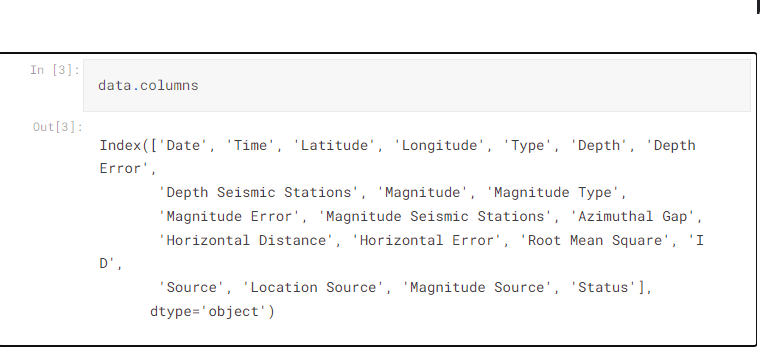
model.add(Dense(16,activation='relu'))

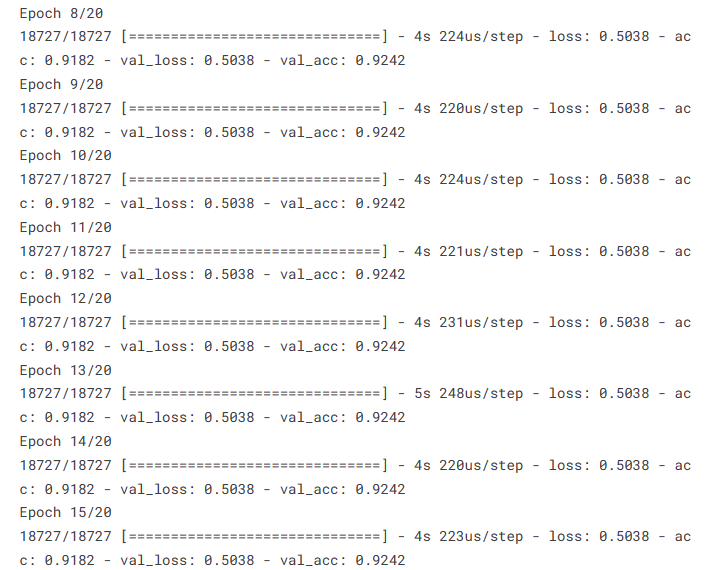
model.add(Dense(2,activation='softmax'))

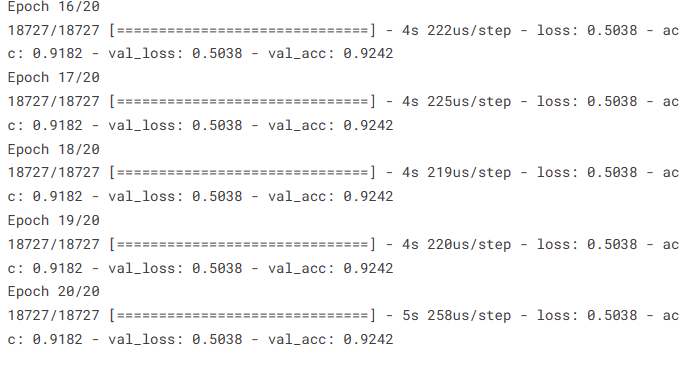
model.compile(optimizer='SGD',loss='squared\_hinge',metrics=['accuracy'])

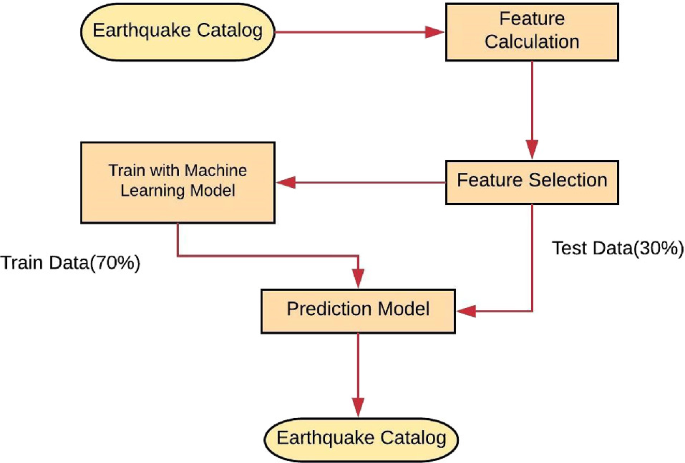
model.fit(X\_train,y\_train,batch\_size=10,epochs=20,verbose=1,validation\_data=(X\_test,y\_test))











# NeuralNetwork(Keras):

* YoubuildanearthquakepredictionKerasneuralnetworkmodel.
* Totunehyperparameters,youdefineaparametergrid.
* GridSearchCVisusedtoadjusttheneuralnetwork'shyperparameters.

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'])returnmodel

fromkeras.wrappers.scikit\_learnimportKerasClassifier

model=KerasClassifier(build\_fn=create\_model,verbose=0)neurons=[16]

batch\_size=[10]

epochs=[10]

activation=['sigmoid','relu']optimizer = ['SGD', 'Adadelta']loss=['squared\_hinge']

param\_grid=dict(neurons=neurons,batch\_size=batch\_size,epochs=epochs,activation=activation,optimizer=optimizer,loss=loss)

fromkeras.modelsimportSequentialfromkeras.layersimportDense

defcreate\_model(neurons,activation,optimizer,loss):model=Sequential()

# ModelEvaluation:

* Youusethetrainingdatatocreatethefinalneuralnetworkmodel.
* Usingthetestdata,youevaluatethemodel.
* Thetrainedneuralnetworkmodelissavedtoafilecalled"earthquake.h5"instep eight.
* ItappearsthatyouarenowaskingforchangestothecodeandasuitableCSVfile. Hereareafewideas:

# DataQuality:

* Ensurethatyourseismicdataareofhighquality.Theperformanceofthemodeldependsonit.

# FeatureEngineering:

Investigateanddevelopfreshcharacteristicsthatcouldraisepredictionprecision

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))

# HyperparameterTuning:

* To identify the ideal model configuration, experiment with varioushyperparametersforboththeneuralnetworkandtheRandomForestRegressor.

# DataScaling:

* Scalingornormalizingyourinputfeaturesissomethingtothinkabout,especiallyifyou'reutilizingneuralnetworks.

# Cross-Validation:

* Usecross-validationtoevaluatethestabilityofthemodelanddecreaseover fitting.

# ModelEvaluationMetrics:

* Toevaluatetheperformanceofamodel,useappropriateevaluationmetricsfor regression tasks, such as Mean Absolute Error (MAE) or Mean SquaredError(MSE).

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))

# DataSource:

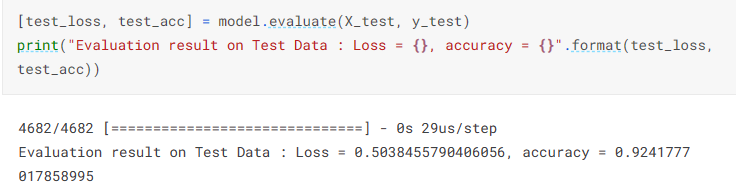
* Check that the earthquake dataset you have access to is reputable and current.On websites like the USGS Earthquake Hazards Program,you can find earthquake datasets.

[test\_loss,test\_acc]=model.evaluate(X\_test,y\_test)

print("EvaluationresultonTestData:Loss=**{}**,accuracy=**{}**".format(test\_loss,test\_acc))

# Filedirectories:

* Ensure that the file Directories you use to load data and save models are precise and easy to retrieve.



model.save('earthquake.h5')

**Conclusion:**

An earthquake prediction model using Python and artificial intelligence represents a crucial application that showcases the potential of AI and data science in addressing real-world problems. It offers the possibility of saving lives and reducing the impact of natural disasters by providing timely information and insights for decision-makers and the public.

This design thinking document provides a structured roadmap for developing an earthquake prediction model, ensuring that each crucial step is addressed systematically. It promotes a data-driven approach and continuous improvement for achieving accurate earthquake magnitude predictions.