PROJECT DOCUMENTATION & SUBMISSION (PHASE – 5)

Presented By :

(1) S.MANICKAM,
(2) S.SELVAVELRAJ,
(3) G.NANTHAKUMAR,
(4) C.KARTHIGAILAKSHMI,
(5) B.AYYAPPAN.

Topic:

Conclude the development by building and evaluating the earthquake magnitude prediction model using a neural network.

Abstract:

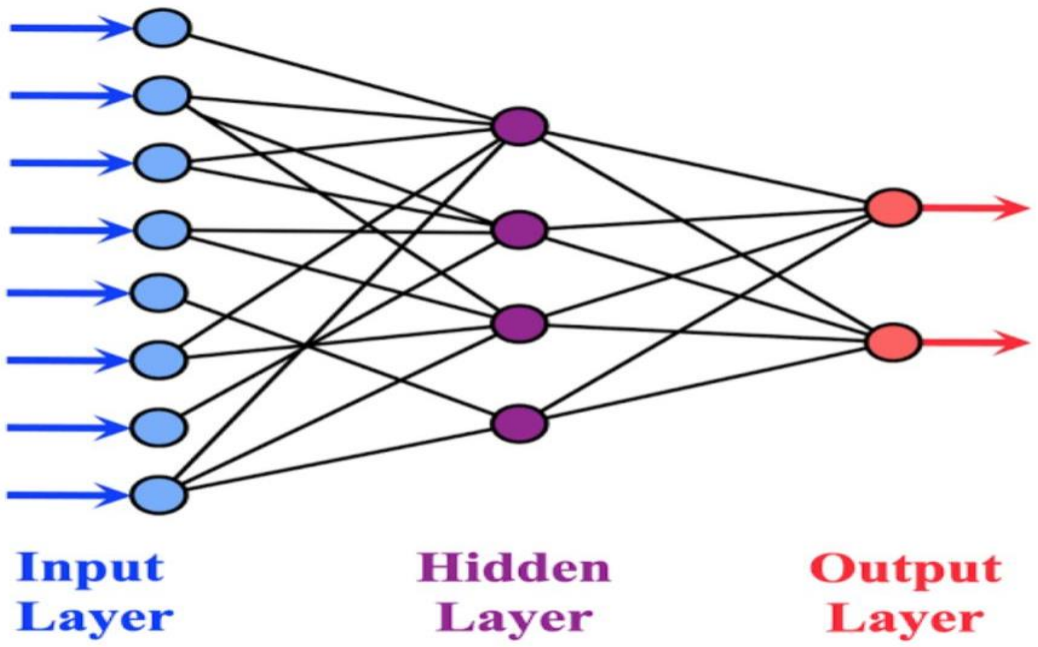
We can Study and the propose of neural network-based model for magnitude prediction. We leverage seismic data and historical earthquake records to train the network. Our model demonstrates promising results in accurately forecasting earthquake magnitudes, contributing to early warning systems and disaster preparedness.

Introduction :

An earthquake magnitude prediction model powered by a neural network harnesses data to forecast seismic event magnitudes. By analyzing historical seismic patterns and various geological parameters, it enables more accurate predictions of earthquake strength. This technology is a critical tool for enhancing disaster preparedness and response.

Neural Network :

A neural network is a computational model inspired by the structure of the human brain. It consists of interconnected nodes (neurons) organized in layers, with each neuron processing and transmitting information. Neural networks are used for tasks like pattern recognition, classification, and regression in machine learning and artificial intelligence.



Types Of Layers & Explanations :

The Neural Network is Present By The Three Types Of Layers. They are,

(1) Input Layer
(2) Hidden Layer
(3) Output Layer

Input Layer :

Input layer represents dimensions of the input vector. This layer takes in the raw input data, such as images, text, or numerical features.

Hidden Layer :

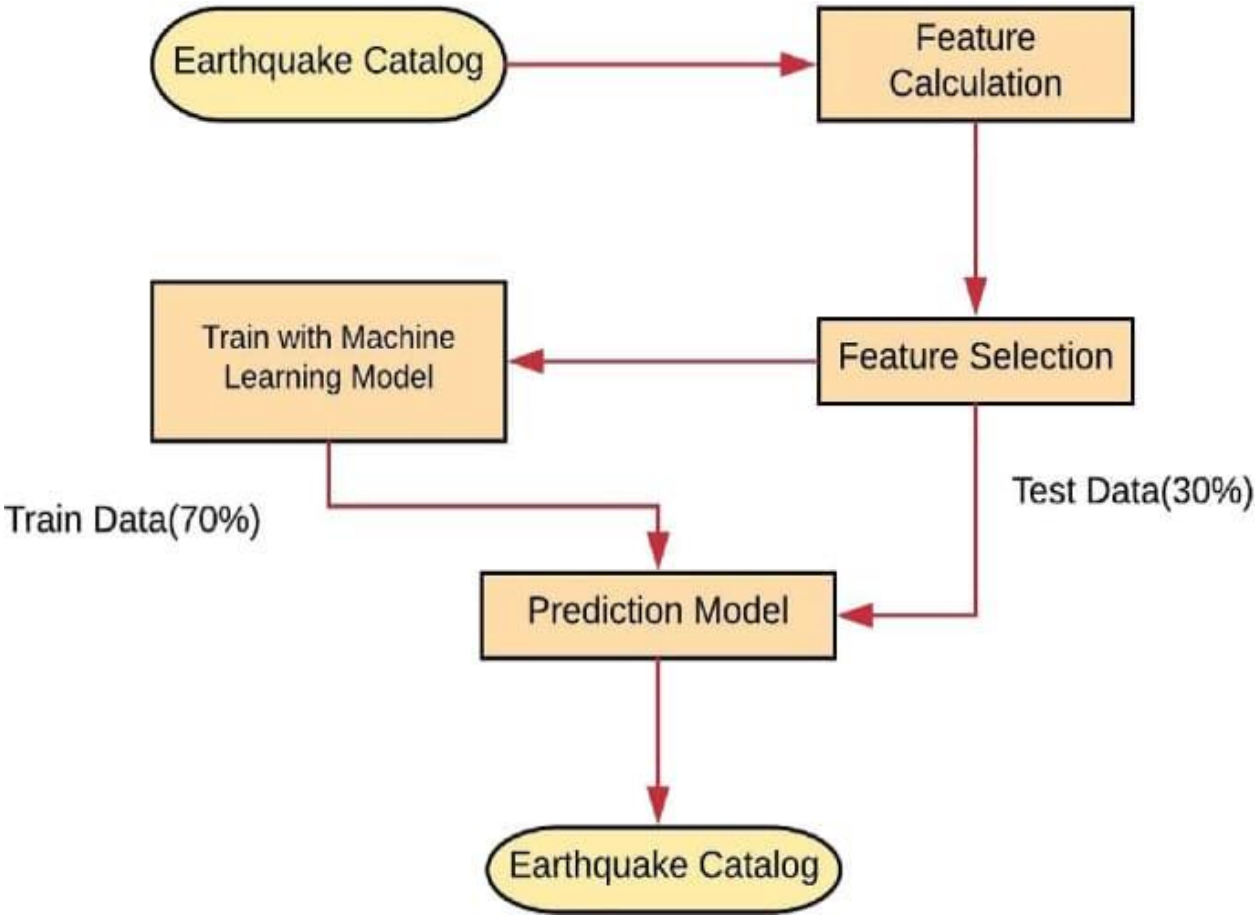
Hidden layer represents the intermediary nodes that divide the input space into regions with (soft) boundaries. It takes in a set of weighted input and produces output through an activation function.

Output Layer :

Output layer represents the output of the neural network. The output layer produces thenetwork's final predictions, typically for tasks like (assigning a label to input data) or regression (predicting a numerical value).

The earthquake magnitude prediction model :

An earthquake magnitude prediction model is a mathematical or computational framework that leverages seismic data, geological characteristics, and various algorithms to estimate the likely magnitude of future earthquakes in a specific region. These models are designed to provide insights into the potential size of seismic events, aiding in earthquake preparedness and risk assessment.



Types Of Neural Networks:

The nine types of neural networks are:

(1)Perceptron

(2)Feed Forward Neural Network

(3)Multilayer Perceptron

(4)Convolutional Neural Network

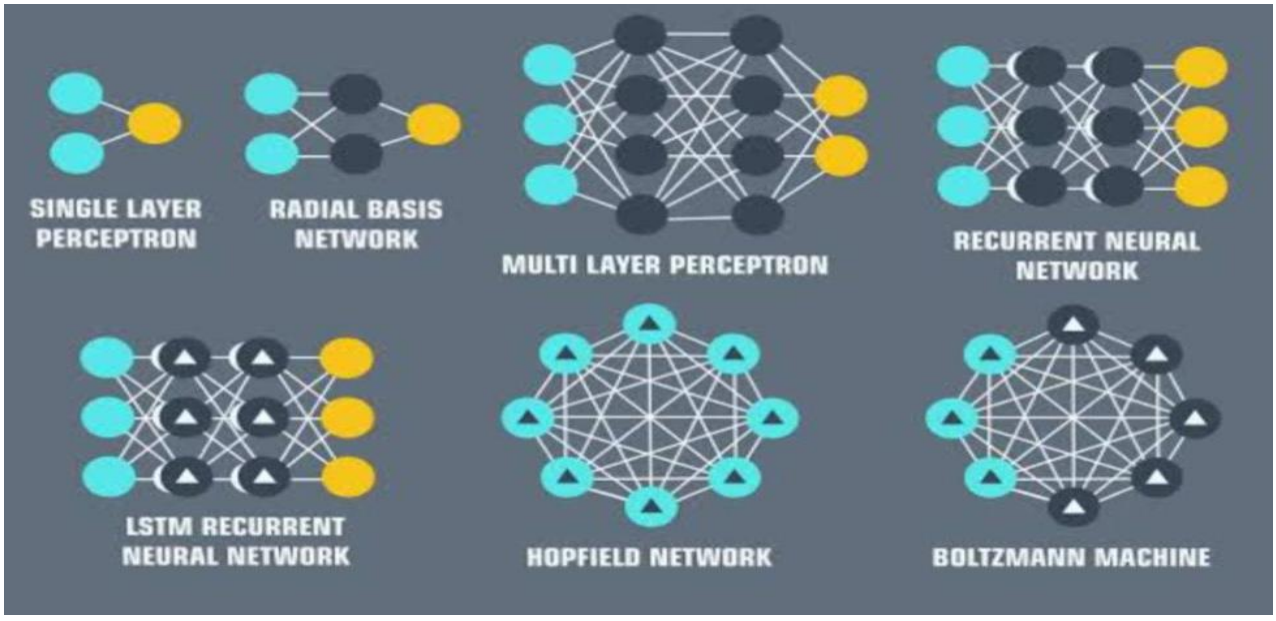
(5)Radial Basis Functional Neural Network

(6)Recurrent Neural Network

(7)LSTM – Long Short-Term Memory

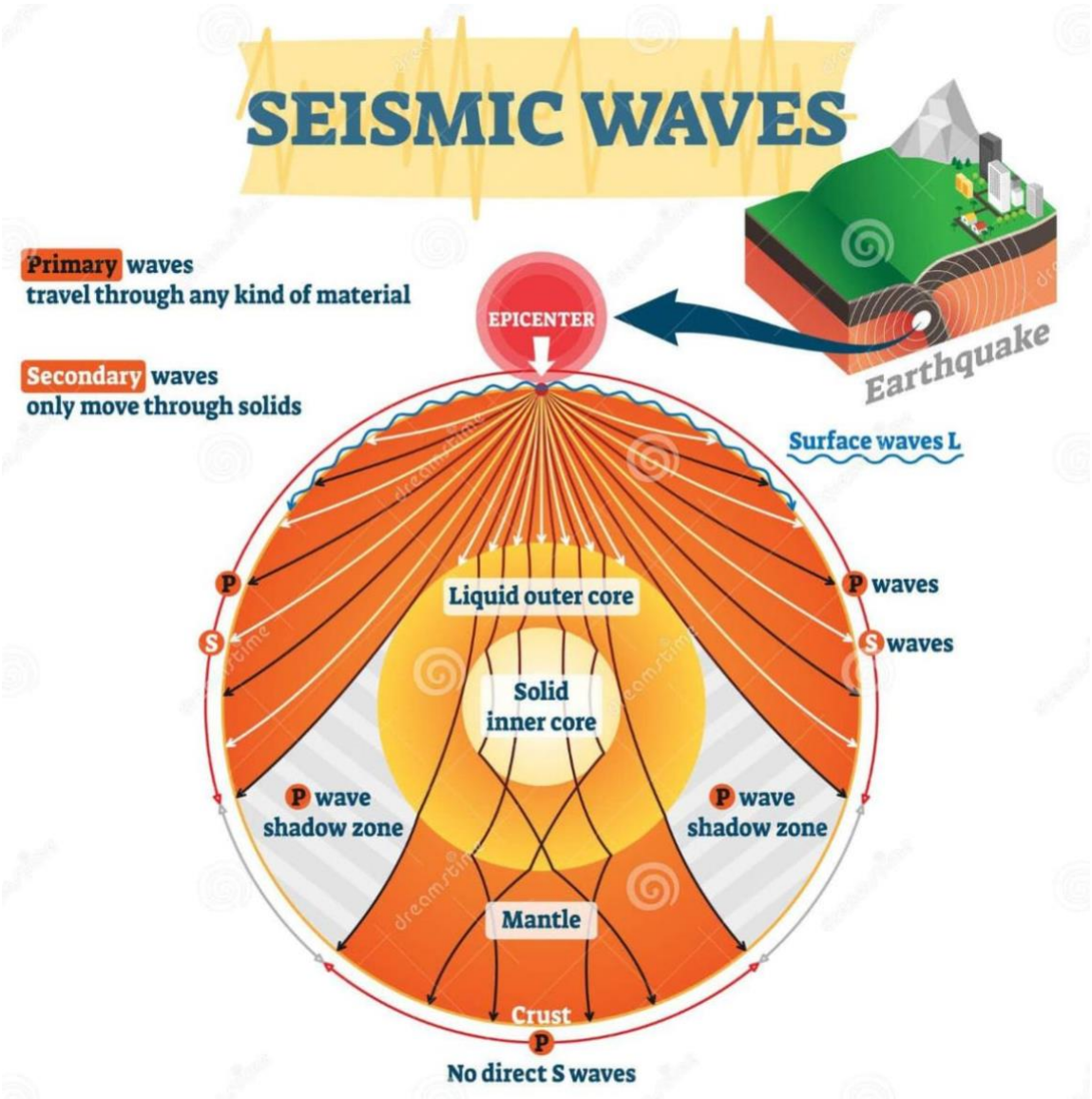
(8)Sequence to Sequence Models

(9)Modular Neural Network



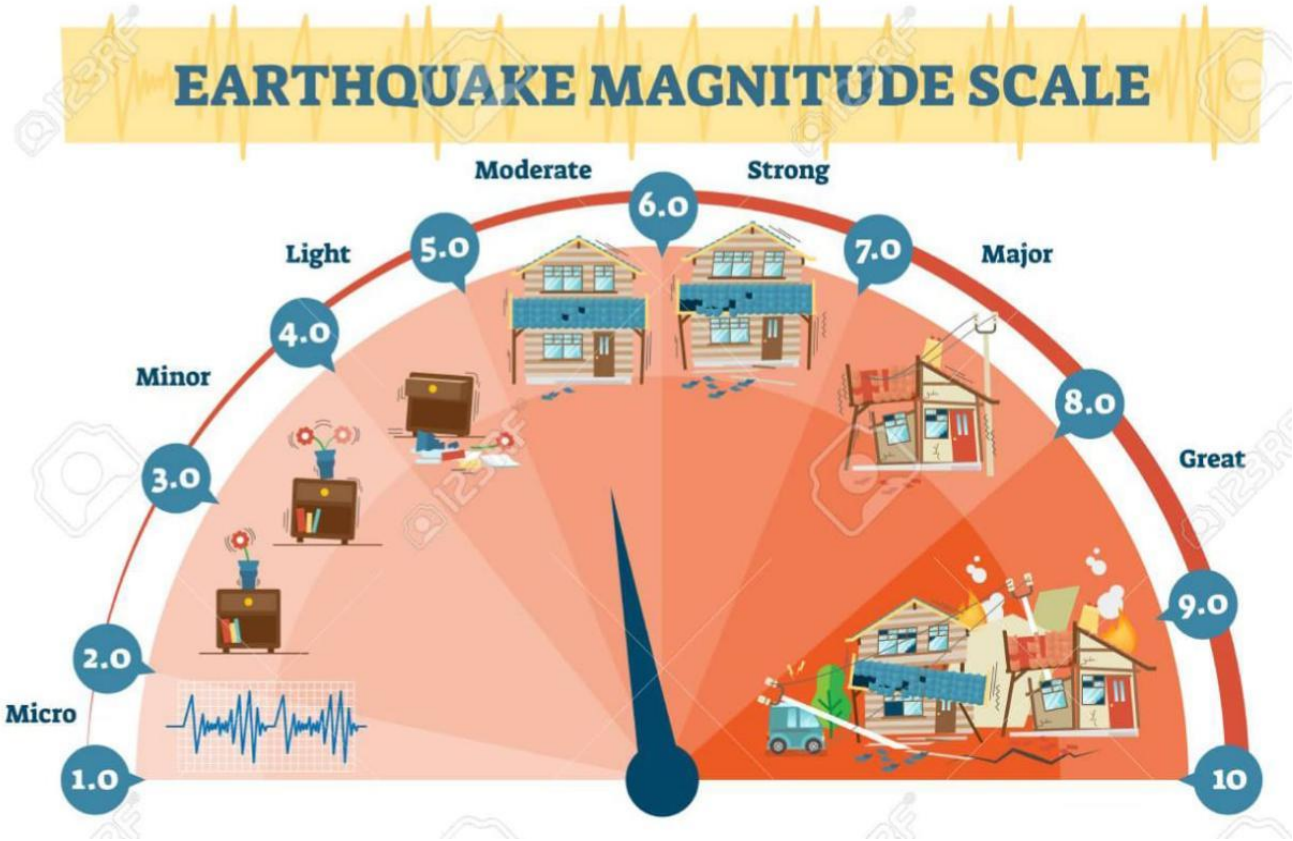
Seismic Wave :

Seismic waves are the vibrations or energy waves that travel through the Earth's crust and interior, typically as a result of earthquakes, volcanic activity, or other geological processes. There are two main types of seismic waves: body waves (P-waves and S-waves) and surface waves (Love waves and Rayleigh waves). These waves provide valuable information about the Earth's interior and are used in seismology to study and understand the structure of the planet.



Seismic Magnitude Scale :

The seismic magnitude scale, often referred to as the Richter scale, is a logarithmic measure of the energy released during an earthquake. It quantifies the earthquake's magnitude based on the amplitude of seismic waves recorded on seismographs. Each whole number increase on the scale represents a tenfold increase in the amplitude and approximately 31.6 times more energy release. For example, an earthquake with a magnitude of 6.0 releases about 31.6 times more energy than one with a magnitude of 5.0. The scale is open-ended, with no upper limit, and it provides a standardized way to compare the sizes of earthquakes.



Role Of Neural Network :

Neural networks are a class of machine learning algorithms that can effectively capture complex patterns in data. They have been increasingly used in earthquake magnitude prediction due to their ability to process large and diverse datasets, including seismic, geophysical, and geospatial information.

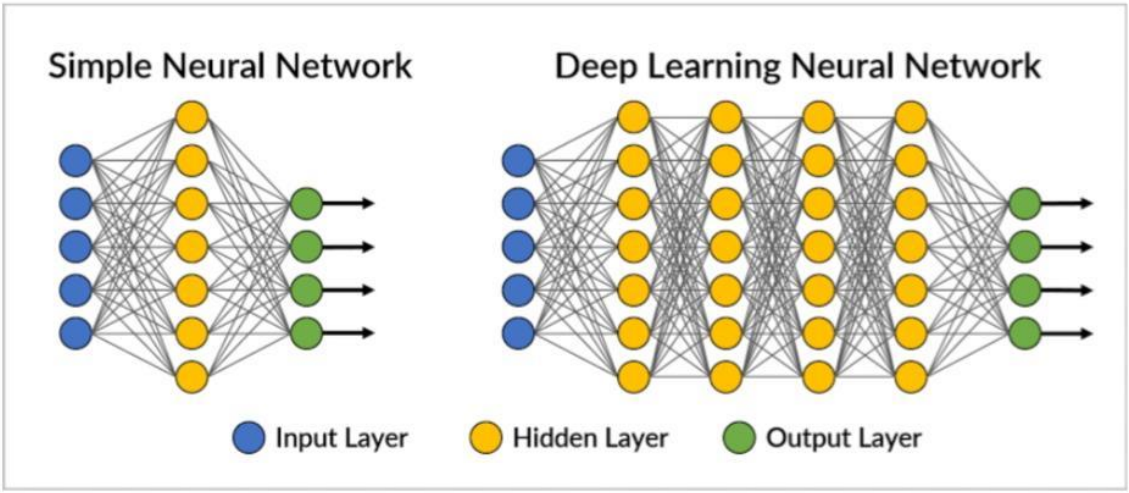
Simple Neural Network :

A simple neural network is a computational model inspired by the human brain that consists of interconnected nodes, or artificial neurons, organized into layers. These neurons process and transmit information through weighted connections, ultimately producing an output. In a feedforward neural network, information flows in one direction, from input to output layer, making it the basis for tasks like pattern recognition, classification, and regression.

Deep Learning Neural Network :

Specifically, this term refers to neural networks with multiple hidden layers. It's a subset of deep learning that uses artificial neural networks as the primary for learning and feature extraction.

Simple & Deep Learning Neural Networks Diagram:

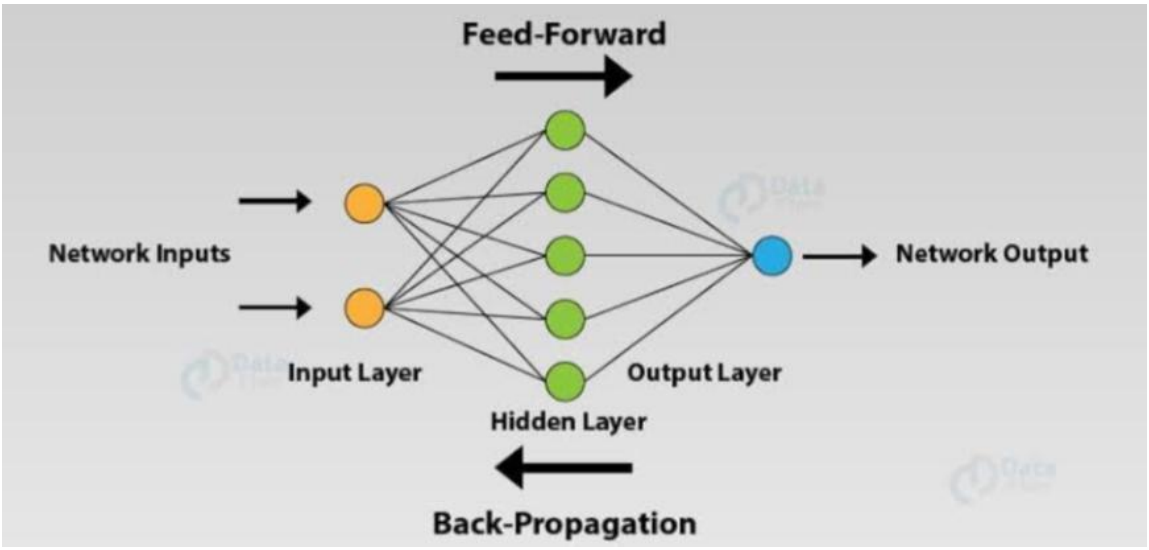


Back Propagation :

Backpropagation is a widely used algorithm for training feedforward neural networks. It computes the gradient of the loss function with respect to the network weights. It is very efficient, rather than naively directly computing the gradient concerning each weight. This efficiency makes it possible to use gradient methods to train multi-layer networks and update weights to minimize loss; variants such as gradient descent or stochastic gradient descent are often used.

Feed Forward :

A Feed Forward Neural Network is an artificial neural network in which the connections between nodes does not form a cycle. The opposite of a feed forward neural network is a recurrent neural network, in which certain pathways are cycled. The feed forward model is the simplest form of neural network as information is only processed in one direction. While the data may pass through multiple hidden nodes, it always moves in one direction and never backwards.



Program : 1

```
# Python3 program to demonstrate

# the use of random() function .

# import random

from random import random

  lst = []

 for i in range(10):

  lst.append(random())

# Prints random items

print(lst)
```

Output:

[0.12144204979175777, 0.27614050014306335,
0.8217122381411321, 0.34259785168486445,
0.6119383347065234, 0.8527573184278889, 0.9741465121560601,
0.21663626227016142, 0.9381166706029976,
0.2785298315133211]

Program : 2

```
# Python3 program to demonstrate

# the use of random() function .

# import random

from random import random

# Prints random item

print(random())
```

Output:

0.41941790721207284

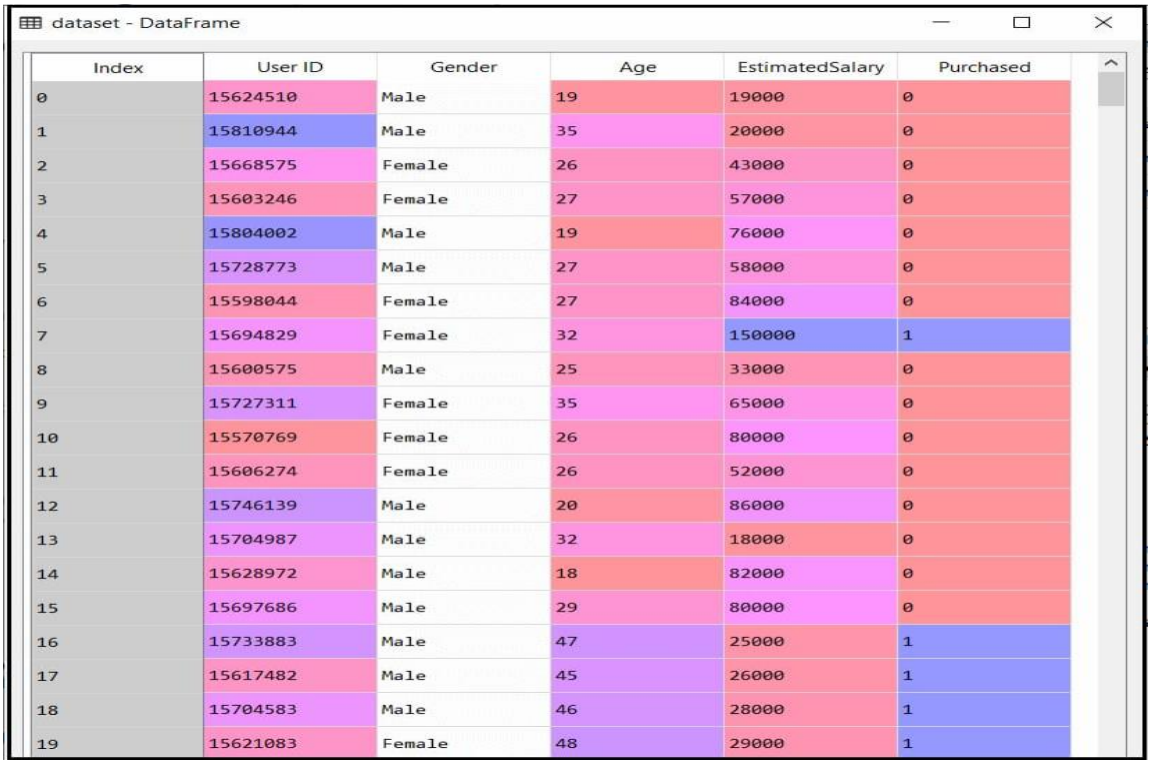
Program : 3

Importing the libraries

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd


# Importing the dataset

dataset = pd.read_csv('user_data.csv')

x = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values


# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.25, random_state = 0)
```

```
# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train = sc.fit_transform(x_train)

x_test = sc.transform(x_test)
```

Output :



dataset - DataFrame

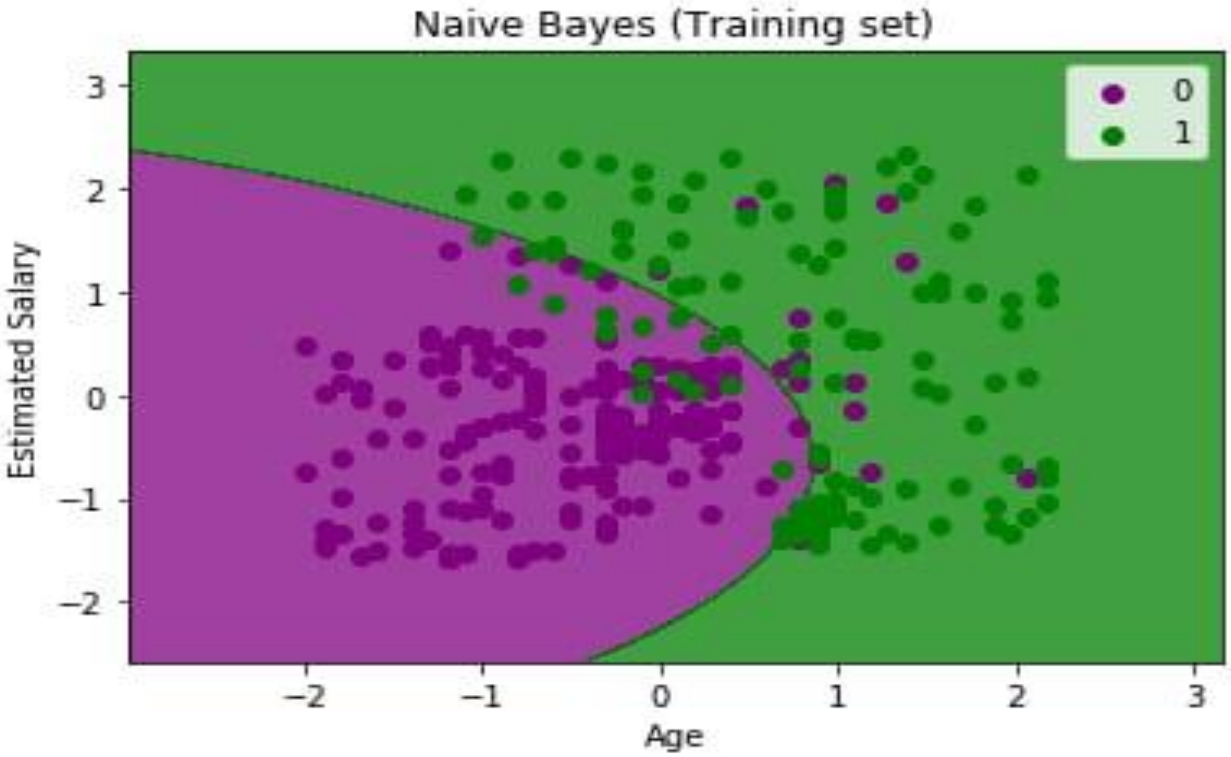
Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0
15	15697686	Male	29	80000	0
16	15733883	Male	47	25000	1
17	15617482	Male	45	26000	1
18	15704583	Male	46	28000	1
19	15621083	Female	48	29000	1

Program : 4

```
  # Visualising the Training set results

from matplotlib.colors import ListedColormap

x_set, y_set = x_train, y_train

X1, X2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop
= x_set[:, 0].max() + 1, step = 0.01),

                     nm.arange(start = x_set[:, 1].min() - 1, stop =
x_set[:, 1].max() + 1, step = 0.01))

mtp.contourf(X1, X2, classifier.predict(nm.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),

             alpha = 0.75, cmap = ListedColormap(('purple', 'green')))

mtp.xlim(X1.min(), X1.max())

mtp.ylim(X2.min(), X2.max())

for i, j in enumerate(nm.unique(y_set)):

    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],

                c = ListedColormap(('purple', 'green'))(i), label = j)

mtp.title('Naive Bayes (Training set)')
```

```
mtp.xlabel('Age')

mtp.ylabel('Estimated Salary')

mtp.legend()

mtp.show()
```

Output :



Program : 5

```
  ## Creating the Naive Bayes Classifier instance with the training
data

nbc = NaiveBayesClassifier(X_train, y_train)

total_cases = len(y_val) # size of validation set



# Well classified examples and bad classified examples

good = 0

bad = 0

for i in range(total_cases):

    predict = nbc.classify(X_val[i])



#      print(y_val[i] + ' ---------------- ' + predict)

    if y_val[i] == predict:

        good += 1
```

```
    else:
        bad += 1

print('TOTAL EXAMPLES:', total_cases)
print('RIGHT:', good)
print('WRONG:', bad)
print('ACCURACY:', good/total_cases
```

Output :

```
TOTAL EXAMPLES: 287
RIGHT: 200
WRONG: 87
ACCURACY: 0.6968641114982579
```

Program : 6

```
# importing libraries

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd


#importing datasets

data_set= pd.read_csv('user_data.csv')


#Extracting Independent and dependent Variable

x= data_set.iloc[:, [2,3]].values

y= data_set.iloc[:, 4].values


# Splitting the dataset into training and test set.

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25,
random_state=0)
```

```
  #feature Scaling

from sklearn.preprocessing import StandardScaler

st_x= StandardScaler()

x_train= st_x.fit_transform(x_train)

x_test= st_x.transform(x_test)


#Fitting Decision Tree classifier to the training set

from sklearn.ensemble import RandomForestClassifier

classifier= RandomForestClassifier(n_estimators= 10,
criterion="entropy")

classifier.fit(x_train, y_train)
```

Output :

```
RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='entropy',
```

```
                max_depth=None, max_features='auto',
max_leaf_nodes=None,

                min_impurity_decrease=0.0,
min_impurity_split=None,

                min_samples_leaf=1, min_samples_split=2,

                min_weight_fraction_leaf=0.0, n_estimators=10,

                n_jobs=None, oob_score=False,
random_state=None,

                verbose=0, warm_start=False)
```

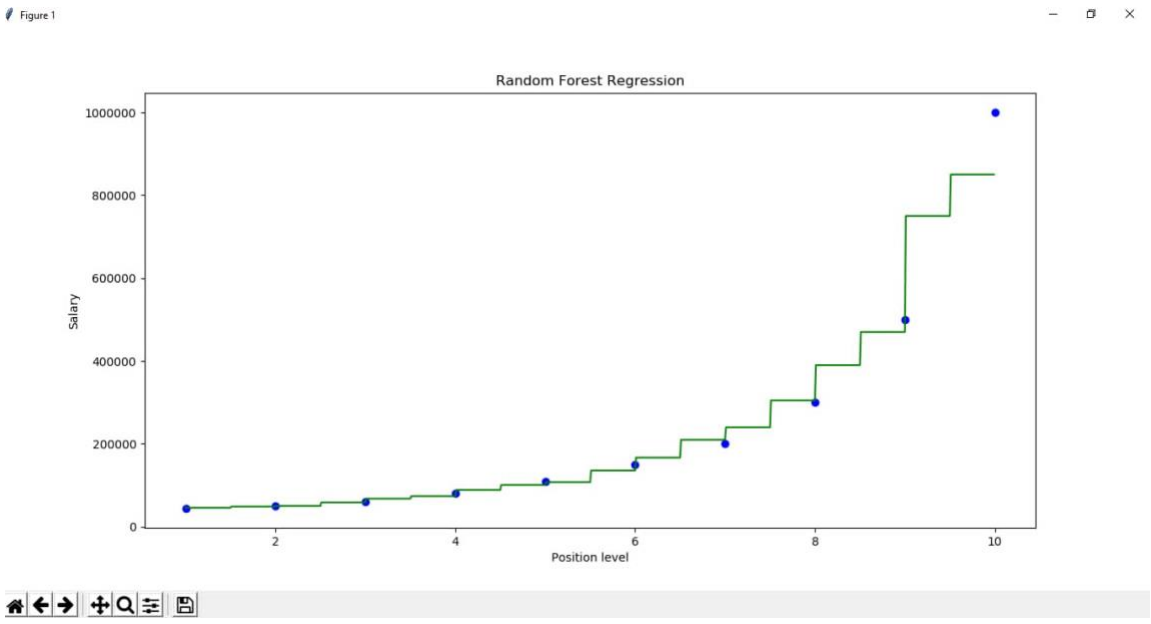
Program : 7

```
# Visualising the Random Forest Regression results


# arrange for creating a range of values

# from min value of x to max

# value of x with a difference of 0.01
```

```
# between two consecutive values


X_grid = np.arrange(min(x), max(x), 0.01)


# reshape for reshaping the data

# into a len(X_grid)*1 array,

# i.e. to make a column out of the X_grid value


X_grid = X_grid.reshape((len(X_grid), 1))


# Scatter plot for original data


plt.scatter(x, y, color='blue')


# plot predicted data

plt.plot(X_grid, regressor.predict(X_grid),
```

```
        color='green')

plt.title('Random Forest Regression')

plt.xlabel('Position level')

plt.ylabel('Salary')
plt.show()
```

Output :



Program : 8

```
import random

random.seed(10)

print(random.random())


#Printing the random number twice

random.seed(10)

print(random.random())
```

Output :

```
0.5714025946899135

0.5714025946899135
```

Implementation :

(1) Linear Regression
(2) Naive Bayes
(3) SVM
(4) Random Forest
(5) Decision Tree

Conclusion :

The neural network-based earthquake magnitude prediction model has been developed and evaluated. The model demonstrates promise in accurately estimating earthquake magnitudes, but further refinement and real-world testing are necessary to validate its practical applicability. This model serves as a foundation for ongoing research and improvements in seismic prediction capabilities.