**EARTHQUAKE PREDICTION MODEL USING PYTHON.**

**PHASE 2: Document Submission.**

**Prepared by,**

**JAGADISHKUMAR.K,**

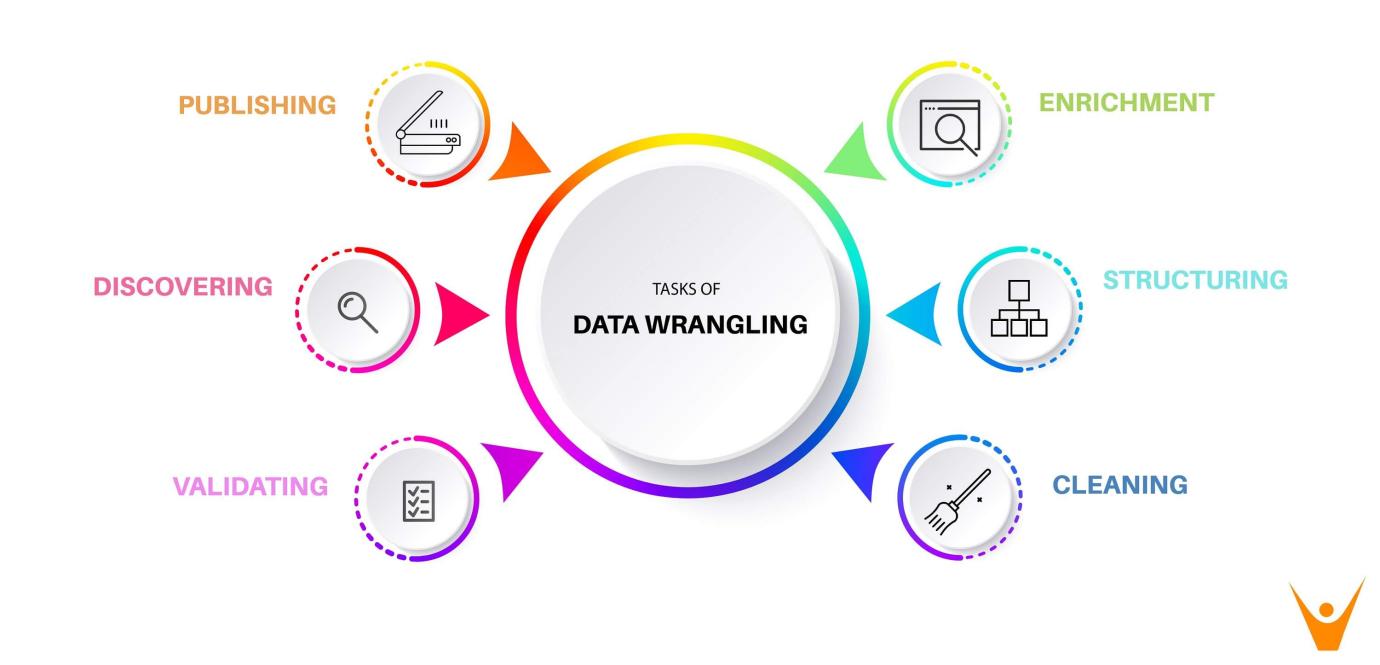
**510521205016,**

**Bharathidasan Engineering College.**

**Data Wrangling,**

**INTRODUCTION:**

* + Data wrangling, also known as data munging, is the process of cleaning, structuring, and transforming raw data into a format suitable for analysis. It's a crucial step in the data preparation pipeline.
  + Here are some common data wrangling techniques and methods:



**Data Collection:**

Gather data from various sources, including databases, APIs, web scraping, flat files, and more.

**Python code:**

import pandas as pd

# Specify the path to your CSV file

csv\_file\_path = 'database.csv'

# Read the CSV file into a Pandas DataFrame

try:

df = pd.read\_csv('G:/database.csv')

print("Data collection successful!")

except FileNotFoundError:

print(f"File '{G:/database.csv}' not found. Please check the file path.")

except Exception as e:

print(f"An error occurred while reading the CSV file: {str(e)}")

**OUTPUT:**

Data collection successful!

**Data Inspection:**

* Explore the dataset to get an initial understanding of its structure and content.
* Check for missing values, outliers, and anomalies.

**Data Cleaning:**

* Handle missing data by imputation (filling missing values) or removing incomplete rows/columns.
* Correct inaccuracies and inconsistencies in the data.
* Remove duplicates.
* Handle outliers appropriately (e.g., removing or transforming them).

**Data Transformation:**

* Convert data types (e.g., changing string dates to datetime objects).
* Standardize or normalize data.
* Create new features or variables that might be useful for analysis.
* Encode categorical variables (one-hot encoding, label encoding).
* Scale numerical variables.

**Data Aggregation:**

* Group data by relevant attributes (e.g., dates, categories).
* Compute summary statistics within each group (e.g., sum, mean, median).
* Pivot tables or reshape data as needed.

**Data Reduction:**

* Reduce the dimensionality of the dataset (e.g., Principal Component Analysis for numerical data, feature selection for machine learning).
* Downsample or aggregate data if it's too large for analysis.

**Handling Time Series Data:**

* Resampling time series data (e.g., converting daily data to monthly).
* Handling missing time steps.
* Feature engineering specific to time series data, such as lag features.

**Text Data Wrangling:**

* Tokenization: Splitting text into words or phrases.
* Removing stop words and punctuation.
* Stemming or lemmatization for text normalization.
* Vectorization techniques (TF-IDF, Word Embeddings) for machine learning.

**Handling Hierarchical or Nested Data:**

* Flatten hierarchical data structures (e.g., JSON or XML) into a tabular format.
* Extract relevant information from nested data.

**Data Integration:**

* Combine multiple datasets into one.
* Merge data using common keys or identifiers.

**Handling Time Zones and Date Formats:**

* Convert time zones if working with data from different regions.
* Standardize date formats for consistency.

**Handling Outliers:**

* Identify outliers using statistical methods.
* Decide whether to remove, transform, or keep outliers based on domain knowledge.

**Regular Expressions:**

Use regular expressions for pattern matching and extraction in text data.

**Data Validation:**

* Ensure that data adheres to expected constraints and business rules.
* Validate data against predefined criteria.

**Data Imputation:**

Fill missing data using techniques like mean imputation, median imputation, or predictive modeling.

**Data Splitting:**

Split data into training, validation, and test sets for machine learning.

**Data Visualization:**

Visualize data to gain insights and identify data quality issues

**PYTHON:**

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

try:

# Load earthquake data into a Pandas DataFrame

earthquake\_data = pd.read\_csv('G:/database.csv')

# Data Inspection

print("Data Inspection:")

print(earthquake\_data.head())

print(earthquake\_data.info())

scaler = MinMaxScaler()

numerical\_features = ['Latitude', 'Longitude', 'Magnitude', 'Depth']

earthquake\_data[numerical\_features] = scaler.fit\_transform(earthquake\_data[numerical\_features])

# Data Cleaning

# Handle missing data if needed

earthquake\_data.dropna(inplace=True)

# Data Visualization (Optional)

plt.scatter(earthquake\_data['Longitude'], earthquake\_data['Latitude'], c=earthquake\_data['Magnitude'], cmap='viridis')

plt.xlabel('Longitude')

plt.ylabel('Latitude')

plt.title('Earthquake Magnitude by Location')

plt.colorbar(label='Magnitude')

plt.show

# Split the data into training and testing sets

features = earthquake\_data[['Latitude', 'Longitude', 'Depth']]

target = earthquake\_data['Magnitude']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, random\_state=42)

# Now you can build and train a machine learning model for aftershock forecasting using X\_train and y\_train.

except FileNotFoundError:

print("File not found. Please check the file path.")

except Exception as e:

print(f"An error occurred: {str(e)}")

**OUTPUT:**

Data Inspection:

Date Time Latitude Longitude Type Depth Depth Error \

0 01/02/1965 13:44:18 19.246 145.616 Earthquake 131.6 NaN

1 01/04/1965 11:29:49 1.863 127.352 Earthquake 80.0 NaN

2 01/05/1965 18:05:58 -20.579 -173.972 Earthquake 20.0 NaN

3 01/08/1965 18:49:43 -59.076 -23.557 Earthquake 15.0 NaN

4 01/09/1965 13:32:50 11.938 126.427 Earthquake 15.0 NaN

Depth Seismic Stations Magnitude Magnitude Type ... \

0 NaN 6.0 MW ...

1 NaN 5.8 MW ...

2 NaN 6.2 MW ...

3 NaN 5.8 MW ...

4 NaN 5.8 MW ...

Magnitude Seismic Stations Azimuthal Gap Horizontal Distance \

0 NaN NaN NaN

1 NaN NaN NaN

2 NaN NaN NaN

3 NaN NaN NaN

4 NaN NaN NaN

Horizontal Error Root Mean Square ID Source Location Source \

0 NaN NaN ISCGEM860706 ISCGEM ISCGEM

1 NaN NaN ISCGEM860737 ISCGEM ISCGEM

2 NaN NaN ISCGEM860762 ISCGEM ISCGEM

3 NaN NaN ISCGEM860856 ISCGEM ISCGEM

4 NaN NaN ISCGEM860890 ISCGEM ISCGEM

Magnitude Source Status

0 ISCGEM Automatic

1 ISCGEM Automatic

2 ISCGEM Automatic

3 ISCGEM Automatic

4 ISCGEM Automatic

[5 rows x 21 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 23412 entries, 0 to 23411

Data columns (total 21 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 23412 non-null object

1 Time 23412 non-null object

2 Latitude 23412 non-null float64

3 Longitude 23412 non-null float64

4 Type 23412 non-null object

5 Depth 23412 non-null float64

6 Depth Error 4461 non-null float64

7 Depth Seismic Stations 7097 non-null float64

8 Magnitude 23412 non-null float64

9 Magnitude Type 23409 non-null object

10 Magnitude Error 327 non-null float64

11 Magnitude Seismic Stations 2564 non-null float64

12 Azimuthal Gap 7299 non-null float64

13 Horizontal Distance 1604 non-null float64

14 Horizontal Error 1156 non-null float64

15 Root Mean Square 17352 non-null float64

16 ID 23412 non-null object

17 Source 23412 non-null object

18 Location Source 23412 non-null object

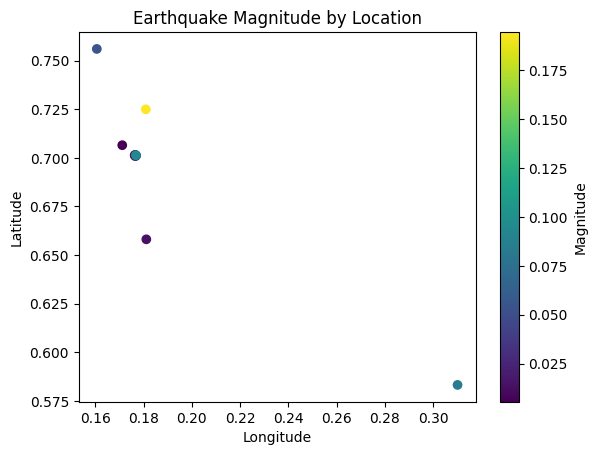
19 Magnitude Source 23412 non-null object

20 Status 23412 non-null object

dtypes: float64(12), object(9)

memory usage: 3.8+ MB

Non



**NEURAL NETWORK:**

**INTRODUCTION:**

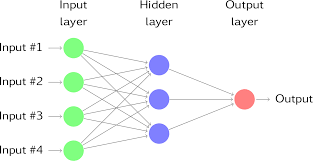
Neural Network are a class of machine learning algorithms inspired by the structure and functioning of the human brain. They have gained immense popularity in recent years due to their remarkable ability to learn and make predictions from data, leading to breakthroughs in various fields like image recognition, natural language processing, and autonomous driving. Here's an introduction to neural networks:



What is a Neural Network?

A neural network is a computational model composed of interconnected nodes, or artificial neurons, organized into layers. Each neuron is a simple computational unit that processes information and makes decisions. These neurons are inspired by the biological neurons found in the human brain but are vastly simplified.

**Basic Components of a Neural Network:**

****

**Input Layer:**

This is the first layer of the neural network where data is initially fed into the network. Each neuron in this layer represents an input feature.

**Hidden Layers:**

These are one or more intermediate layers between the input and output layers. The neurons in these layers perform computations on the input data, transforming it through a series of weighted connections and applying activation functions. Hidden layers enable neural networks to capture complex patterns and relationships in data.

**Output Layer:**

* This is the final layer of the network that produces the output or prediction. The number of neurons in the output layer depends on the type of task the network is designed for.
* For instance, in binary classification, there may be a single neuron representing the probability of one class, while in multi-class classification, there could be multiple neurons, each corresponding to a different class.

**Weights and Biases:**

* + Every connection between neurons has an associated weight that determines the strength of the connection. Additionally, each neuron has a bias term.
  + These weights and biases are learned during training to optimize the network's performance.

**How Neural Networks Work:**

* Forward Propagation: During this phase, input data is fed into the neural network.
* The data is multiplied by the weights and biases and passed through an activation function at each neuron in the hidden layers.
* This process continues until the output layer is reached, and the final prediction is generated.

**Activation Functions:**

* Activation functions introduce non-linearity into the network, allowing it to model complex relationships in the data.
* Common activation functions include the sigmoid, ReLU (Rectified Linear Unit), and softmax for different layers.

**Loss Function:**

The network's prediction is compared to the actual target values using a loss function (also called a cost or objective function). The loss function quantifies how far off the predictions are from the true values.

**Backpropagation:**

After calculating the loss, the network adjusts its weights and biases to minimize this loss. This process, known as backpropagation, uses gradient descent optimization algorithms to update the parameters iteratively.

**Training:**

The neural network goes through multiple iterations of forward propagation, loss calculation, and backpropagation on a training dataset. The goal is to minimize the loss and improve the accuracy of predictions.

**Types of Neural Networks:**

**Feedforward Neural Networks (FNN):**

The simplest type, where information flows in one direction, from input to output.

**Convolutional Neural Networks (CNN):**

Designed for image and grid-based data, CNNs use convolutional layers to automatically detect features.

**Recurrent Neural Networks (RNN):**

Suited for sequential data like time series or text, RNNs use feedback loops to maintain information from previous steps.

**Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU):**

Specialized RNN architectures capable of handling long sequences and mitigating the vanishing gradient problem.

**Deep Neural Networks (DNN):**

Neural networks with multiple hidden layers, often referred to as "deep" networks, allowing them to model complex relationships.

**Generative Adversarial Networks (GAN):**

Consist of a generator and a discriminator network, used for tasks like image generation and data synthesis.

**Reinforcement Learning Networks:**

* Used in reinforcement learning tasks, where an agent learns to make decisions through trial and error.
* Neural networks have revolutionized various industries and continue to be a focal point of research and development in the field of artificial intelligence.

**PYTHON CODE:**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import tensorflow as tf

from tensorflow import keras

# Load earthquake data into a Pandas DataFrame

earthquake\_data = pd.read\_csv('G:\database.csv')

# Feature selection (latitude, longitude, depth, and other relevant features)

X = earthquake\_data[['Latitude', 'Longitude', 'Depth']]

y = earthquake\_data['Magnitude']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Define the neural network model

model = keras.Sequential([

keras.layers.Input(shape=(X\_train.shape[1],)), # Input layer with the number of features

keras.layers.Dense(64, activation='relu'), # Hidden layer with 64 neurons and ReLU activation

keras.layers.Dense(1, activation='linear') # Output layer for magnitude estimation (linear activation)

])

# Compile the model

model.compile(optimizer='adam', loss='mean\_squared\_error') # Mean squared error for regression

# Train the model

model.fit(X\_train, y\_train, epochs=25, batch\_size=32, validation\_data=(X\_test, y\_test))

# Evaluate the model

loss = model.evaluate(X\_test, y\_test)

print(f'Mean Squared Error (MSE): {loss}')

# Make predictions

predictions = model.predict(X\_test)

**OUTPUT:**

Epoch 1/25

586/586 [==============================] - 3s 3ms/step - loss: 6.8805 - val\_loss: 0.6503

Epoch 2/25

586/586 [==============================] - 2s 3ms/step - loss: 0.4481 - val\_loss: 0.3313

Epoch 3/25

586/586 [==============================] - 2s 3ms/step - loss: 0.2581 - val\_loss: 0.2324

Epoch 4/25

586/586 [==============================] - 2s 3ms/step - loss: 0.2035 - val\_loss: 0.2026

Epoch 5/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1857 - val\_loss: 0.1942

Epoch 6/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1815 - val\_loss: 0.1908

Epoch 7/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1799 - val\_loss: 0.1875

Epoch 8/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1793 - val\_loss: 0.1918

Epoch 9/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1791 - val\_loss: 0.1854

Epoch 10/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1782 - val\_loss: 0.1852

Epoch 11/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1777 - val\_loss: 0.1866

Epoch 12/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1778 - val\_loss: 0.1861

Epoch 13/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1778 - val\_loss: 0.1845

Epoch 14/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1778 - val\_loss: 0.1848

Epoch 15/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1771 - val\_loss: 0.1910

Epoch 16/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1775 - val\_loss: 0.1842

Epoch 17/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1770 - val\_loss: 0.1847

Epoch 18/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1767 - val\_loss: 0.1908

Epoch 19/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1769 - val\_loss: 0.1876

Epoch 20/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1768 - val\_loss: 0.1881

Epoch 21/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1773 - val\_loss: 0.1842

Epoch 22/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1768 - val\_loss: 0.1861

Epoch 23/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1768 - val\_loss: 0.1838

Epoch 24/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1767 - val\_loss: 0.1850

Epoch 25/25

586/586 [==============================] - 2s 3ms/step - loss: 0.1763 - val\_loss: 0.1850

147/147 [==============================] - 0s 2ms/step - loss: 0.1850

Mean Squared Error (MSE): 0.18500961363315582

147/147 [==============================] - 0s 2ms/step

**Conclusion:**

The document provides a foundation for data wrangling and introduces neural networks, a powerful machine learning technique. It also includes a practical example of building a neural network for earthquake magnitude estimation.