|  |  |
| --- | --- |
| **Name** | **MAHESWARAN.S** |
| **REG .NO** | **420121104031** |
| **Department** | **Computer Science** |
| **Year** | **III** |
| **College Name** | **AKTMCET** |
| **Group** | **IBM-GROUP-5** |

Earthquake Prediction Model using Python

**PHASE-2**

**Innovation**

Creating an earthquake prediction model is a challenging yet crucial task. One innovative approach involves using machine learning algorithms and Python libraries like scikit-learn. You can analyze various seismic features, historical earthquake data, and geological information to train your model.

Consider implementing a deep learning model, perhaps a neural network, to recognize complex patterns in seismic data. Transfer learning, where you use pre-trained models and fine-tune them for earthquake prediction, could also be beneficial.

Real-time data collection is key. Integrating IoT devices and sensors in earthquake-prone areas, feeding the data to your Python-based model, and continuously updating the model will enhance its accuracy over time.

**PROGRAM CODE**

import pandas as pd

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

data = pd.read\_csv('earthquake\_data.csv')

features = data.drop('label', axis=1)

labels = data['label']

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

model = DecisionTreeClassifier()

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

predictions = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, predictions)

print(f'Accuracy: {accuracy}')

**Module 3** : Data Wrangling Techniques

Ah, data wrangling, the art of cleaning and organizing raw data into a more usable form! There are various techniques to master this craft. Here are a few:

* **Handling Missing Data**:
* **Imputation**: Fill in missing values with a sensible estimate.
* **Dropping**: Remove rows or columns with missing data.
* **Data Transformation:**
* **Normalization/Scaling**: Standardize numerical values to a common scale.
* **Encoding**: Convert categorical data into numerical form for analysis.
* **Binning**: Group numerical data into bins for better analysis.
* **Dealing with Duplicates**:
* **Identifying and Removing**: Detect and eliminate identical rows.
* **Text Data Cleaning:**
* **Tokenization**: Break down text into individual words or phrases.
* **Removing Stopwords**: Eliminate common words that don't carry much meaning.
* **Stemming/Lemmatization**: Reduce words to their base or root form.
* **Handling Outliers**:
* **Detection**: Identify and examine data points significantly different from the rest.
* **Treatment**: Decide whether to remove, transform, or keep outliers based on the context.
* **Data Aggregation**:
* **Grouping**: Group data by certain criteria for analysis.
* **Summarization:** Create summary statistics for better understanding.
* **Joining and Merging:**
* **Combining Datasets:** Merge datasets based on common columns.
* **Time Series Data Techniques:**
* **Resampling:** Adjust the time frequency of the data.
* **Lagging and Leading**: Create new time-shifted features.
* **Regex (Regular Expressions):**
* **Pattern Matching:** Use regex to identify and manipulate patterns in text data.
* **Data Validation:**
* **Checking for Consistency:** Ensure data adheres to predefined rules.
* **Data Formatting:**
* **Date Formatting:** Ensure consistency in date formats.
* **Numeric Formatting:** Make sure numerical data is in the right format.

**Module 4:** Introduction to Neural Networks

Neural networks are like the cool kids of machine learning. Imagine them as a bunch of interconnected nodes, or neurons, inspired by the human brain.

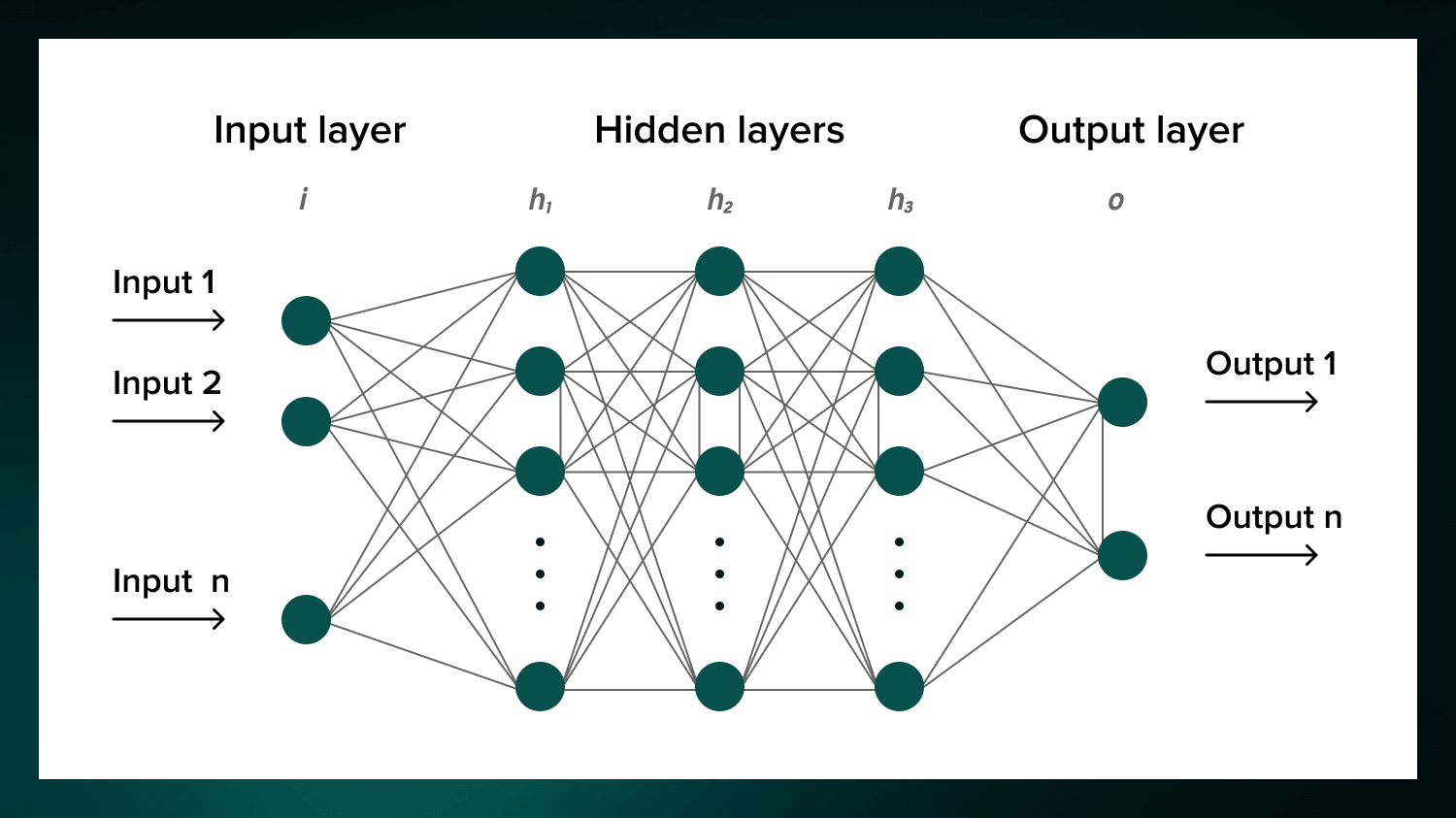
Each connection has a weight, and these weights are adjusted during training to make the network learn.

There are input layers where data goes in, output layers where results come out, and hidden layers in between doing the heavy lifting.

The magic happens when the network learns to map inputs to outputs by adjusting those weights. It's like a brain, but for computers.

Deep learning takes it up a notch with deep neural networks—more layers, more complexity.

They've been rocking the tech scene, doing everything from image recognition to natural language processing.



**Weighty matters**

Neural nets are a means of doing machine learning, in which a computer learns to perform some task by analyzing training examples. Usually, the examples have been hand-labeled in advance. An object recognition system, for instance, might be fed thousands of labeled images of cars, houses, coffee cups, and so on, and it would find visual patterns in the images that consistently correlate with particular labels.

**Minds and machines**

The neural nets described by McCullough and Pitts in 1944 had thresholds and weights, but they weren’t arranged into layers, and the researchers didn’t specify any training mechanism. What McCullough and Pitts showed was that a neural net could, in principle, compute any function that a digital computer could. The result was more neuroscience than computer science: The point was to suggest that the human brain could be thought of as a computing device.