EARTHQUAKE PREDICTION MODEL USING PYTHON

PROJECT DONE BY

Team Leader:

PRIYA GUPTA.K (922321106301)

Team members:

ABITHA.J (922321106003)

NITHYA.A (922321106020)

SHRISHTITHA.S (922321106034)

SUBHIKSHA.A (922321106038)

SWETHA.A.B (922321106040)

PHASE 4: DEVELOPMENT PART – 2

TOPIC: MACHINE LEARING ALGORITHM

OBJECTIVE:

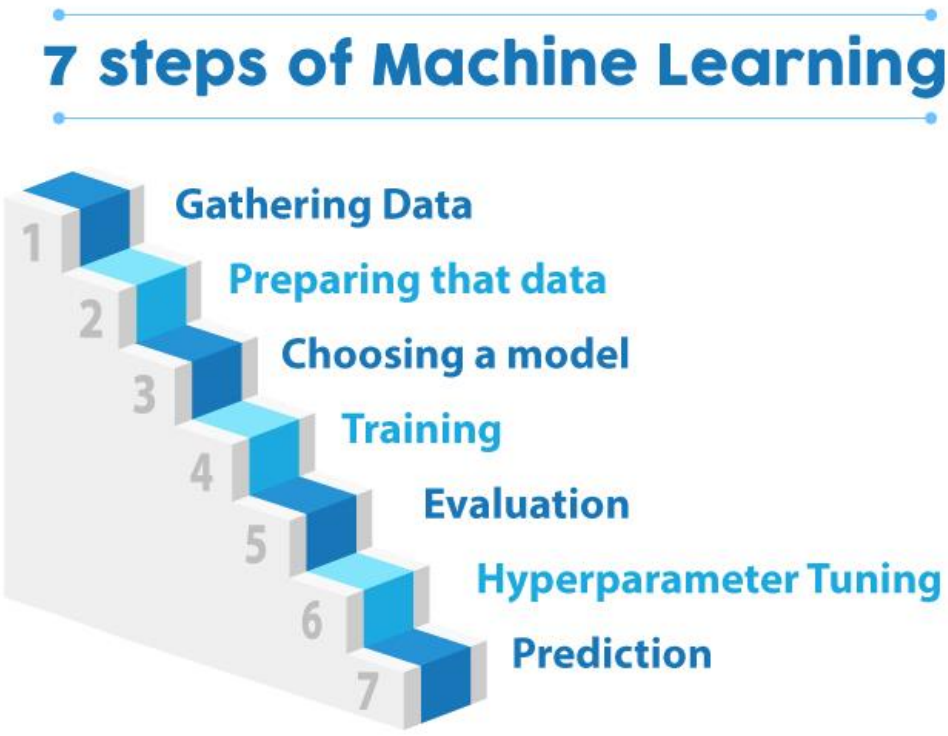
This Introduction will guide you through the initial steps of the process. In the Machine Learning Algorithm there are in some steps, they are classifies as Data collection, Data preparation, Choosing a model, Training, Evaluation, Parameter tuning and Predictions. Then, The Machine Learning Algorithm classified into four main types as supervised, Unsupervised, Semi-Supervised and Reinforcement. In these types, the supervised is divided into two types as Regression and Classification. The Unsupervised is classified into two types as Clustering and Association. The Semi-Supervised is defined as the combination of supervised and unsupervised. The Reinforcement is divided into two types ad Positive Reinforcement and Negative Reinforcement.

MACHINE LEARNING ALGORITHM:

Machine learning algorithms are mathematical model mapping methods used to learn or uncover underlying patterns embedded in the data. Machine learning comprises a group of computational algorithms that can perform pattern recognition, classification, and prediction on data by learning from existing data. The most common machine learning approaches in biology are support vector machines and artificial neural networks. ANN model with deep neuron layers can be used to predict sequence specificities of DNA- and RNA-binding proteins, noncoding variants, alternative splicing, and quantitative structure–activity relationship of drugs. Deep learning models have outperformed other machine learning methods in identifying more complex features from data. To achieve complex results, deep learning techniques require a higher volume of data and computational time, compared to other machine learning algorithms. Omic data such as genome, transcriptome, epigenome, proteome, and metabolome may be integrated into a

single model, which has large dimensions, and requires extensive time to build an appropriate model. Data collection can be minimized by reducing the dimension of input data, which can be done before or after data integration with principal component analysis (PCA), or after data integration with feature selection algorithms. Mode of action by network identification (MNI) combines reverse engineering network modeling with machine learning to decipher regulatory interactions. MNI uses a training set of multidimensional omic data to identify genetic components and network that correspond to a specific state. MNI, using a set of ordinary differential equations, directed graph relating the amounts of biomolecules to each other can be generated. For example, when transcriptomic data are used as training data, regulatory influences between genes can be inferred. In addition to MNI, another network-based system CellNet classifies cellular states based on the status of gene regulatory network. Both MNI and CellNet utilize machine learning integrated reverse engineering methods.

7 STEPS OF MACHINE LEARNING:



DATA COLLECTION:

Data collection means pooling data by scraping, capturing, and loading it from multiple sources, including offline and online sources. High volumes of data collection or data creation can be the hardest part of a machine learning project, especially at scale.

Collecting data for training the ML model is the basic step in the machine learning pipeline. The predictions made by ML systems can only be as good as the data on which they have been trained. Following are some of the problems that can arise in data collection:

- Inaccurate data. The collected data could be unrelated to the problem statement.
- Missing data. Sub-data could be missing. That could take the form of empty values in columns or missing images for some class of prediction.
- Data imbalance. Some classes or categories in the data may have a disproportionately high or low number of corresponding samples. As a result, they risk being under-represented in the model.
- Data bias. Depending on how the data, subjects and labels themselves are chosen, the model could propagate inherent biases on gender, politics, age or region, for example. Data bias is difficult to detect and remove.

Several techniques can be applied to address those problems:

- Pre-cleaned, freely available datasets. If the problem statement (for example, image classification, object recognition) aligns with a clean, pre-existing, properly formulated dataset, then take advantage of existing, open-source expertise.
- Web crawling and scraping. Automated tools, bots and headless browsers can crawl and scrape websites for data.
- Private data. ML engineers can create their own data. This is helpful when the amount of data required to train the model is small and the problem statement is too specific to generalize over an open-source dataset.
- Custom data. Agencies can create or crowd source the data for a fee.

PREPARATION OF DATA:

Data preparation is the action of gathering the data you need, massaging it into a format that's computer-readable and understandable, and asking hard questions of it to check it for completeness and bias. Good data preparation can lead to more accurate and efficient algorithms, while making it easier to pivot to new analytics problems, adapt when model accuracy drifts and save data scientists and business user considerable time and effort down the line.

Data preparation consists of several steps, which consume more time than other aspects of machine learning application development. They are,

- Problem formulation
- Data collection and Discovery
- Data Exploration
- Data cleaning and Validation
- Data structuring
- Feature engineering and selection

PROBLEM FORMULATION:

Data preparation for building machine learning models is a lot more than just cleaning and structuring data. In many cases, it's helpful to begin by stepping back from the data to think about the underlying problem you're trying to solve. "To build a successful ML model," Carroll advised, "you must develop a detailed understanding of the problem to inform what you do and how you do it."

Start by spending time with the people that operate within the domain and have a good understanding of the problem space, synthesizing what you learn through conversations with them and using your experience to create a set of hypotheses that describes the factors and forces involved. This simple step is often skipped or underinvested in, Carroll noted, even though it can make a significant difference in deciding what data to capture. It can also provide useful guidance on how the data should be transformed and prepared for the machine learning model.

DATA COLLECTION AND DISCOVERY:

Once a data science team has formulated the machine learning problem to be solved, it needs to inventory potential data sources within the enterprise and from external third parties. The data collection process must consider not only what the data is purported to represent, but also why it was collected and what it might mean, particularly when used in a different context. It's also essential to consider factors that may have biased the data.

"To reduce and mitigate bias in machine learning models," said Sophia Yang, a senior data scientist at Anaconda, "data scientists need to ask themselves where and how the data was collected to determine if there were significant biases that might have been captured." To train a machine learning model that predicts customer behavior, for example, look at the data and ensure the data set was collected from diverse people, geographical areas and perspectives.

"The most important step often missed in data preparation for machine learning is asking critical questions of data that otherwise looks technically correct," Finkelshteyn said. In addition to investigating bias, he recommended determining if there's reason to believe that important missing data may lead to a partial picture of the analysis being done. In some cases, analytics teams use data that works technically but produces inaccurate or incomplete results, and people who use the resulting models build on these faulty learning without knowing something is wrong.

DATA EXPLORATION:

Data scientists need to fully understand the data they're working with early in the process to cultivate insights into its meaning and applicability. "A common mistake is to launch into model building without taking the time to really understand the data you've wrangled," Carroll said.

Data exploration means reviewing such things as the type and distribution of data contained within each variable, the relationships between variables and how they vary relative to the outcome you're predicting or interested in achieving.

This step can highlight problems like collinearity -- variables that move together -- or situations where standardization of data sets and other data transformations are necessary. It can also surface opportunities to improve model performance, like reducing the dimensionality of a data set.

Data visualizations can also help improve this process. "This might seem like an added step that isn't needed," Yang conjectured, "but our brains are great at spotting patterns along with data that doesn't match the pattern." Data scientists can easily see trends and explore the data correctly by creating suitable visualizations before drawing conclusions. Popular data visualization tools include Tableau, Microsoft Power BI, D3.js and Python libraries such as Matplotlib, Bokeh and the HoloViz stack.

DATA CLEANING AND VALIDATION:

Various data cleansing and validation techniques can help analytics teams identify and rectify inconsistencies, outliers, anomalies, missing data and other issues. Missing data values, for example, can often be addressed with imputation tools that fill empty fields with statistically relevant substitutes.

But Blue Yonder's Wick cautioned that semantic meaning is an often overlooked aspect of missing data. In many cases, creating a dedicated category for capturing the significance of missing values can help. In others, teams may consider explicitly setting missing values as neutral to minimize their impact on machine learning models.

A wide range of commercial and open source tools can be used to cleanse and validate data for machine learning and ensure good quality data. Open source technologies such as Great Expectations and Pandera, for example, are designed to validate the data frames commonly used to organize analytics data into two-dimensional tables. Tools that validate code and data processing workflows are also available. One of them is pytest, which, Yang said, data scientists can use to apply a software development unit-test mindset and manually write tests of their workflows.

DATA STRUCTURING:

Once data science teams are satisfied with their data, they need to consider the machine learning algorithms being used. Most algorithms, for example, work better when data is broken into categories, such as age ranges, rather than left as raw numbers.

Two often-missed data preprocessing tricks, Wick said, are data binning and smoothing continuous features. These data regularization methods can reduce a machine learning model's variance by preventing it from being misled by minor statistical fluctuations in a data set.

FEATURE ENGINEERING AND SELECTION:

The last stage in data preparation before developing a machine learning model is feature engineering and feature selection.

Wick said feature engineering, which involves adding or creating new variables to improve a model's output, is the main craft of data scientists and comes in various forms. Examples include extracting the days of the week or other variables from a data set, decomposing variables into separate features, aggregating variables and transforming features based on probability distributions.

Data scientists also must address feature selection -- choosing relevant features to analyze and eliminating nonrelevant ones. Many features may look promising but lead to problems like extended model training and overfitting, which limits a model's ability to accurately analyze new data. Methods such as lasso regression and automatic relevance determination can help with feature selection.

CHOOSING A MODEL:

Model selection in machine learning is the process of selecting the best algorithm and model architecture for a specific job or dataset. It entails assessing and contrasting various models to identify the one that best fits the data & produces the best results.

TRAINING:

Training data is the initial dataset you use to teach a machine learning application to recognize patterns or perform to your criteria, while testing or validation data is used to evaluate your model's accuracy.

- **Relevant:** You will, of course, need data relevant to the task at hand or the problem you're trying to solve. If your goal is to automate customer support processes, you'd use a dataset of your actual customer support data, or it would be skewed. If you're training a model to analyze social media data, you'll need a dataset from Twitter, Facebook, Instagram, or whichever site you'll be analyzing.
- **Uniform:** All data should come from the same source with the same attributes.
- **Representative:** Your training data must have the same data points and factors as the data you'll be analyzing.
- **Comprehensive:** Your training dataset must be large enough for your needs and have the proper scope and range to encompass all of the model's desired use cases.
- **Diverse:** The dataset should reflect the training and user base, or the results will end up skewed. Ensure those tasked with training the model have no hidden biases or bring in a third party to audit the criteria.

EVALUATION:

Model evaluation is the process of using different evaluation metrics to understand a machine learning model's performance, as well as its strengths and weaknesses. Model evaluation is important to assess the efficacy of a model during initial research phases, and it also plays a role in model monitoring.

It is the process that uses some metrics which help us to analyze the performance of the model. As we all know that model development is a multi-step process and a check should be kept on how well the model generalizes future predictions. Therefore evaluating a model plays a vital role so that we can judge the performance of our model. The evaluation also helps to analyze a model's key weaknesses. There are many metrics like Accuracy, Precision, Recall, F1 score, Area under Curve, Confusion Matrix, and Mean Square Error. Cross

Validation is one technique that is followed during the training phase and it is a model evaluation technique as well.

HYPERPARAMETER TUNING:

Hyperparameters directly control model structure, function, and performance. Hyperparameter tuning allows data scientists to tweak model performance for optimal results. This process is an essential part of machine learning, and choosing appropriate hyperparameter values is crucial for success.

Hyperparameters are external configuration variables that data scientists use to manage machine learning model training. Sometimes called *model hyperparameters*, the hyperparameters are manually set before training a model. They're different from parameters, which are internal parameters automatically derived during the learning process and not set by data scientists.

Examples of hyperparameters include the number of nodes and layers in a neural network and the number of branches in a decision tree. Hyperparameters determine key features such as model architecture, learning rate, and model complexity.

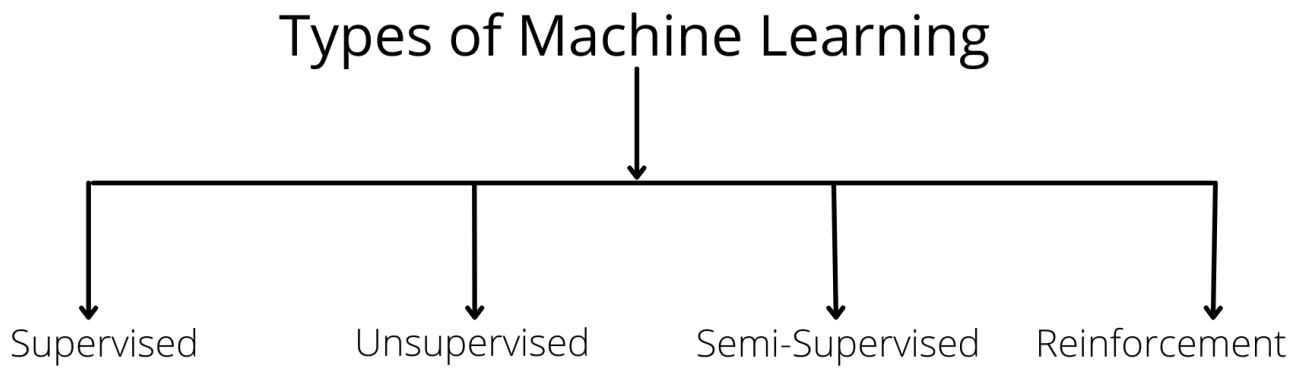
PREDICTION:

There are two types of predictive models. They are Classification models, that predict class membership, and Regression models that predict a number. These models are then made up of algorithms. The algorithms perform the data mining and statistical analysis, determining trends and patterns in data. Predictive analytics software solutions will have built in algorithms that can be used to make predictive models. The algorithms are defined as ‘classifiers’, identifying which set of categories data belongs to.

TYPES OF MACHINE LEARNING ALGORITHMS:

Machine Learning Algorithms are classified into four main types as,

- Supervised
- Unsupervised
- Semi-supervised
- Reinforcement



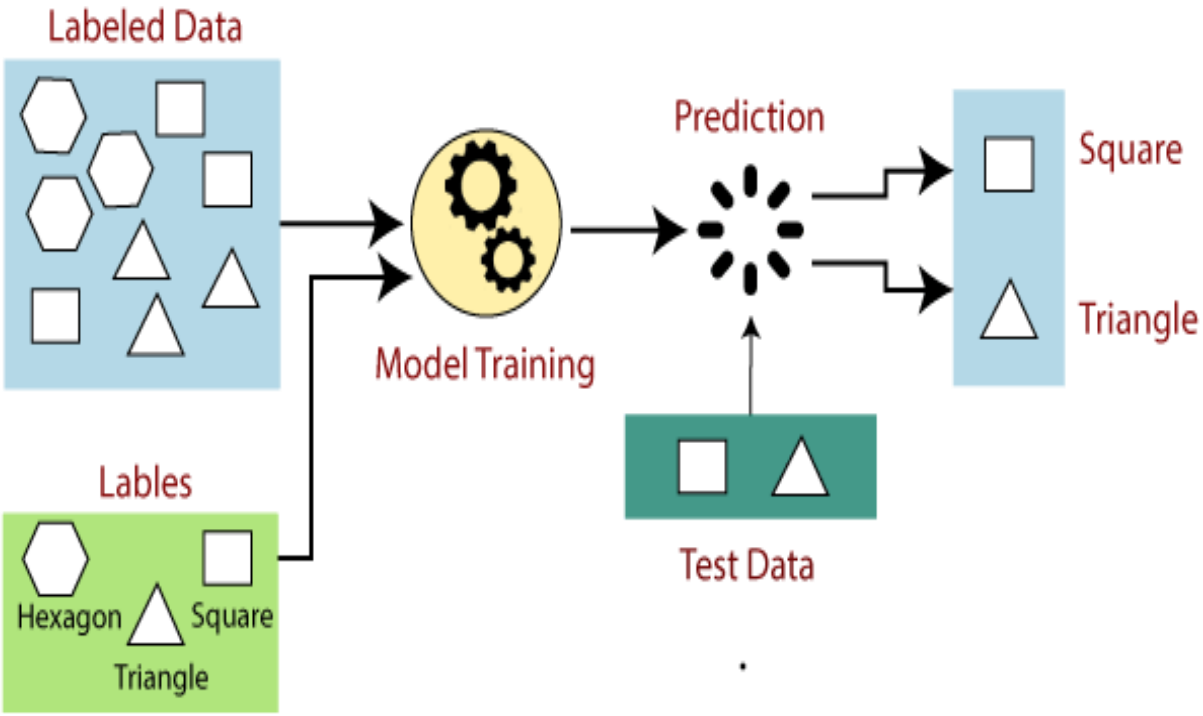
SUPERVISED LEARNING ALGORITHM:

Supervised learning is the types of machine learning in which machines are trained using well "labeled" training data, and on basis of that data, machines predict the output. The labeled data means some input data is already tagged with the correct output. In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher. Supervised learning is a process of providing input data as well as correct output data to the machine learning model.

HOW SUPERVISED LEARNING WORKS?

In supervised learning, models are trained using labeled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

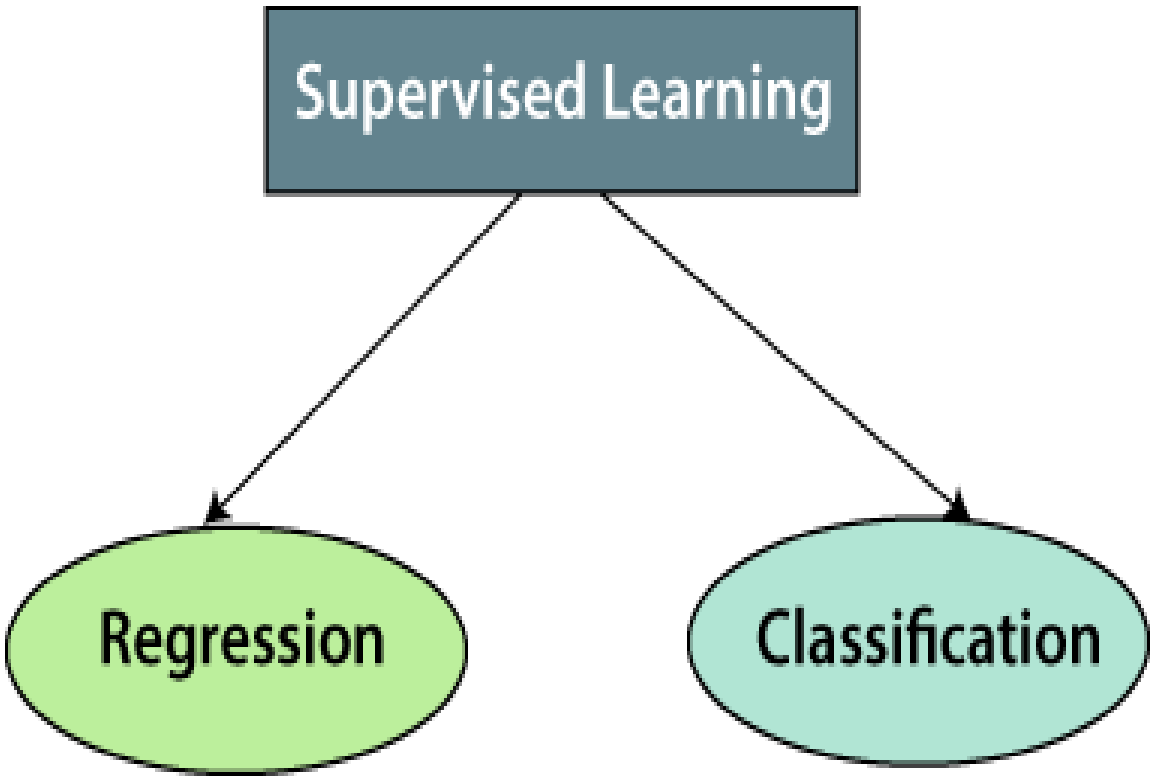
- If the given shape has four sides, and all the sides are equal, then it will be labeled as a **Square**.
- If the given shape has three sides, then it will be labeled as a **triangle**.
- If the given shape has six equal sides then it will be labeled as **hexagon**.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

TYPES OF SUPERVISED MACHINE LEARNING ALGORITHM:

Supervised learning can be further divided into two types of problems:



REGRESSION:

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

CLASSIFICATION:

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

Spam Filtering,

- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

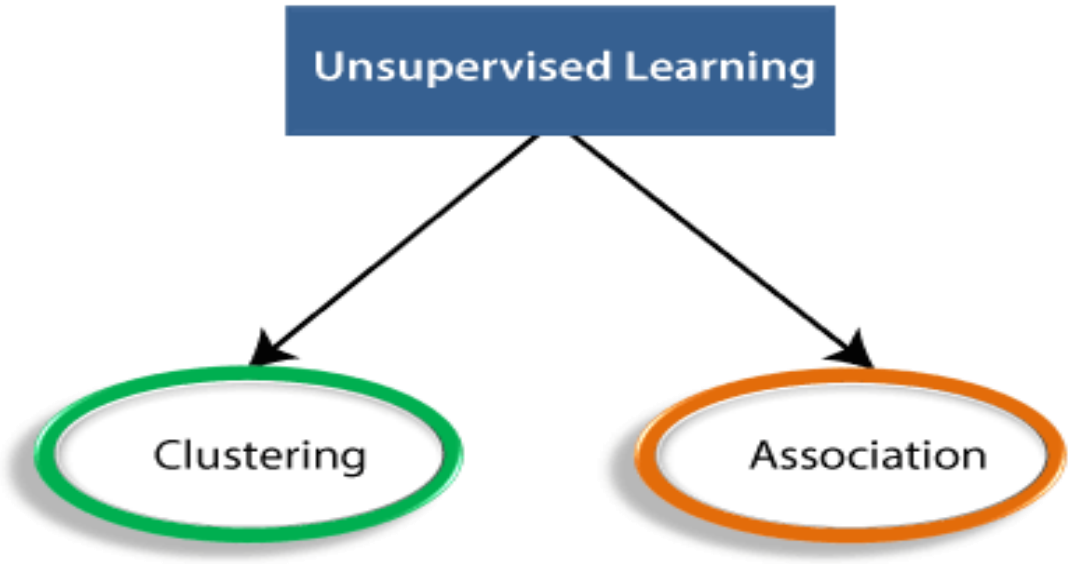
UNSUPERVISED LEARNING ALGORITHM:

Unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.

TYPES OF UNSUPERVISED MACHINE LEARNING ALGORITHM:

The unsupervised learning algorithm can be further categorized into two types of problems:



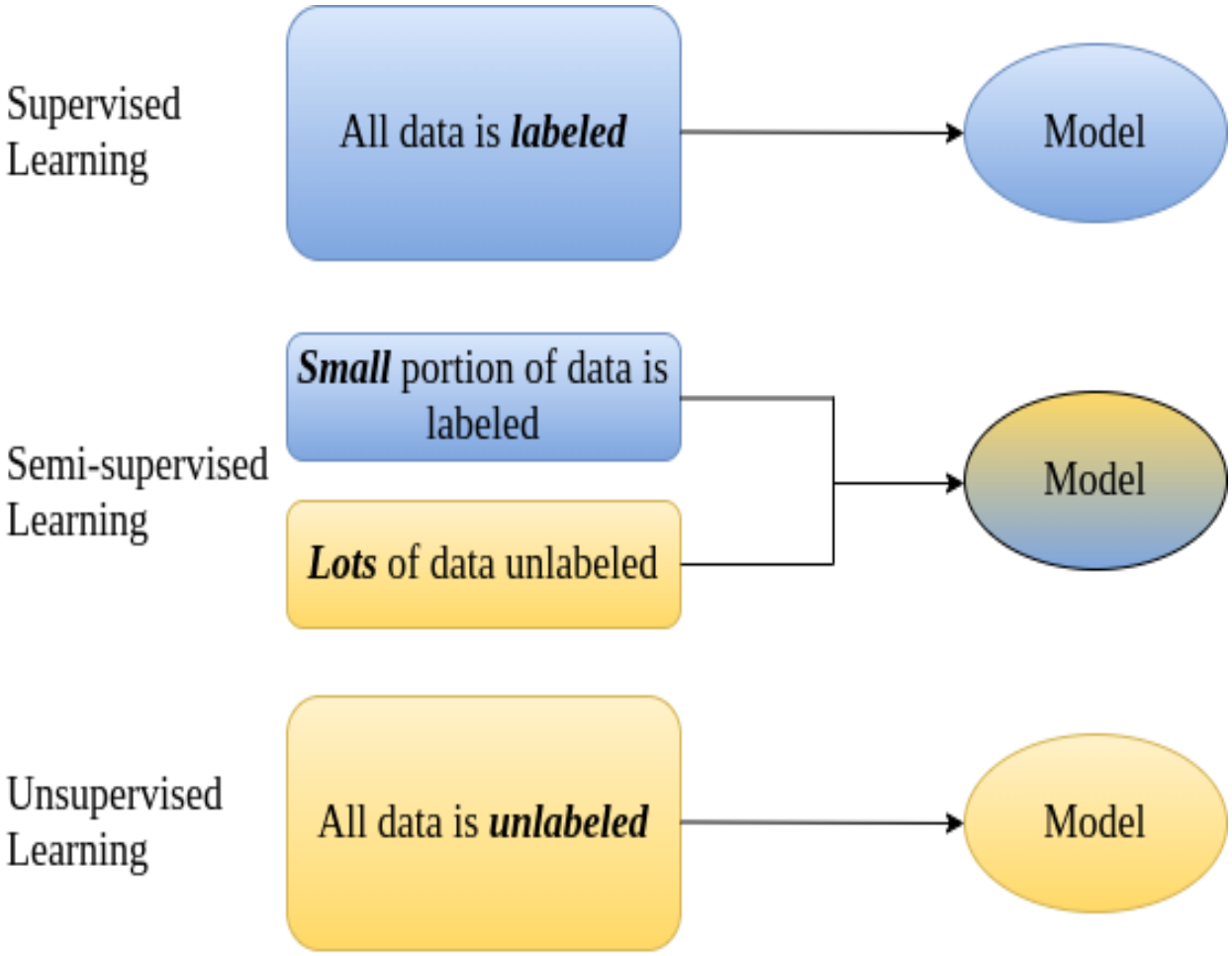
- **Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

- **Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Semi-Supervised Learning:

Semi-supervised learning (SSL) is a machine learning technique that uses a small portion of labeled data and lots of unlabeled data to train a predictive model.

To better understand the SSL concept, we should look at it through the prism of its two main counterparts: supervised learning and unsupervised learning. Their differences are shown in the below graphic.

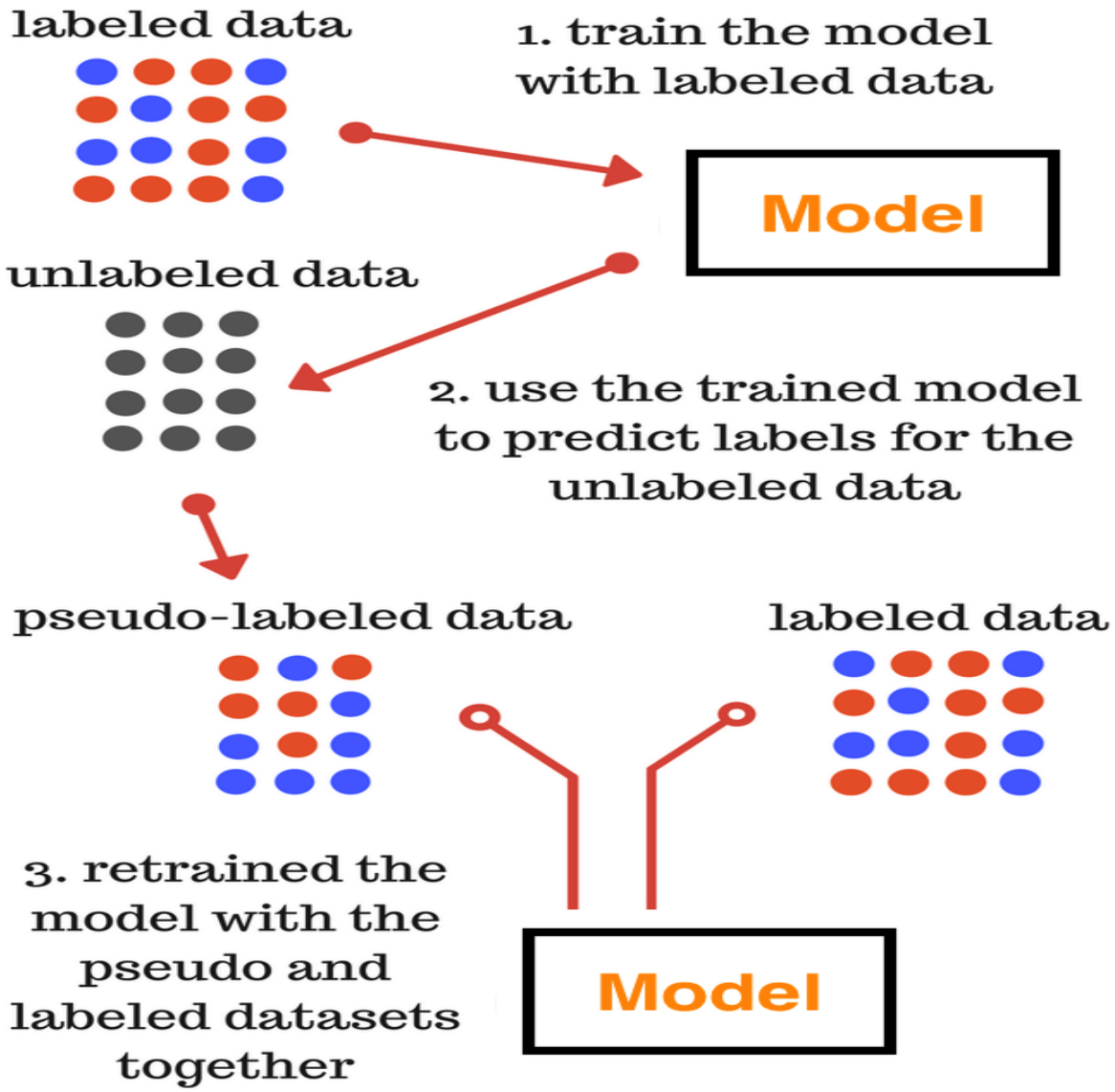


Techniques Used in Semi-Supervised Learning:

There are several techniques for doing SSL, which are described below.

Pseudo-labeling:

Instead of manually labeling the unlabelled data, we give our model approximate labels on the basis of the labeled data. Let's explain pseudo-labeling by breaking the concept into steps as shown in the image below.



The image above describes a process where:

- We train a model with labeled data.
- We use the trained model to predict labels for the unlabeled data, which creates pseudo-labeled data.
- We retrain the model with the pseudo-labeled and labeled data together.

This process happens iteratively as the model improves and is able to perform with a greater degree of accuracy.

Self-Training:

Self-training is a variation of pseudo labeling. The difference with self-training is that we accept only the predictions that have a high confidence and we iterate through this process several times. In pseudo-labeling, however, there is no boundary of confidence that must be met for a prediction to be used in a model. The standard workflow is as follows.

First, you pick a small amount of labeled data, e.g., images showing cats and dogs with their respective tags, and you use this dataset to train a base model with the help of ordinary supervised methods.

Then you apply the process of *pseudo-labeling* — when you take the partially trained model and use it to make predictions for the rest of the database which is yet unlabeled. The labels generated thereafter are called *pseudo* as they are produced based on the originally labeled data that has limitations (say, there may be an uneven representation of classes in the set resulting in bias — more dogs than cats).

From this point, you take the most confident predictions made with your model (for example, you want the confidence of over 80 per cent that a certain image shows a cat, not a dog). If any of the pseudo-labels exceed this confidence level, you add them to the labeled dataset and create a new, combined input to train an improved model.

The process can go through several iterations (10 is a standard amount) with more and more pseudo-labels being added every time. Provided the data is suitable for the process, the performance of the model will keep increasing at each iteration.

Label Propagation:

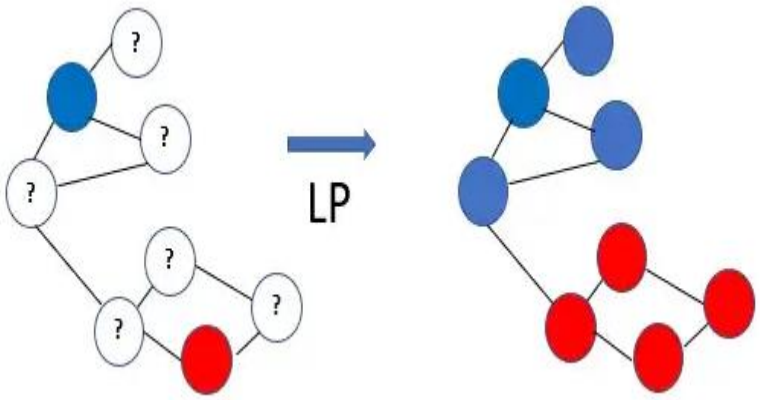
This technique is a graph-based transductive method to infer pseudo-labels for unlabeled data. Unlabeled data points iteratively adopt the label of the majority of their neighbours based on the labeled data points.

Label propagation (LP) makes a few assumptions:

- All classes for the dataset are present in the labeled data;
- Data points that are close have similar labels and;
- Data points in the same cluster will likely have the same label.

Label propagation creates a fully connected graph where the nodes are all the labeled and unlabeled data points. The edges between the two nodes are weighted. The shorter the euclidean distance between two nodes, the larger the weight will be. A larger edge weight allows the label to "travel" easily in the model.

A simple explanation of the working of the label propagation algorithm is shown below:



A representation of what Labeling propagation tries to achieve.

The standard workflow is as follows:

- All nodes have soft labels assigned based on the distribution of labels;
- Labels of a node are propagated to all nodes through edges;
- Each node will update its label iteratively based on the maximum number of nodes in its neighbourhood. The label of a node is persisted from the labeled data, making it possible to infer a broad range of traits that are assortative along the edges of a graph.

The label propagation algorithm stops when every node for the unlabeled data point has the majority label of its neighbor or the number of iterations defined is reached.

REINFORCEMENT LEARNING:

- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that ***"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."*** How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.
- It is a core part of Artificial Intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.

TYPES OF REINFORCEMENT LEARNING:

There are mainly two types of reinforcement learning, which are:

- **Positive Reinforcement**
- **Negative Reinforcement**

POSITIVE REINFORCEMENT:

- The positive reinforcement learning means adding something to increase the tendency that expected behavior would occur again. It impacts positively on the behavior of the agent and increases the strength of the behavior.
- This type of reinforcement can sustain the changes for a long time, but too much positive reinforcement may lead to an overload of states that can reduce the consequences.

NEGATIVE REINFORCEMENT:

- The negative reinforcement learning is opposite to the positive reinforcement as it increases the tendency that the specific behavior will occur again by avoiding the negative condition.
- It can be more effective than the positive reinforcement depending on situation and behavior, but it provides reinforcement only to meet minimum behavior.

REINFORCEMENT LEARNING APPLICATIONS:

- **Robotics:**

  RL is used in **Robot navigation, Robo-soccer, walking, juggling**, etc.

- **Control:**

  RL can be used for **adaptive control** such as Factory processes, admission control in telecommunication, and Helicopter pilot is an example of reinforcement learning.

- **Game Playing:**

  RL can be used in **Game playing** such as tic-tac-toe, chess, etc.

- **Chemistry:**

  RL can be used for optimizing the chemical reactions.

- **Business:**

  RL is now used for business strategy planning.

- **Manufacturing:**

  In various automobile manufacturing companies, the robots use deep reinforcement learning to pick goods and put them in some containers.

- **Finance Sector:**

  The RL is currently used in the finance sector for evaluating trading strategies.

MACHINE LEARNING ALGORITHM:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read_csv("https://github.com/amankharwal/Website-data/blob/master/database.csv")

data.columns
```

```
Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'Depth Error',
       'Depth Seismic Stations', 'Magnitude', 'Magnitude Type',
       'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap',
       'Horizontal Distance', 'Horizontal Error', 'Root Mean Square', 'ID',
       'Source', 'Location Source', 'Magnitude Source', 'Status'],
      dtype='object')
```

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

data.head()
```

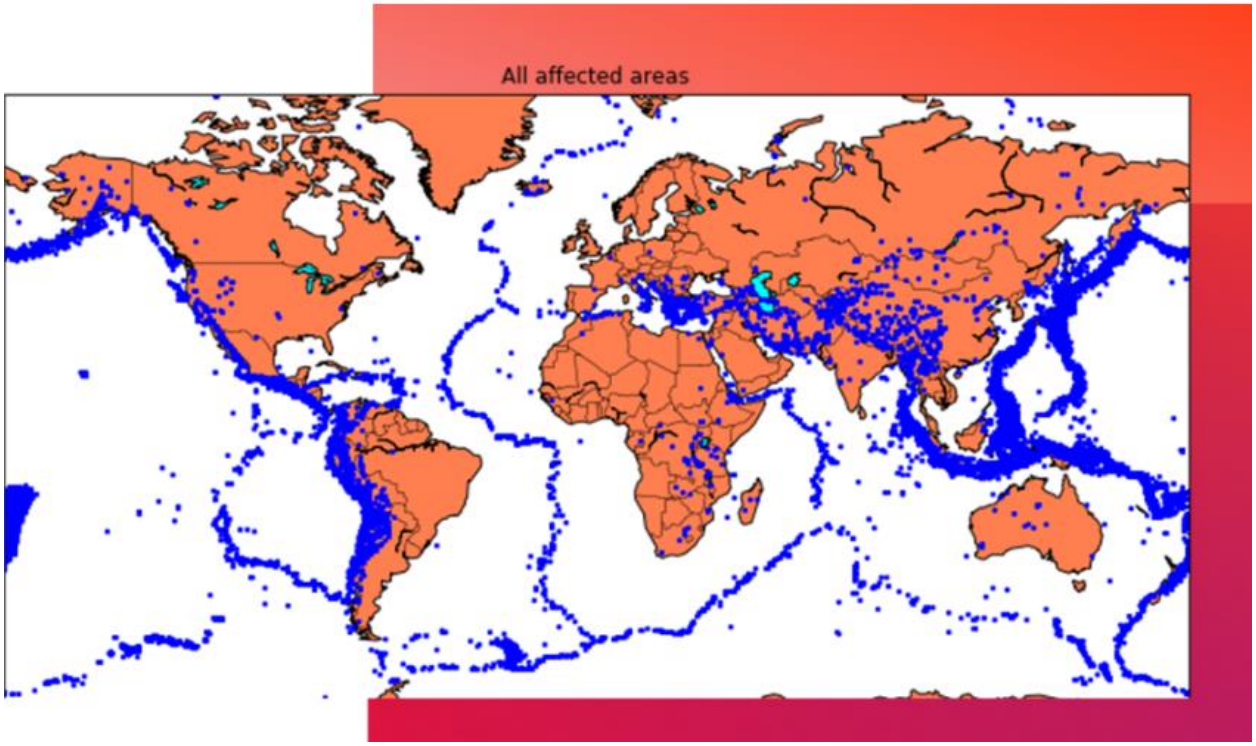
	date	Time	Latitude	Longitude	Depth	Magnitude
0	01/02/1965	13:44:18	19.246	145.616	131.6	6.0
1	01/04/1965	11:29:49	1.863	127.352	80.0	5.8
2	01/05/1965	18:05:58	-20.579	-173.972	20.0	6.2
3	01/08/1965	18:49:43	-59.076	-23.557	15.0	5.8
4	01/09/1965	13:32:50	11.938	126.427	15.0	5.8

```
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()
```

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	-1.57631e+08
1	1.863	127.352	80.0	5.8	-1.57466e+08
2	-20.579	-173.972	20.0	6.2	-1.57356e+08
3	-59.076	-23.557	15.0	5.8	-1.57094e+08
4	11.938	126.427	15.0	5.8	-1.57026e+08

```
from mpl_toolkits.basemap import Basemap

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 = 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)
```

```
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))
```

```
Best: 0.957655 using {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss':
'squared_hinge', 'neurons': 16, 'optimizer': 'SGD'} 0.333316 (0.471398) with: {'activation':
'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer':
'SGD'} 0.000000 (0.000000) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10,
'loss': 'squared_hinge', 'neurons': 16, 'optimizer': 'Adadelta'} 0.957655 (0.029957) with:
{'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_hinge', 'neurons': 16,
'optimizer': 'SGD'} 0.645111 (0.456960) with: {'activation': 'relu', 'batch_size': 10, 'epochs':
10, 'loss': 'squared_hinge', 'neurons': 16, 'optimizer': 'Adadelta'}
```

```
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))

[test_loss, test_acc] = model.evaluate(X_test, y_test)

print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))
```

```
Evaluation result on Test Data : Loss = 0.5038455790406056, accuracy = 0.9241777017858995
```

CONCLUSION:

- The Machine Learning Algorithm describes that the steps of algorithms as gathering data, preparing that data, choosing the model, training, evaluate, hyperparameter tuning, prediction and the types of algorithms as supervised, unsupervised, semi-supervised, reinforcement and the machine learning algorithm code of earthquake prediction using python.

- With theses foundation steps completed, our dataset is now primed for the subsequent stages of building and training a earthquake prediction model using the machine learning algorithms.