A
Report
on

"EARTHQUAKE PREDICTION"
Using Machine Learning

by

NANDITA DAS (20IUT0020138)
SAIKAT DAS (20IUT0020139)
ISHITA SAHA (20IUT0020150)
SHREYASEE DEBNATH (20IUT0020136)
SORAJIT NATH (20IUT0020206)
ANKUR BHOWMIK (18IUT0010067)

Submitted in partial fulfillment of the course requirement of
SPECIAL PROJECT II EC492

For The award of the degree
Bachelor of Technology in Electronics & Communication Engineering
AT
ICFAI UNIVERSITY TRIPURA

As
SPECIAL PROJECT

Under the guidance of
Ms. Debarshita Biswas
Asst. Prof. FST, IUT

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
ICFAI TECHNICAL SCHOOL
FACULTY OF SCIENCE & TECHNOLOGY
THE ICFAI UNIVERSITY

April 2023

The ICFAI University, Tripura

Established under section 4(2) of the Institute of Chartered Financial Analysts of India University, Tripura Act, 2004 Campus Address: Kamalghat (Near Agartala), PIN 799210, Tel 0381-2865752/62, Fax 0381-2865754 Website: www.iutripura.edu.in, Email: registrar@iutripura.edu.in

FACULTY OF SCIENCE & TECHNOLOGY

ICFAI TECHNICAL SCHOOL

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that **Nandita Das, Ishita Saha, Sorajit Nath, Saikat Das, Shreyasee Debnath, Ankur Bhowmik** have satisfactory carried out Special Project-II (EC492) work entitled "**EARTHQUAKE Prediction using Machine Learning ",** for the degree of **Bachelor of Technology** in **Electronics And Communication Engineering** of **ICFAI University Tripura**.

Signature of Project Guide **Signature of HOD(ECE)**

DECLARATION

We hereby declare that the work recorded in this project report entitled **" Earthquake prediction using machine learning"** in fulfillment of the course requirement of Special project for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering from **ICFAI University, Tripura** is a faithful and bonafied project work carried out as special project under the guidance of Ms. Debarshita Biswas., Asst. Prof. ma'am.

The assistance and help received during the course of the investigation have been duly acknowledged.

1. Nandita Das(ID No-20IUT0020138)

2. Sorajit Nath (ID No-20IUT0020206)

3. Shreyasee Debnath (ID No-20IUT0020136)

4. Ishita Saha (ID No-20IUT0020150)

5. Ankur Bhowmik(ID No-18IUT0010067)

6. Saikat Das (ID No-20IUT0020139)

Date:30.03.2023

ACKNOWLEDGEMENT

The satisfaction that accompanies on successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

Firstly, we pay our gratitude to ICFAI University, Tripura for including this internship as part of the course fulfillment to get a practical exposure which will help us to grow into successful professionals in the future. We would also like to thank Dean, FST Dr. Priyangshu Rana Borthakur Sir, Principal, ITS Dr. Prasanta Kumar Sinha Sir, HOD, ECE Prof. Arindam Banerjee sir and Ms. Debarshita Biswas, Asst.prof. ma'am for their guidance to do this internship.

We take this opportunity to express our profound gratitude and deep regards to our Department Faculties and our Special project Guide Ms. Debarshita Biswas, Asst.prof. for their exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by them time to time shall carry us a long way in the journey of life on which we are about to embark.

We are obliged to our project team members for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of my assignment.

Finally, we would like to thank all our friends and family members specially our parents for providing us with moral support in course of our project work.

1. Nandita Das (ID No- 20IUT0020138)
2. Saikat Das (ID No- 20IUT0020139)
3. Ishita Saha (ID N0- 20IUT0020150)
4. Shreyasee Debnath (ID No-20IUT0020136)
5. Sorajit Nath (ID No-20IUT0020206)
6.Ankur Bhowmik(ID No-18IUT0010067)

ABSTRACT

Prediction of earthquakes using machine learning is an active research area aimed at developing accurate and reliable models for earthquake forecasting. Machine learning algorithms have been applied to various types of earthquake data, including seismic signals, geodetic data, and geological information, to identify patterns and relationships that can help predict future earthquakes. The goal of this research is to develop models that can accurately forecast the time, location, and magnitude of earthquakes, which can help improve disaster preparedness and response. While significant progress has been made in this field, predicting earthquakes remains a challenging task due to the complex and nonlinear nature of earthquake processes. Nevertheless, the use of machine learning in earthquake prediction shows promising results and may contribute to reducing the impact of earthquakes on human life and infrastructure.

Signature of Project Guide

Date:

Signature of HOD(ECE)

Date: **Signature(s) of Student(s)**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

CHAPTER 1

1. INTRODUCTION TO MECHINE LEARNING

Machine learning is a branch of artificial intelligence that involves the development of algorithms and statistical models that enable computers to automatically learn from data and improve their performance on a specific task over time without being explicitly programmed.

The process of machine learning involves feeding a computer system with large amounts of data, which it can use to identify patterns and relationships within the data. The computer system then uses this information to make predictions or decisions about new, unseen data.

There are several types of machine learning algorithms, including supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the computer system is trained on labeled data, where each data point has an associated target or output. In unsupervised learning, the computer system is trained on unlabeled data, where it must identify patterns or groupings within the data without any prior knowledge of what the output should be. In reinforcement learning, the computer system learns through trial and error, receiving rewards or punishments based on its actions in a given environment.

Machine learning has numerous applications in various fields, including image recognition, natural language processing, recommender systems, and fraud detection, among others. The ability of computers to learn and improve their performance over time has made machine learning an essential tool for businesses and organizations looking to gain insights from their data and automate decision-making processes. Machine learning is a branch of artificial intelligence that involves the development of algorithms and statistical models that enable computers to automatically learn from data and improve their performance on a specific task over time without being explicitly programmed.

The process of machine learning involves feeding a computer system with large amounts of data, which it can use to identify patterns and relationships within the data. The computer system then uses this information to make predictions or decisions about new, unseen data.

There are several types of machine learning algorithms, including supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the computer system is trained on labeled data, where each data point has an associated target or output. In unsupervised learning, the computer system is trained on unlabeled data, where it must identify patterns or groupings within the data without any prior knowledge of what the output should be. In reinforcement learning, the computer system learns through trial and error, receiving rewards or punishments based on its actions in a given environment.

Machine learning has numerous applications in various fields, including image recognition, natural language processing, recommender systems, and fraud detection, among others. The ability of computers to learn and improve their performance over time has made machine learning an essential tool for businesses and organizations looking to gain insights from their data and automate decision-making processes.

1.2 Classification of Machine Learning

Machine learning implementations are classified into four major categories, depending on the nature of the learning "signal" or "response" available to a learning system which are as follows:

A. Supervised learning:

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled 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. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y)**.**

In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering**,** etc.

How Supervised Learning Works?

In supervised learning, models are trained using labelled 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 tan it predicts the output.

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

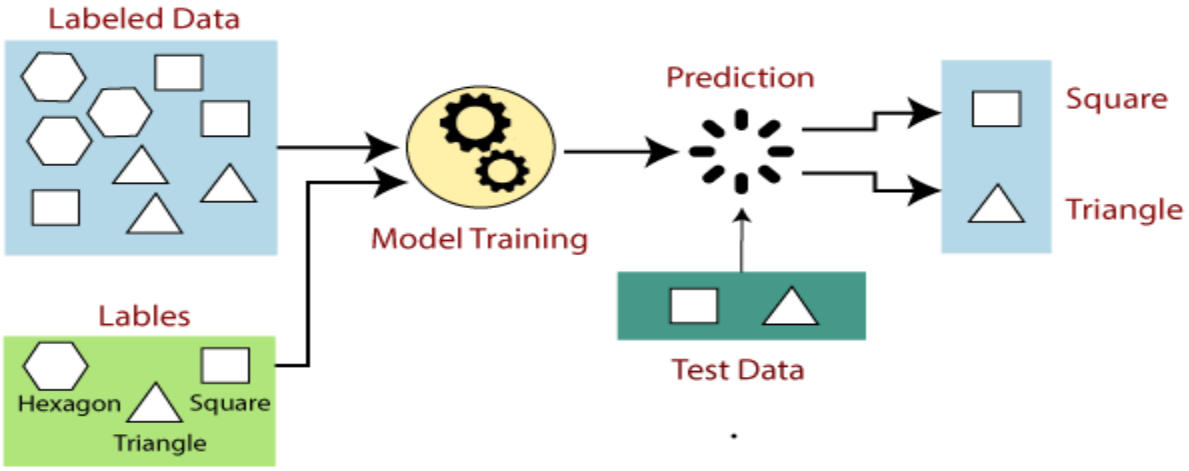


Fig: Supervised learning

Suppose we has 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 labelled as a **Square**.
- If the given shape has three sides, then it will be labelled as a **triangle**.
- If the given shape has six equal sides then it will be labelled 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.

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training **dataset, test dataset, and validation dataset**.
- Determine the input features of the training dataset, which should of enough no ledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If t the model predicts the correct output, which means our model is accurate.

Types of supervised Machine learning Algorithms:

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

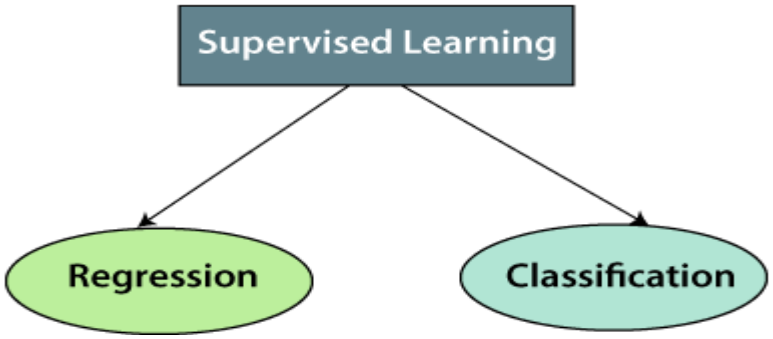


Fig: Types of supervised Machine learning Algorithm

1. 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

2. Classification

Classification algorithms are used when teh 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

Advantages of Supervised learning:

- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering**, etc.

Disadvantages of supervised learning:

- Supervised learning models are not suitable for handling the complex tasks.
- Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- Training required lots of computation times.

- In supervised learning, we need enough knowledge about the classes of object.

Applications of Supervised Learning

Some common applications of Supervised Learning are given below:

- **Image Segmentation:**

  Supervised Learning algorithms are used in image segmentation. In this process, image classification is performed on different image data with pre-defined labels.

- **Medical Diagnosis:**

  Supervised algorithms are also used in the medical field for diagnosis purposes. It is done by using medical images and past labelled data with labels for disease conditions. With such a process, the machine can identify a disease for the new patients.

- **Fraud Detection :**

  Supervised Learning classification algorithms are used for identifying fraud transactions, fraud customers, etc. It is done by using historic data to identify the patterns that can lead to possible fraud.

- **Spam detection :**

  In spam detection & filtering, classification algorithms are used. These algorithms classify an email as spam or not spam. The spam emails are sent to the spam folder.

- **Speech Recognition :**

  Supervised learning algorithms are also used in speech recognition. The algorithm is trained with voice data, and various identifications can be done using the same, such as voice-activated passwords, voice commands, etc.

B. Unsupervised machine learning

As the name suggests, 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. Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data

but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:

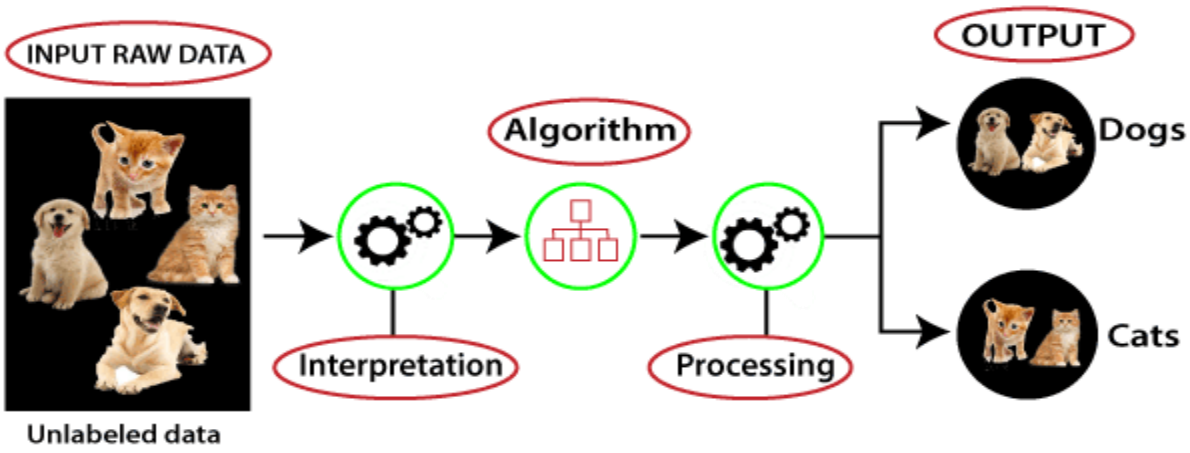


Fig: Unsupervised Learning

Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

Types of Unsupervised Learning Algorithm:

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

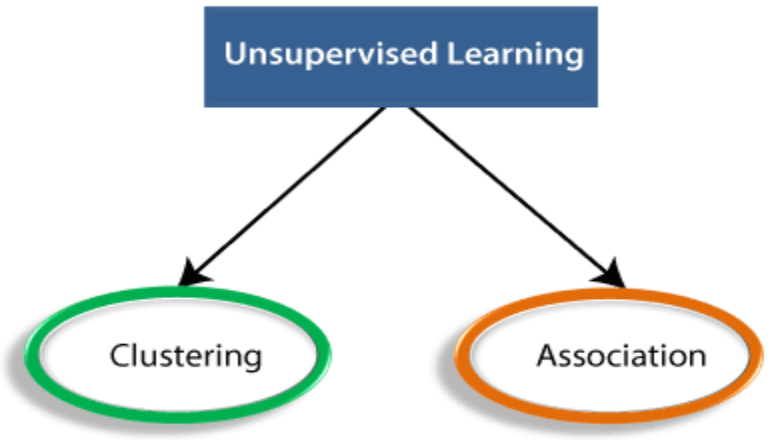


Fig: Unsupervised Learning Algorithm Types

- **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.

Unsupervised Learning algorithms:

Below is the list of some popular unsupervised learning algorithms:

- K-means clustering
- KNN (k-nearest neighbours)
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis
- Independent Component Analysis
- Apriori algorithm
- Singular value decomposition

Advantages of Unsupervised Learning

- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

Disadvantages of Unsupervised Learning

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

Applications of Unsupervised Learning

- **Network Analysis:** Unsupervised learning is used for identifying plagiarism and copyright in document network analysis of text data for scholarly articles.
- **Recommendation Systems:** Recommendation systems widely use unsupervised learning techniques for building recommendation applications for different web applications and e-commerce websites.
- **Anomaly Detection:** Anomaly detection is a popular application of unsupervised learning, which can identify unusual data points within the dataset. It is used to discover fraudulent transactions.
- **Singular Value Decomposition:** Singular Value Decomposition or SVD is used to extract particular information from the database. For example, extracting information of each user located at a particular location.

1.3. Semi-Supervised Learning

Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning**.** It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabeled datasets during the training period.

Although Semi-supervised learning is the middle ground between supervised and unsupervised learning and operates on the data that consists of a few labels, it mostly consists of unlabeled data. As labels are costly, but for corporate purposes, they may have few labels. It is completely different from supervised and unsupervised learning as they are based on the presence & absence of labels.

To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced. The main aim of semi-supervised learning is to effectively use all the available data, rather than only labelled data like in supervised learning. Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabeled data into labelled data. It is because labelled data is a comparatively more expensive acquisition than unlabeled data.

We can imagine these algorithms with an example. Supervised learning is where a student is under the supervision of an instructor at home and college. Further, if that student is self- analysing the same concept without any help from the instructor, it comes under unsupervised learning. Under semi-supervised learning, the student has to revise himself after analyzing the same concept under the guidance of an instructor at college.

Advantages and disadvantages of Semi-supervised Learning

Advantages:

- It is simple and easy to understand the algorithm.
- It is highly efficient.
- It is used to solve drawbacks of Supervised and Unsupervised Learning algorithms.

Disadvantages:

- Iterations results may not be stable.
- We cannot apply these algorithms to network-level data.
- Accuracy is low.

1.4. Reinforcement Learning

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance. Agent gets rewarded for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.

Due to its way of working, reinforcement learning is employed in different fields such as Game theory, Operation Research, Information theory, multi-agent systems.

A reinforcement learning problem can be formalized using Markov Decision Process (MDP). In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.

Categories of Reinforcement Learning

Reinforcement learning is categorized mainly into two types of methods/algorithms:

- **Positive Reinforcement Learning:** Positive reinforcement learning specifies increasing the tendency that the required behaviour would occur again by adding something. It enhances the strength of the behaviour of the agent and positively impacts it.
- **Negative Reinforcement Learning:** Negative reinforcement learning works exactly opposite to the positive RL. It increases the tendency that the specific behaviour would occur again by avoiding the negative condition.

Real-world Use cases of Reinforcement Learning

- **Video Games:**
  RL algorithms are much popular in gaming applications. It is used to gain super-human performance. Some popular games that use RL algorithms are AlphaGO **and** AlphaGO Zero**.**
- **Resource Management:**
  The "Resource Management with Deep Reinforcement Learning" paper showed that how to use RL in computer to automatically learn and schedule resources to wait for different jobs in order to minimize average job slowdown.
- **Robotics:**
  RL is widely being used in Robotics applications. Robots are used in the industrial and manufacturing area, and these robots are made more powerful with reinforcement learning. There are different industries that have their vision of building intelligent robots using AI and Machine learning technology.

- **Text Mining**

  Text-mining, one of the great applications of NLP, is now being implemented with the help of Reinforcement Learning by Salesforce company.

Advantages and Disadvantages of Reinforcement Learning

Advantages

- It helps in solving complex real-world problems which are difficult to be solved by general techniques.
- The learning model of RL is similar to the learning of human beings; hence most accurate results can be found.
- Helps in achieving long term results.

Disadvantage

- RL algorithms are not preferred for simple problems.
- RL algorithms require huge data and computations.
- Too much reinforcement learning can lead to an overload of states which can weaken the results.

CHAPTER 2

OBJECTIVE

CHAPTER 2

The prediction of earthquakes using machine learning techniques aims to provide early warning and reduce the potential impact of seismic activity on human lives and infrastructure. Earthquakes are a natural phenomenon that can cause significant damage to buildings, roads, and other critical infrastructure, resulting in economic losses and loss of life. Machine learning algorithms can analyze large amounts of data and identify patterns that may be indicators of an impending earthquake.

One objective of earthquake prediction is to improve our understanding of the underlying processes that cause earthquakes. Machine learning can help identify factors such as changes in the earth's crust or seismic activity patterns that are associated with earthquakes. By analyzing this data, scientists can better understand how earthquakes occur and what warning signs to look for.

Another objective is to develop accurate and reliable earthquake prediction models. Machine learning algorithms can be trained to analyze historical seismic data and other environmental factors, such as temperature and pressure changes, to predict the likelihood of an earthquake. These models can be used to develop early warning systems that can provide crucial information to people living in areas at risk of earthquakes.

Finally, the goal of earthquake prediction using machine learning is to save lives and protect infrastructure. By providing early warning, people can take action to protect themselves, such as evacuating buildings or seeking shelter. Emergency responders can also be better prepared to respond to earthquake-related emergencies, reducing the impact of the disaster.

In conclusion, the prediction of earthquakes using machine learning has the potential to revolutionize our understanding of seismic activity and improve our ability to mitigate the impact of earthquakes on society. Through the development of accurate prediction models and early warning systems, machine learning can help save lives and protect critical infrastructure.

CHAPTER 3

METHOD

METHOD

3.1 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm

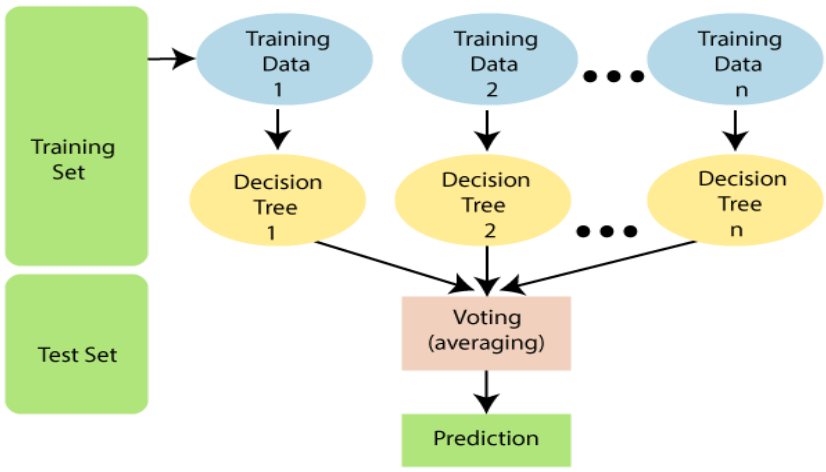


Fig: Random Forest Algorithm

Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.

- The predictions from each tree must have very low correlations.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

<=”” li=””>

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:

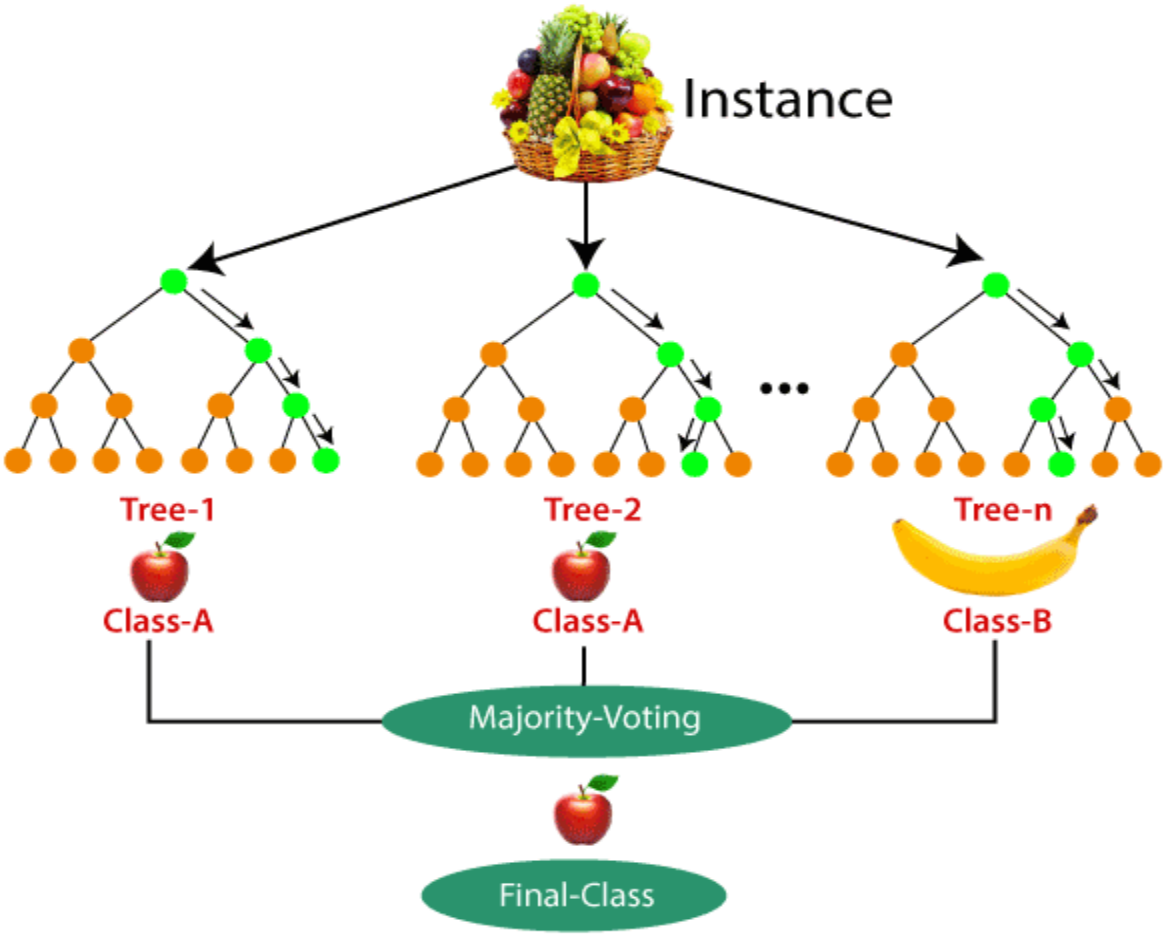


Fig: Decision Tree

Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

Python Implementation of Random Forest Algorithm

Now we will implement the Random Forest Algorithm tree using Python. For this, we will use the same dataset "user_data.csv", which we have used in previous classification models. By using the same dataset, we can compare the Random Forest classifier with other classification models such as Decision tree Classifier, KNN, SVM, Logistic Regression, etc.

Implementation Steps are given below:

- Data Pre-processing step
- Fitting the Random forest algorithm to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.

Python Implementation of Random Forest Algorithm

Now we will implement the Random Forest Algorithm tree using Python. For this, we will use the same dataset "user_data.csv", which we have used in previous classification models. By using the same dataset, we can compare the Random Forest classifier with other classification models such as Decision tree Classifier, KNN, SVM, Logistic Regression, etc.

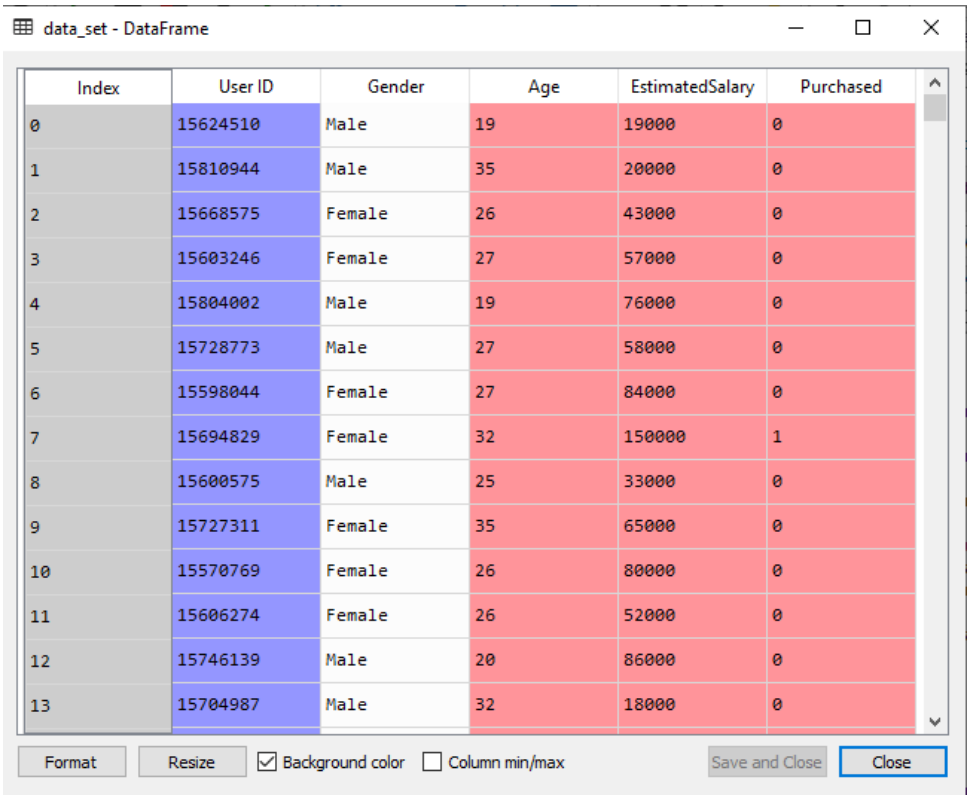
Implementation Steps are given below:

- Data Pre-processing step
- Fitting the Random forest algorithm to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.

1.Data Pre-Processing Step:

Below is the code for the pre-processing step:

```
1. # importing libraries
2. import numpy as nm
3. import matplotlib.pyplot as mtp
4. import pandas as pd
5.
6. #importing datasets
7. data_set= pd.read_csv('user_data.csv')
8.
9. #Extracting Independent and dependent Variable
10. x= data_set.iloc[:, [2,3]].values
11. y= data_set.iloc[:, 4].values
12.
13. # Splitting the dataset into training and test set.
14. from sklearn.model_selection import train_test_split
15. x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
16.
17. #feature Scaling
18. from sklearn.preprocessing import StandardScaler
```



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

```
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

In the above code, we have pre-processed the data. Where we have loaded the dataset, which is given as:

2. Fitting the Random Forest algorithm to the training set:

Now we will fit the Random forest algorithm to the training set. To fit it, we will import the **RandomForestClassifier** class from the **sklearn.ensemble** library. The code is given below:

```
#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)
```

In the above code, the classifier object takes below parameters:

- **n_estimators=** The required number of trees in the Random Forest. The default value is 10. We can choose any number but need to take care of the overfitting issue.
- **criterion=** It is a function to analyze the accuracy of the split. Here we have taken "entropy" for the information gain.

Output:

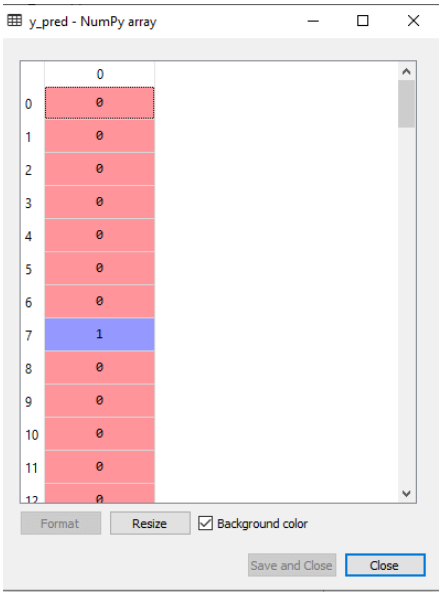
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

3. Predicting the Test Set result

Since our model is fitted to the training set, so now we can predict the test result. For prediction, we will create a new prediction vector y_pred. Below is the code for it:

```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

Output: The prediction vector is given as:



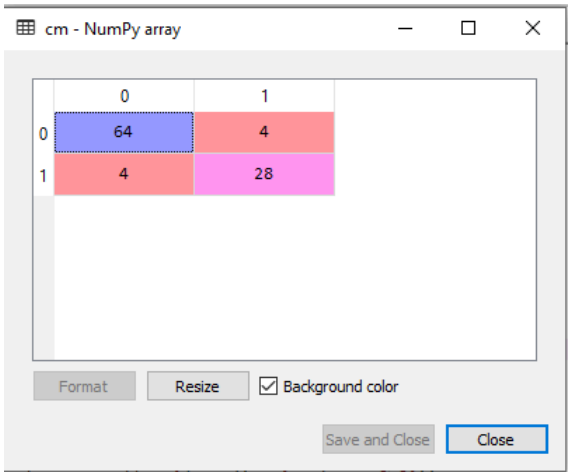
By checking the above prediction vector and test set real vector, we can determine the incorrect predictions done by the classifier.

4. Creating the Confusion Matrix

Now we will create the confusion matrix to determine the correct and incorrect predictions. Below is the code for it:

```
1. #Creating the Confusion matrix
2. from sklearn.metrics import confusion_matrix
3. cm= confusion_matrix(y_test, y_pred)
```

Output:



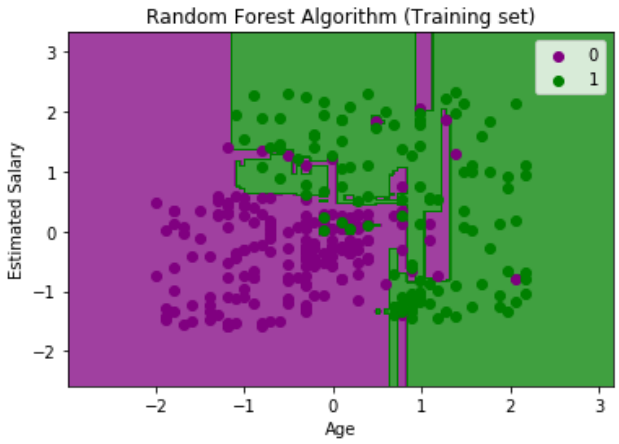
As we can see in the above matrix, there are **4+4= 8 incorrect predictions** and **64+28= 92 correct predictions.**

5. Visualizing the training Set result

Here we will visualize the training set result. To visualize the training set result we will plot a graph for the Random forest classifier. The classifier will predict yes or No for the users who have either Purchased or Not purchased the SUV car as we did in Logistic Regression. Below is the code for it:

```
1. from matplotlib.colors import ListedColormap
2. x_set, y_set = x_train, y_train
3. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
4. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
5. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
6. alpha = 0.75, cmap = ListedColormap(('purple','green' )))
7. mtp.xlim(x1.min(), x1.max())
8. mtp.ylim(x2.min(), x2.max())
9. for i, j in enumerate(nm.unique(y_set)):
10.     mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
11.         c = ListedColormap(('purple', 'green'))(i), label = j)
12. mtp.title('Random Forest Algorithm (Training set)')
13. mtp.xlabel('Age')
14. mtp.ylabel('Estimated Salary')
15. mtp.legend()
16. mtp.show()
```

Output:



The above image is the visualization result for the Random Forest classifier working with the training set result. It is very much similar to the Decision tree classifier. Each data point corresponds to each user of the user_data, and the purple and green regions are the prediction regions. The purple region is classified for the users who did not purchase the SUV car, and the green region is for the users who purchased the SUV.

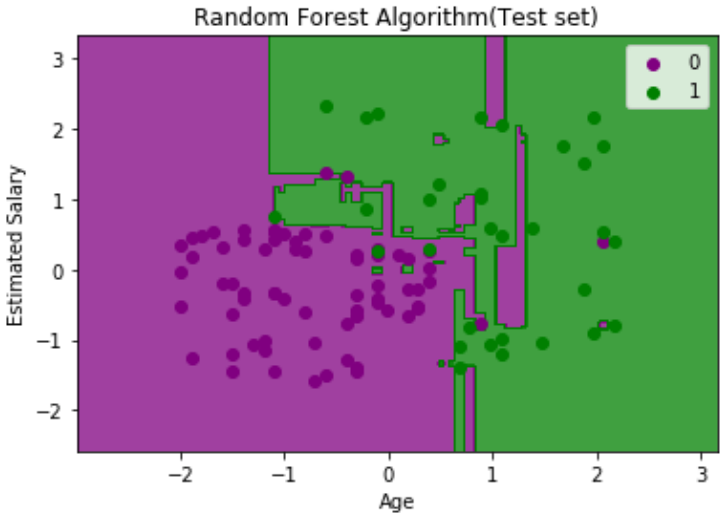
So, in the Random Forest classifier, we have taken 10 trees that have predicted Yes or NO for the Purchased variable. The classifier took the majority of the predictions and provided the result.

6. Visualizing the test set result

Now we will visualize the test set result. Below is the code for it:

```
#Visulaizing the test set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
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('Random Forest Algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:



3.2 kaggle

Kaggle is an online community and platform for data scientists, machine learning engineers, and anyone interested in solving complex data problems. The platform was founded in 2010 and acquired by Google in 2017. It provides a platform for data enthusiasts to participate in data science competitions, collaborate on data-driven projects, and learn from the community.

One of the most popular features of Kaggle is its data science competitions, where users compete to produce the best predictive model for a given problem. These competitions are sponsored by companies and organizations seeking innovative solutions to their data problems. Participants compete for cash prizes, recognition, and the opportunity to showcase their skills to potential employers. Kaggle has hosted competitions for a wide range of industries, including healthcare, finance, and technology.

In addition to competitions, Kaggle also offers a wide range of datasets for users to work on. These datasets are provided by companies, organizations, and individuals, and cover a wide range of topics, including climate change, financial markets, and social media. Users can explore and analyze these datasets using various tools, including Python and R.

Kaggle also provides a platform for collaboration, where users can join or create teams to work on data projects together. This allows users to learn from each other, share ideas, and collaborate on complex data problems.

Finally, Kaggle provides a variety of educational resources for users to learn more about data science, machine learning, and AI. These resources include tutorials, courses, and forums, where users can ask questions and get help from the community.

In summary, Kaggle is a vibrant and thriving community of data enthusiasts, offering a wide range of opportunities for data scientists to showcase their skills, learn new techniques, and collaborate on complex data-driven projects.

3.3 SVM(Support Vector Machine)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

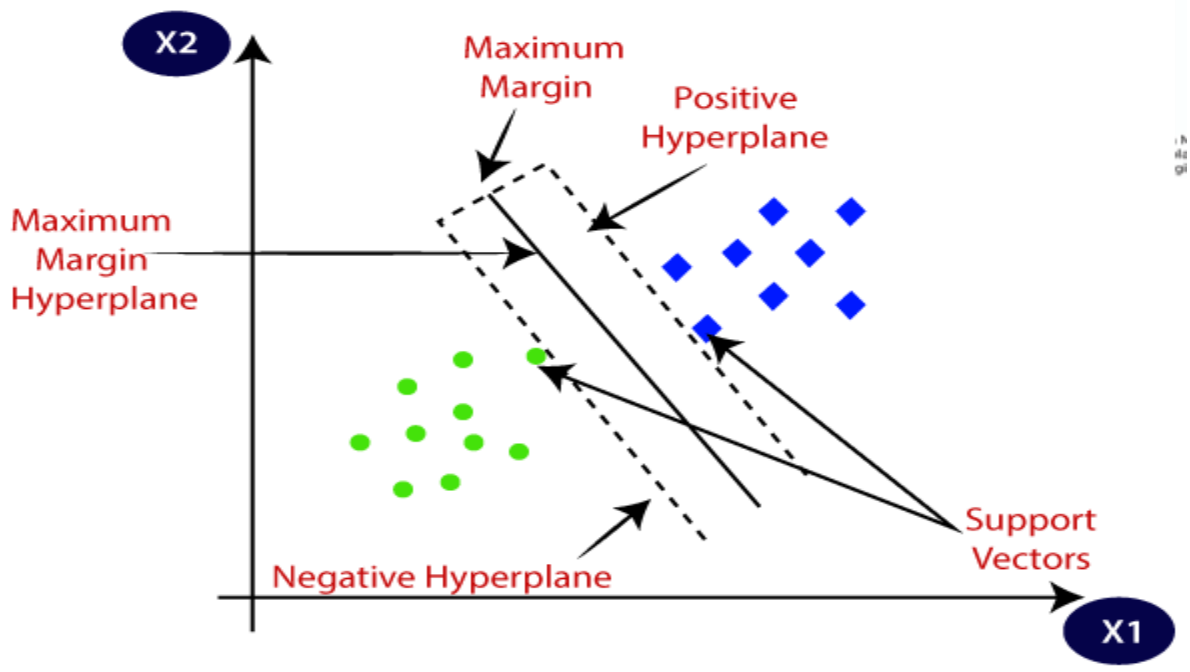


Fig: Support Vector Machine

Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs,

and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:

PlayNext
Unmute

Current TimeÂ 0:00

/

DurationÂ 18:10
Loaded: 0.37%
Â
Fullscreen
Backward Skip 10sPlay VideoForward Skip 10s

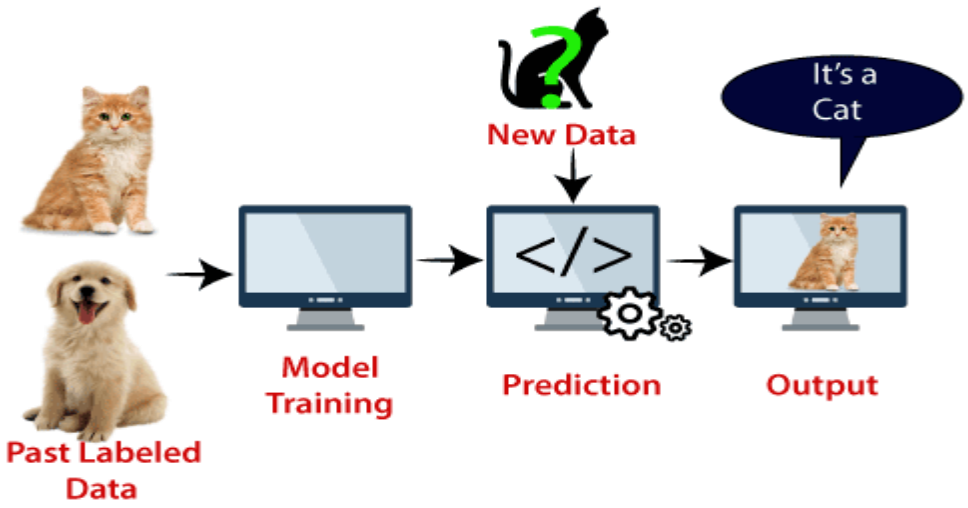


Fig: Example of SVM

SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

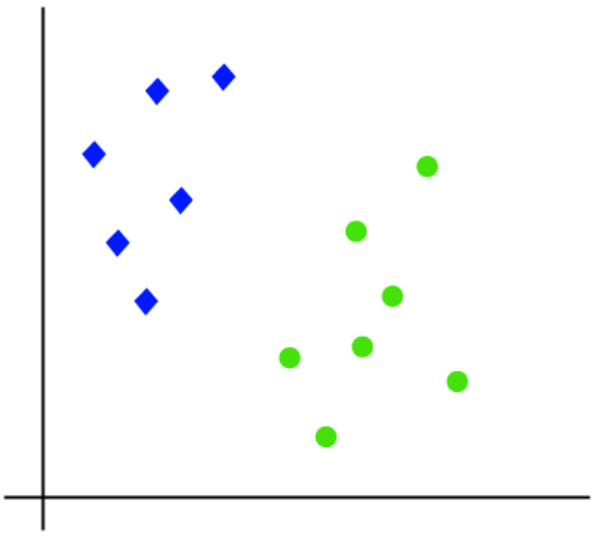


Fig: Hyper plane and support vectors in the SVM Algorithm

How does SVM works?

Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

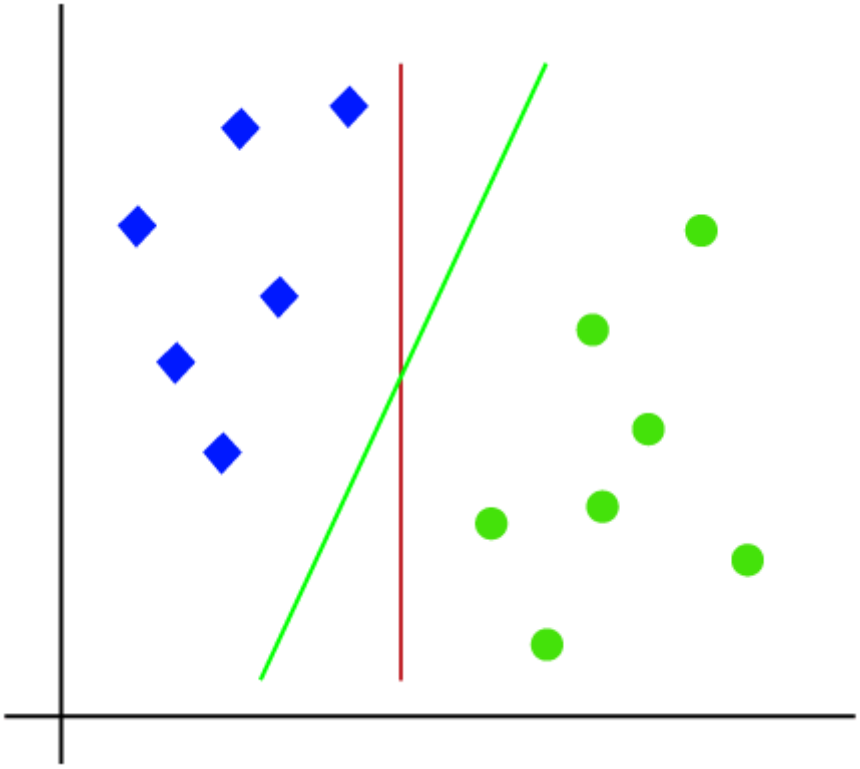


Fig: Linear SVM

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.

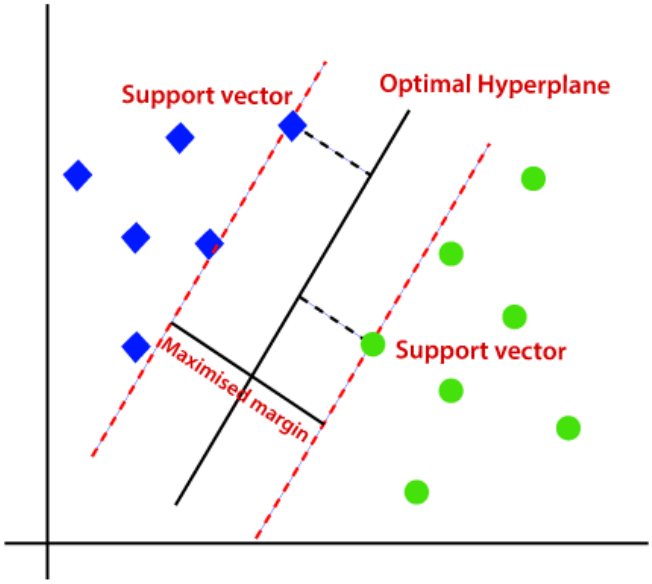


Fig Support Vector and Optimal Hyperplane

Non-Linear SVM:

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:

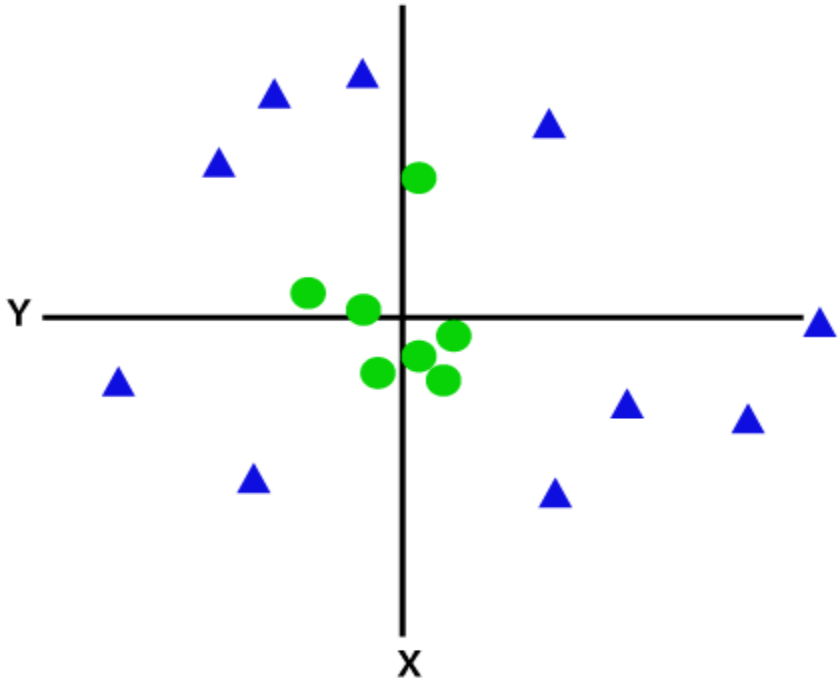


Fig: Non Linear SVM

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x^2 +y^2$

By adding the third dimension, the sample space will become as below image:

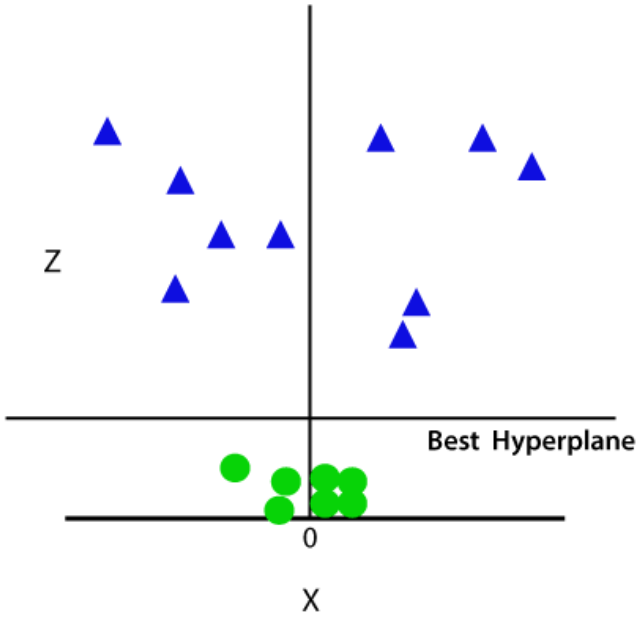


Fig: Non Linear SVM

So now, SVM will divide the datasets into classes in the following way. Consider the below image:

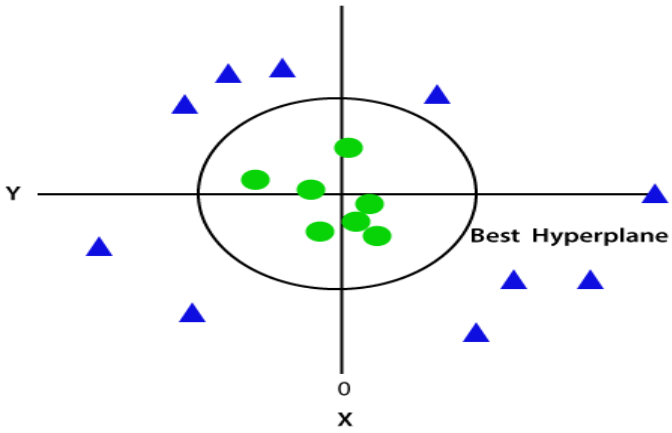


Fig: Non Linear SVM

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

Hence we get a circumference of radius 1 in case of non-linear data.

Python Implementation of Support Vector Machine

Now we will implement the SVM algorithm using Python. Here we will use the same dataset **user_data**, which we have used in Logistic regression and KNN classification.

- ### Data Pre-processing step

Till the Data pre-processing step, the code will remain the same. Below is the code:

```
#Data Pre-processing Step
# 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)
```

After executing the above code, we will pre-process the data. The code will give the dataset as:

data_set - 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

Format | Resize | Background color | Column min/max | Save and Close | Close

The scaled output for the test set will be:

x_test - NumPy array

	0	1
0	-0.804802	0.504964
1	-0.0125441	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.0459058
9	0.878746	-0.770734
10	-0.804802	-0.596776
11	-1.00287	-0.422817
12	-0.111576	-0.422817

Format | Resize | Background color | Save and Close | Close

y_test - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

Format | Resize | Background color | Save and Close | Close

Fitting the SVM classifier to the training set:

Now the training set will be fitted to the SVM classifier. To create the SVM classifier, we will import **SVC** class from **Sklearn.svm** library. Below is the code for it:

```
1. from sklearn.svm import SVC # "Support vector classifier"
2. classifier = SVC(kernel='linear', random_state=0)
3. classifier.fit(x_train, y_train)
```

In the above code, we have used **kernel='linear'**, as here we are creating SVM for linearly separable data. However, we can change it for non-linear data. And then we fitted the classifier to the training dataset(x_train, y_train)

Output:

```
Out[8]:
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
   decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
   kernel='linear', max_iter=-1, probability=False, random_state=0,
   shrinking=True, tol=0.001, verbose=False)
```

The model performance can be altered by changing the value of **C(Regularization factor), gamma, and kernel**.

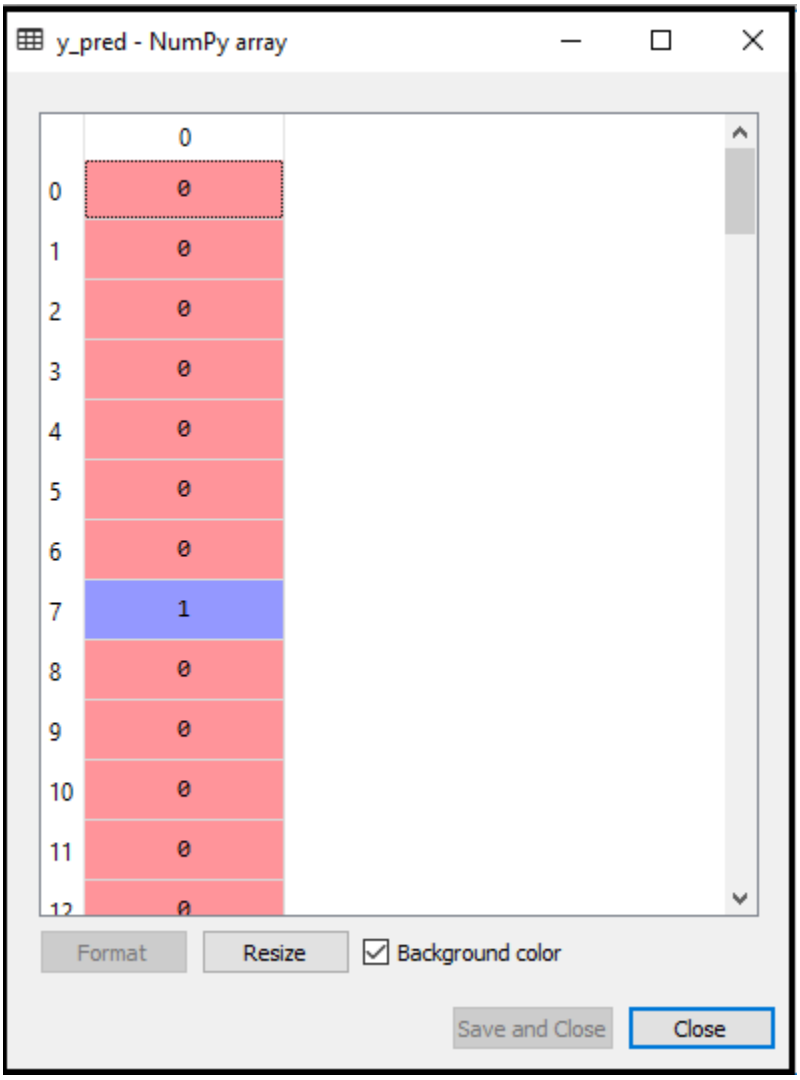
- ## Predicting the test set result:

  Now, we will predict the output for test set. For this, we will create a new vector y_pred. Below is the code for it:

```
1. #Predicting the test set result
2. y_pred= classifier.predict(x_test)
```

After getting the y_pred vector, we can compare the result of **y_pred** and **y_test** to check the difference between the actual value and predicted value.

Output: Below is the output for the prediction of the test set:



- **Creating the confusion matrix:**

  Now we will see the performance of the SVM classifier that how many incorrect predictions are there as compared to the Logistic regression classifier. To create the confusion matrix, we need to import the **confusion_matrix** function of the sklearn library. After importing the function, we will call it using a new variable **cm**. The function takes two parameters, mainly **y_true**(the actual values) and **y_pred** (the targeted value return by the classifier). Below is the code for it:

```
1. #Creating the Confusion matrix
2. from sklearn.metrics import confusion_matrix
3. cm= confusion_matrix(y_test, y_pred)
```

Output:

As we can see in the above output image, there are 66+24= 90 correct predictions and 8+2= 10 correct predictions. Therefore we can say that our SVM model improved as compared to the Logistic regression model.

- **Visualizing the training set result:**

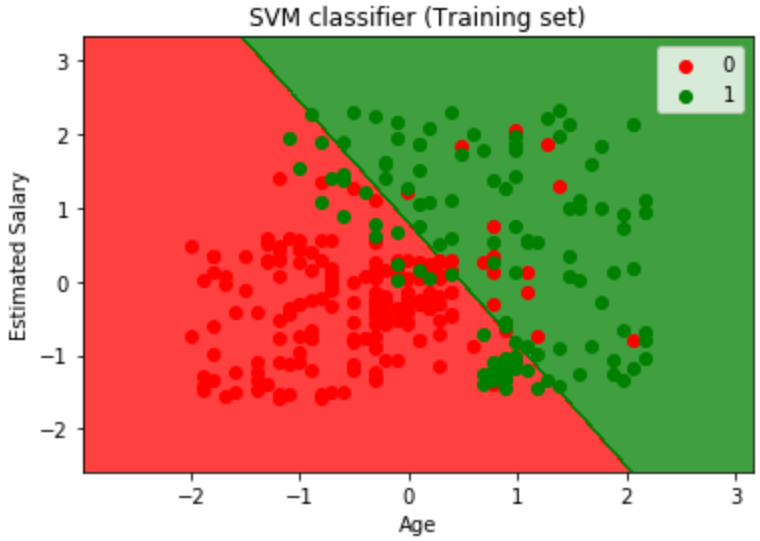
  Now we will visualize the training set result, below is the code for it:

```
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(('red', '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(('red', 'green'))(i), label = j)
mtp.title('SVM classifier (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
```

15. mtp.legend()
16. mtp.show()

Output:

By executing the above code, we will get the output as:



As we can see, the above output is appearing similar to the Logistic regression output. In the output, we got the straight line as hyperplane because we have **used a linear kernel in the classifier**. And we have also discussed above that for the 2d space, the hyperplane in SVM is a straight line.

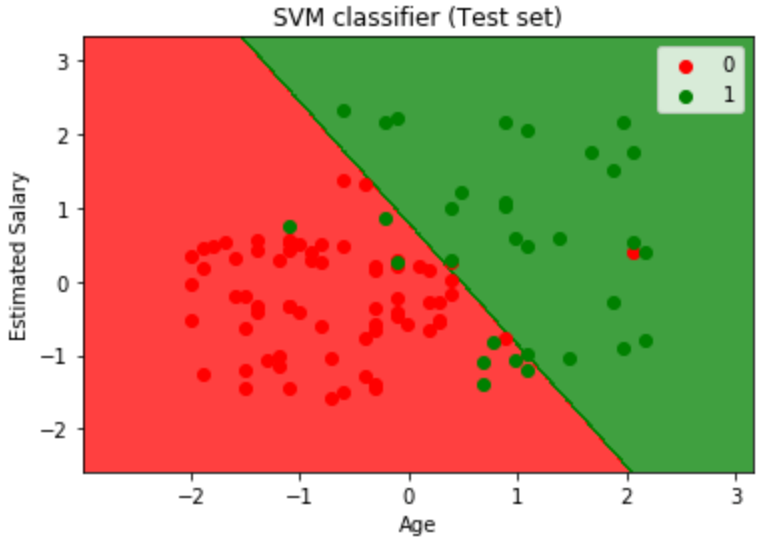
- **Visualizing the test set result:**

```
#Visulaizing the test set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
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(('red','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(('red', 'green'))(i), label = j)
```

13. mtp.title('SVM classifier (Test set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()

Output:

By executing the above code, we will get the output as:



As we can see in the above output image, the SVM classifier has divided the users into two regions (Purchased or Not purchased). Users who purchased the SUV are in the red region with the red scatter points. And users who did not purchase the SUV are in the green region with green scatter points. The hyperplane has divided the two classes into Purchased and not purchased variable.

3.4 Gradient Boosting Algorithm

Gradient boosting is a method standing out for its prediction speed and accuracy, particularly with large and complex datasets. From Kaggle competitions to machine learning solutions for business, this algorithm has produced the best results. We already know that errors play a major role in any machine learning algorithm. There are mainly two types of error, bias error and variance error. Gradient boost algorithm *helps us minimize bias error* of the model

Before getting into the details of this algorithm we must have some knowledge about AdaBoost Algorithm which is again a boosting method. This algorithm starts by building a decision stump and then assigning equal weights to all the data points. Then it increases the weights for all the points which are misclassified and lowers the weight for those that are easy to classify or are correctly classified. A new decision stump is made for these weighted data points. The idea behind this is to improve the predictions made by the first

stump. I have talked more about this algorithm here. Read this article before starting this algorithm to get a better understanding.

What is boosting?

While studying machine learning you must have come across this term called Boosting. It is the most misinterpreted term in the field of Data Science. The principle behind boosting algorithms is first we built a model on the training dataset, then a second model is built to rectify the errors present in the first model. Let me try to explain to you what exactly does this means and how does this works.

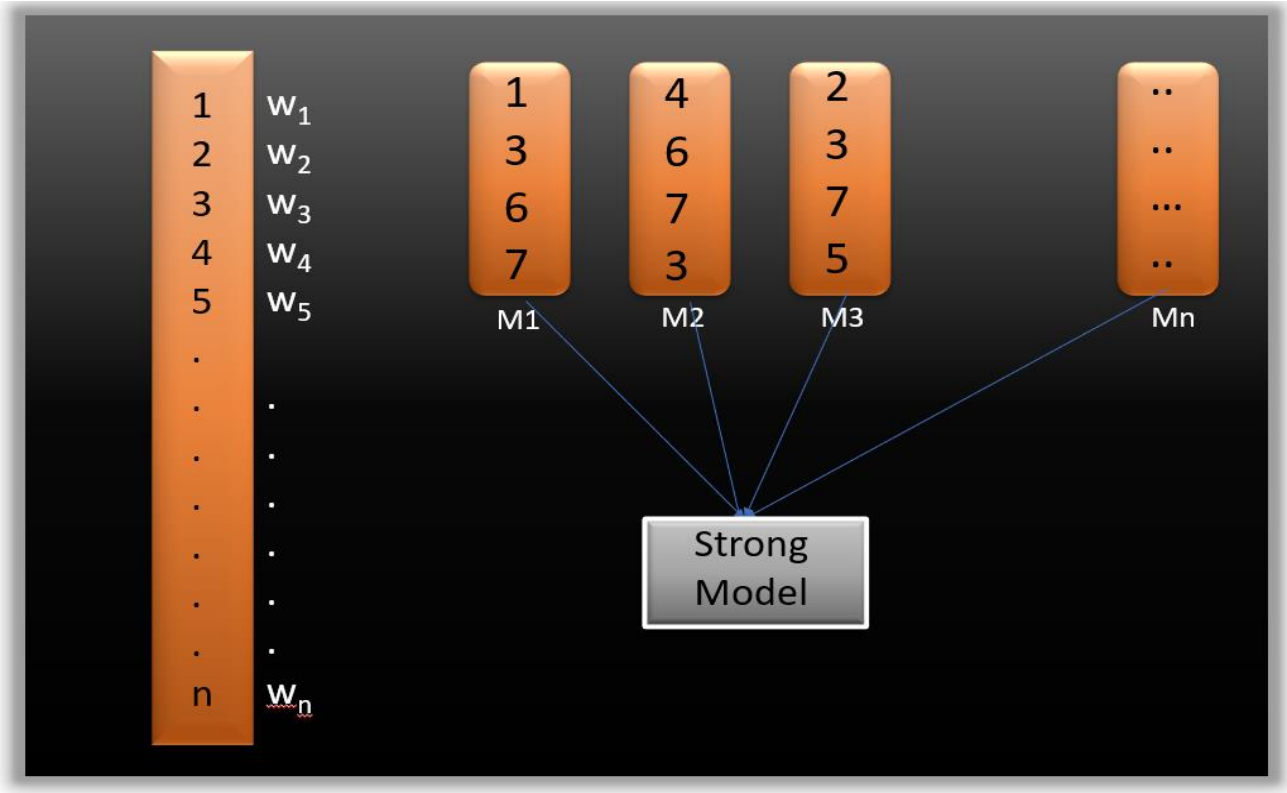


Fig: Boosting Algorithm

Suppose you have n data points and 2 output classes (0 and 1). You want to create a model to detect the class of the test data. Now what we do is randomly select observations from the training dataset and feed them to model 1 (M1), we also assume that initially, all the observations have an equal weight that means an equal probability of getting selected.

Remember in assembling techniques the weak learners combine to make a strong model so here M1, M2,

M3….Mn all are weak learners.

Since M1 is a weak learner, it will surely misclassify some of the observations. Now before feeding the observations to M2 what we do is update the weights of the observations which are wrongly classified. You can think of it as a bag that initially contains 10 different color balls but after some time some kid takes out his favorite color ball and put 4 red color balls instead inside the bag. Now off-course the probability of selecting a red ball is higher. This same phenomenon happens in Boosting techniques, when an observation is wrongly classified, its weight get's updated and for those which are correctly classified, their weights get decreased. The probability of selecting a wrongly classified observation gets increased hence in the next model only those observations get selected which were misclassified in model 1.

Similarly, it happens with M2, the wrongly classified weights are again updated and then fed to M3. This procedure is continued until and unless the errors are minimized, and the dataset is predicted correctly. Now when the new datapoint comes in (Test data) it passes through all the models (weak learners) and the class which gets the highest vote is the output for our test data.

What is a Gradient boosting Algorithm?

The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model. But how do we do that? How do we reduce the error? This is done by building a new model on the errors or residuals of the previous model.

When the target column is continuous, we use **Gradient Boosting Regressor** whereas when it is a classification problem, we use **Gradient Boosting Classifier**. The only difference between the two is the *"Loss function"*. The objective here is to minimize this loss function by adding weak learners using gradient descent. Since it is based on loss function hence for regression problems, we'll have different loss functions like Mean squared error (**MSE**) and for classification, we will have different for e.g **log-likelihood**.

CHAPTER 4

WORKING OPERATION

4. CHAPTER 4

WORKING OPERATION

Earthquake prediction using machine learning is an active research area, but predicting earthquakes with high accuracy and reliability is a challenging task. While machine learning algorithms have shown promising results in various fields, predicting earthquakes is still considered a complex problem that requires a combination of multiple approaches.

Here are some of the working operations of earthquake prediction using machine learning:

Data Collection: The first step in predicting earthquakes using machine learning is to collect data from various sources. Data can be collected from seismometers, GPS receivers, satellite images, and other sensors that measure the deformation of the Earth's crust. This data is then processed and analyzed to extract features that can be used for training the machine learning model.

Feature Extraction: Feature extraction involves the process of identifying patterns and features from the collected data. This process is important as it determines the quality of the input data that is fed into the machine learning model. There are various techniques used for feature extraction, including Fourier transforms, wavelet transforms, and Principal Component Analysis (PCA).

Training the Model: Once the data is collected and features are extracted, the next step is to train the machine learning model. This involves selecting an appropriate machine learning algorithm, splitting the data into training and testing sets, and selecting appropriate hyperparameters. The most commonly used algorithms for earthquake prediction are decision trees, random forests, neural networks, and support vector machines.

Testing and Validation: Once the model is trained, it needs to be tested and validated to ensure its accuracy and reliability. This is done by using the testing set and comparing the predicted results with the actual results. Various performance metrics are used to evaluate the model, such as accuracy, precision, recall, and F1-score.

Deployment: After the model is tested and validated, it is deployed in real-world scenarios to predict earthquakes. The model takes input from various sensors and provides predictions based on the learned patterns.

In summary, earthquake prediction using machine learning involves collecting data, feature extraction, training the model, testing and validation, and deployment. While machine learning algorithms have shown promising results, predicting earthquakes is still a complex problem that requires further research and development.

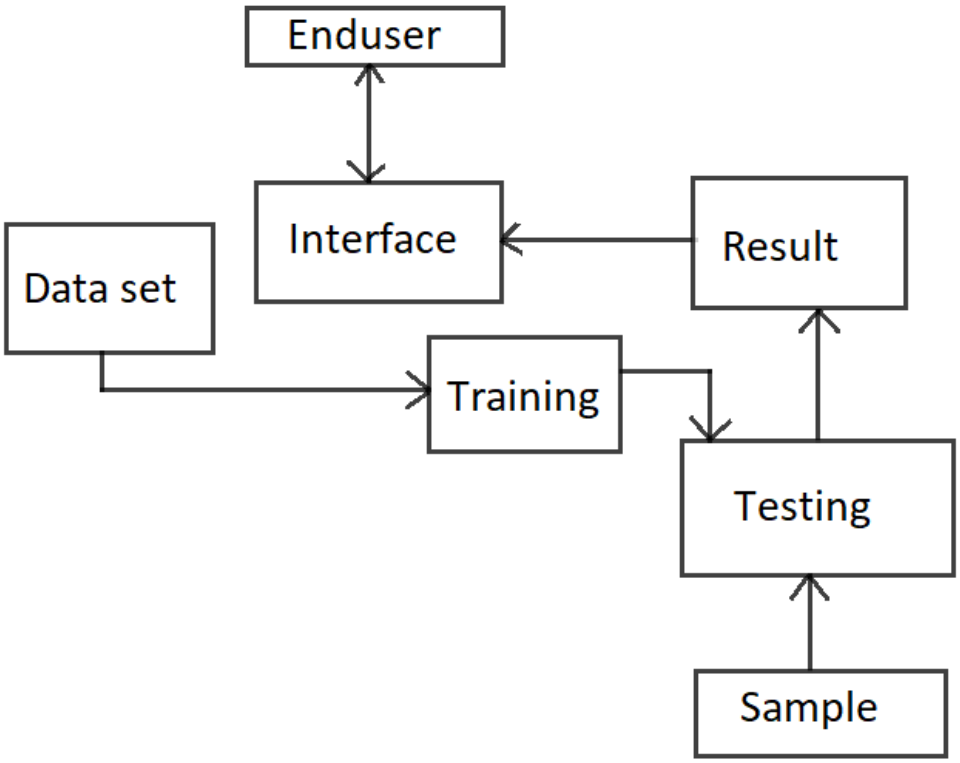


Fig: 4 – Block diagram of this project

CHAPTER 5

PROGRAMMING

CHAPTER 5

PROGRAMMING

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import pickle
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier


data = pd.read_csv("dataset.csv")

data = np.array(data)
print(data)
X = data[:, 0:-1]
y = data[:, -1]
y = y.astype('int')
X = X.astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

#  print(X_train,y_train)
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
# y_pred = rfc.predict(X_test)

# print(metrics.accuracy_score(y_test, y_pred))
pickle.dump(rfc,open('model.pkl','wb'))
```

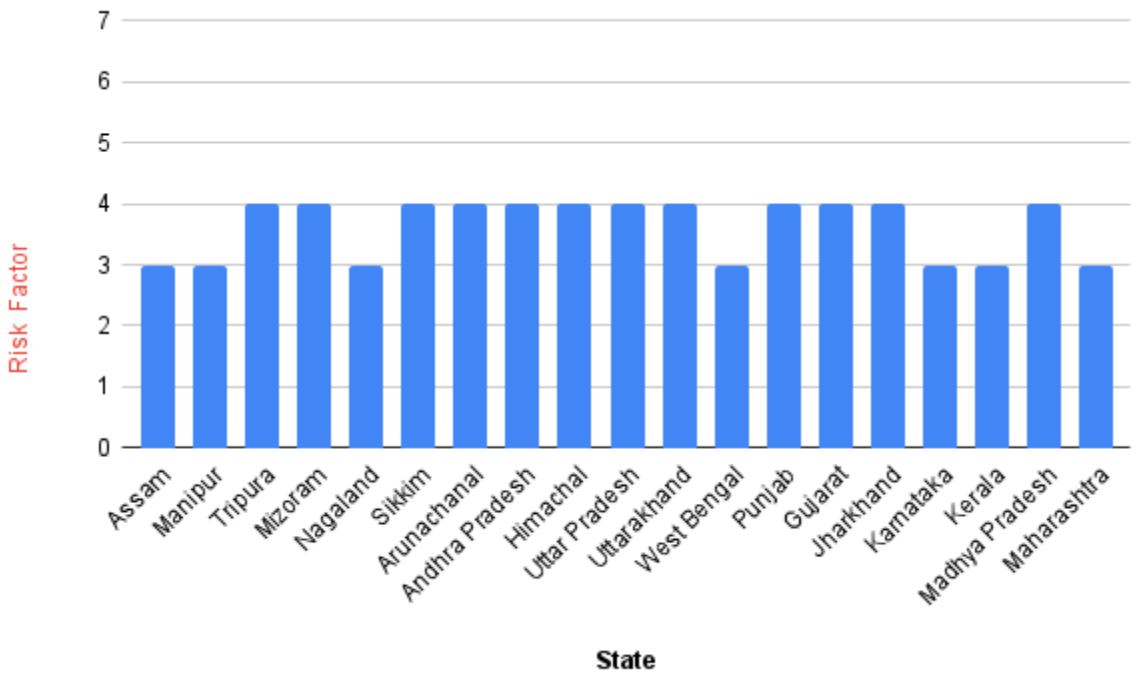
Using Flask

```
from flask import Flask,request, url_for, redirect, render_template
import pickle
import numpy as np

app = Flask(__name__)

model=pickle.load(open('model.pkl','rb'))
```

```
@app.route('/')
def home():
    return render_template('homepage.html')


@app.route('/home')
def home2():
    return render_template('homepage.html')



@app.route('/aboutproject')
def aboutproject():
    return render_template('aboutproject.html')



@app.route('/prediction' , methods=['POST','GET'])
def prediction():
    if request.method == "POST":
        data1 = int(float(request.form['Longitude']))
        data2 = int(float(request.form['Latitude']))
        data3 = int(float(request.form['Height']))
        print(data1,data2,data3)
        arr = np.array([[data1, data2, data3]])
        output= model.predict(arr)
    else:
        return render_template('prediction.html')


    def to_str(var):
     return str(list(np.reshape(np.asarray(var), (1, np.size(var)))[0]))[1:-1]


    # return render_template('prediction.html')

    if (output<4):
        return render_template('prediction.html',p=to_str(output), q=' No ')
    elif (output>4 & output<6):
        return render_template('prediction.html',p=to_str(output), q= ' Low ')
    elif (output>6 & output<8):
        return render_template('prediction.html',p=to_str(output), q=' Moderate
')
    elif (output>8 & output<9):
        return render_template('prediction.html',p=to_str(output), q=' High ')
    elif (output>9):
```

```
        return render_template('prediction.html',p=to_str(output), q=' Very High
')

    else :
        return render_template('prediction.html',p=' N.A.', q= ' Undefined ')


if __name__ == "__main__":
    app.run(debug=True)
```

The graph is given below which we find using data by putting longitude, latitude and depth from the sea level:



CHAPTER 6

RESULT

CHAPTER 6

RESULT

In this machine learning project, we built an earthquake prediction system by using random forest classifier. Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision tree at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set:587–588 Random forests generally outperform decision tree, but their accuracy is lower than gradient boosted tree. However, data characteristics can affect their performance.

RANDOM FOREST

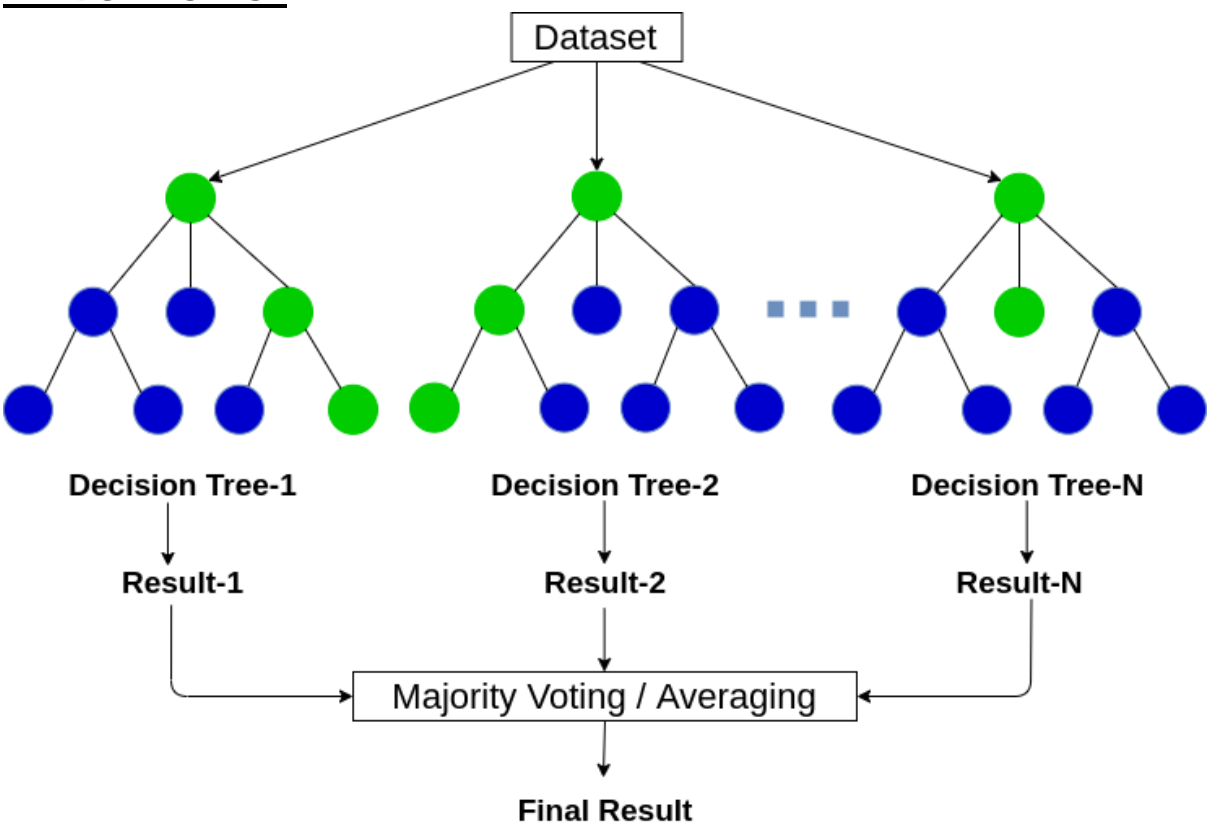


Fig: Random Forest Algorithm Decision Tree

Advantages of random forest

It can perform both regression and classification tasks. A random forest produces good predictions that can be understood easily. It can handle large datasets efficiently. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.

Importance

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Application

Real life applications of Random Forests

- Fraud detection for bank accounts, credit card.
- Detect and predict the drug sensitivity of a medicine.
- Identify a patient's disease by analyzing their medical records.

- Predict estimated loss or profit while purchasing a particular stock.

We have also took the help of some existing methods :

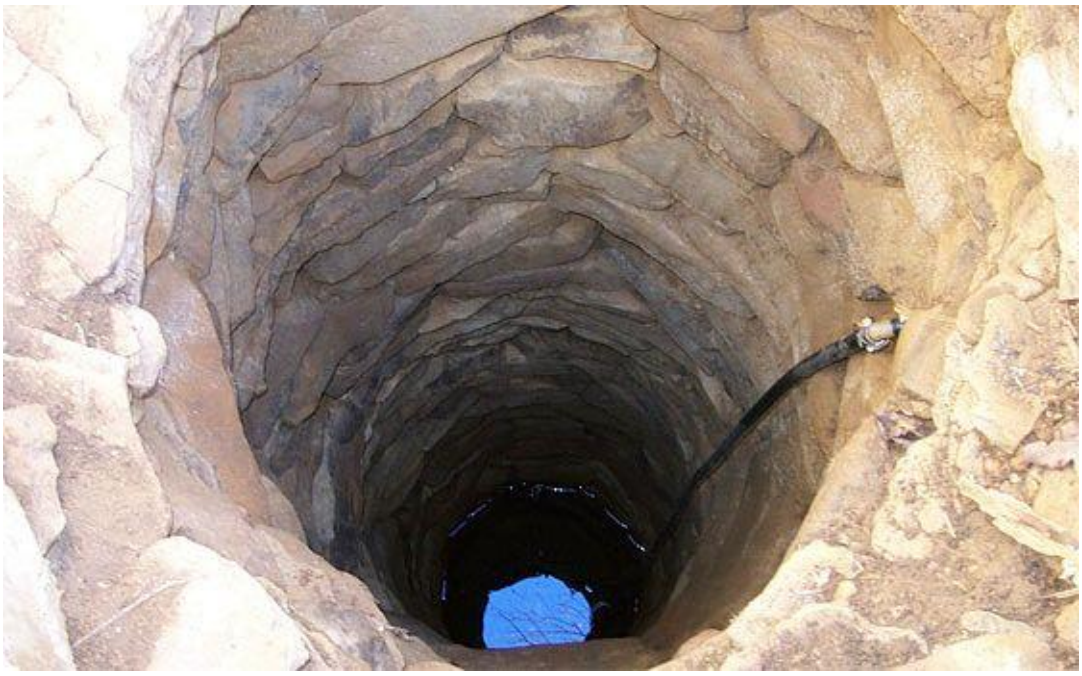
- Prediction based on animal behavior.
- Change in the Water level in wells
- SES
- Radon Emission.
- Foreshocks or minor shocks before major earthquake • Statistical Probability.

Prediction based on animal behaviour.

Rats, snakes, and centipedes reportedly left their homes and headed for safety several days before a destructive earthquake. Anecdotal evidence abounds of animals, fish, birds, reptiles, and insects exhibiting strange behaviour anywhere from weeks to seconds before an earthquake. Owners reported increased neediness, barking, and howling in their dogs, and some dogs who were so restless they even escaped.

Change in the Water level in wells

• We can drill wells in certain locations to measure dramatic changes in water levels.
• Large surface waves force particles of rock near to the surface which changes water levels in a wells.
• Water levels can be affected by any fault creeps, crust tilts, or other seismic activity.
• Unusual decrease of water level were consistently observed in 78% of wells in the central Taiwan about 250 days before the earthquake.



SES(Seismic Electrical Signals)

• Can predict earthquake of magnitude >5 within 100km of the epicentral location and in a 2 hour to 11 day time window.

• Measures the geoelectric potential differences between electrodes stool into the ground at desired distances.
• Distinguish meaningful pre-seismic signals, if they exist at all, from these noise.
Radon Emission
• Radon is continuously produced in the rocks by the decay of radioactive elements.
• lithe rock is suddenly fractured due to build up of strain then the accumulated radon will be released and dissolved in the ground water.
• Thus the sudden increase in radon abundances in streams, well and ground water might be useful method for predicting of earthquake.

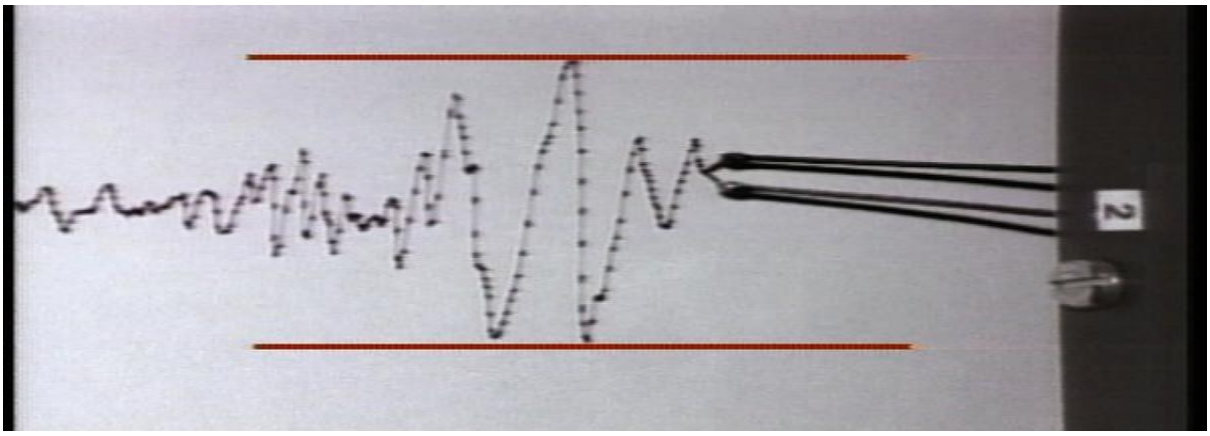


Fig: Seismic Electrical Signal

Foreshocks or minor shocks before major earthquake
• Foreshocks are the best known predictor.
• Foreshocks are observed sometimes before the main shock.
• A Foreshock are smaller in magnitude
• This can be used as the warning system to ensure safety

Statistical Probability

- Statistical probability involves the laws governing random events and their data collection, analysis, interpretation, and display. In other words, we can say that in a classical probability, the possible outcomes are at equal odds.
- This method is using in the project.

For this project we have build one website using html, CSS and JS. It takes three values latitude, longitude and height from sea level.

- **Latitude:**
  Latitude is a coordinate that specifies the north south position of a point on the surface of the earth or another celestial body. Latitude is given as an angle that ranges from -90 degree at the south pole to 90 degree at the north pole, with 0 degree at the equator.

- **Longitude:**
  Longitude measures distance east or west of the prime meridian. Lines of longitude are called meridians, are imaginary lines that divide the earth. They run north to south from pole to pole, but they measures the distance east or west. Longitude is measured in degrees, minutes and seconds.

- **Height:**

  Height above mean sea level is a measure of the vertical distance (height, elevation or altitude) of a location in reference to a historic mean sea level taken as a vertical datum. In geodesy, it is formalized as orthometric heights.

whole project is basically for India, it is being design the data of India only. For now we are using more then 2500 Indian location and their past earthquake magnitudes and in the website the Indian coordinates. The concept we used is web development(Frontend and Backend), Machine learning (linear regression model) and language used is python. The model is numpy, Flask and Pandas.

Machine Learning:

Machine learning is a branch of artificial intelligence (Al) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. IBM has a rich history with machine learning.
There are primarily three types of machine learning: Supervised, Unsupervised, and Reinforcement Learning.

Simply put, machine learning allows the user to feed a computer algorithm an immense amount of data and have the computer analyze and make data-driven recommendations and decisions based on only the input data.
How Machine Learning Works. Machine learning uses two types of techniques: supervised learning, which trains a model on known input and output data so that it can predict future outputs, and unsupervised learning, which finds hidden patterns or intrinsic structures in input data.

Python:

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation via the off-side rule. Python is dynamically typed and garbage- collected.
Python is often used as a support language for software developers, for build control and management, testing, and in many other ways. SCons for build control. Buildbot and Apache Gump for automated continuous compilation and testing. Roundup or Trac for bug tracking and project management.

NumPy:

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

Flask

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it.

Pandas:

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Tectonic Plate movement:

plate tectonics, theory dealing with the dynamics of Earth's outer shell—the lithosphere—that revolutionized Earth sciences by providing a uniform context for understanding mountain-building processes, volcanoes, and earthquakes as well as the evolution of Earth's surface and reconstructing its past continents and oceans.
The concept of plate tectonics was formulated in the 1960s. According to the theory, Earth has a rigid outer layer, known as the lithosphere, which is typically about 100 km (60 miles) thick and overlies a plastic (moldable, partially molten) layer called the asthenosphere. The lithosphere is broken up into seven very large continental- and ocean-sized plates, six or seven medium-sized regional plates, and several small ones. These plates move relative to each other, typically at rates of 5 to 10 cm (2 to 4 inches) per year, and interact along their boundaries, where they converge, diverge, or slip past one another. Such interactions are thought to be responsible for most of Earth's seismic and volcanic activity, although earthquakes and volcanoes can occur in plate interiors. Plate motions cause mountains to rise where plates push together, or converge, and continents to fracture and oceans to form where plates pull apart, or diverge. The continents are embedded in the plates and drift passively with them, which over millions of years results in significant changes in Earth's geography.

The theory of plate tectonics is based on a broad synthesis of geologic and geophysical data. It is now almost universally accepted, and its adoption represents a true scientific revolution, analogous in its consequences to quantum mechanics in physics or the discovery of the genetic code in biology. Incorporating the much older idea of continental drift, as well as the concept of seafloor spreading, the theory of plate tectonics has provided an overarching framework in which to describe the past geography of continents and oceans, the processes controlling creation and destruction of landforms, and the evolution of Earth's crust, atmosphere, biosphere, hydrosphere, and climates. During the late 20th and early 21st centuries, it became apparent that plate-tectonic processes profoundly influence the composition of Earth's atmosphere and oceans, serve as a prime cause of long-term climate change, and make significant contributions to the chemical and physical environment in which life evolves.

Principles of plate tectonic:

In essence, plate-tectonic theory is elegantly simple. Earth's surface layer, 50 to 100 km (30 to 60 miles) thick, is rigid and is composed of a set of large and small plates. Together, these plates constitute the lithosphere, from the Greek lithos, meaning "rock." The lithosphere rests on and slides over an underlying partially molten (and thus weaker but generally denser) layer of plastic partially molten rock known as the asthenosphere, from the Greek asthenos, meaning "weak." Plate movement is possible because the lithosphere-asthenosphere boundary is a zone of detachment. As the lithospheric plates move across Earth's surface, driven by forces as yet not fully understood, they interact along their boundaries, diverging, converging, or slipping past each other. While the interiors of the plates are presumed to remain essentially undeformed, plate boundaries are the sites of many of the principal processes that shape the terrestrial surface, including earthquakes, volcanism, and orogeny (that is, formation of mountain ranges).

The process of plate tectonics may be driven by convection in Earth's mantle, the pull of heavy old pieces of crust into the mantle, or some combination of both. For a deeper discussion of plate-driving mechanisms.

CHAPTER 7

FUTURE SCOPE

Chapter 7

Future scope:

Future scope is making a mobile application which can tell you by accessing your location that if the area is prone to earthquake or not. Also it will notify if the chances of earthquake in that area are very high or there is no probability of earthquake. It will make people more safe if the application sends them notification when probability is that the earthquake will occur. Also it will show the magnitude of how severe the earthquake will be.

The goal of earthquake prediction is to give warning of potentially damaging earthquakes early enough to allow appropriate response to the disaster, enabling people to minimize loss of life and property. The U.S. Geological Survey conducts and supports research on the likelihood of future earthquakes. This research includes field, laboratory, and theoretical investigations of earthquake mechanisms and fault zones. A primary goal of earthquake research is to increase the reliability of earthquake probability estimates. Ultimately, scientists would like to be able to specify a high probability for a specific earthquake on a particular fault within a particular year. Scientists estimate earthquake probabilities in two ways: by studying the history of large earthquakes in a specific area and the rate at which strain accumulates in the rock.

Scientists study the past frequency of large earthquakes in order to determine the future likelihood of similar large shocks. For example, if a region has experienced four magnitude 7 or larger earthquakes during 200 years of recorded history, and if these shocks occurred randomly in time, then scientists would assign a 50 percent probability (that is, just as likely to happen as not to happen) to the occurrence of another magnitude 7 or larger quake in the region during the next 50 years. But in many places, the assumption of random occurrence with time may not be true, because when strain is released along one part of the fault system, it may actually increase on another part. Four magnitude 6.8 or larger earthquakes and many magnitude 6 - 6.5 shocks occurred in the San Francisco Bay region during the 75 years between 1836 and 1911. For the next 68 years (until 1979), no earthquakes of magnitude 6 or larger occurred in the region. Beginning with a magnitude 6.0 shock in 1979, the earthquake activity in the region increased dramatically; between 1979 and 1989, there were four magnitude 6 or greater earthquakes, including the magnitude 7.1 Loma Prieta earthquake. This clustering of earthquakes leads scientists to estimate that the probability of a magnitude 6.8 or larger earthquake occurring during the next 30 years in the San Francisco Bay region is about 67 percent (twice as likely as not). Another way to estimate the likelihood of future earthquakes is to study how fast strain accumulates. When plate movements build the strain in rocks to a critical level, like pulling a rubber band too tight, the rocks will suddenly break and slip to a new position. Scientists measure how much strain accumulates along a fault segment each year, how much time has passed since the last earthquake along the segment, and how much strain was released in the last earthquake. This information is then used to calculate the time required for the accumulating strain to build to the level that results in an earthquake. This simple model is complicated by the fact that such detailed information about faults is rare. In the United States, only the San Andreas fault system has adequate records for using this prediction method. Both of these methods, and a wide array of monitoring techniques, are being tested along part of the San Andres fault. For the past 150 years, earthquakes of about magnitude 6 have occurred an average of every 22 years on the San Andreas fault near Park field, California.

Both of these methods, and a wide array of monitoring techniques, are being tested along part of the San Andres fault. For the past 150 years, earthquakes of about magnitude 6 have occurred an average of every 22 years on the San Andreas fault near Park field, California. The last shock was in 1966. Because of the consistency and similarity of these earthquakes, scientists have started an experiment to "capture" the next Park field earthquake. A dense web of monitoring instruments was deployed in the region during the late 1980s. The main goals of the ongoing Park field Earthquake Prediction Experiment are to record the geophysical signals before and after the expected earthquake; to issue a short-term prediction; and to develop effective methods of communication between earthquake scientists and community officials responsible for disaster response and mitigation. This project has already made important contributions to both earth science and public policy.

Scientific understanding of earthquakes is of vital importance to the Nation. As the population increases, expanding urban development and construction works encroach upon areas susceptible to earthquakes. With a greater understanding of the causes and effects of earthquakes, we may be able to reduce damage and loss of life from this destructive phenomenon.

CHAPTER 8

CONCLUSION

Conclusion:

Predicting earthquakes using machine learning is a complex and challenging task. While there have been some promising results in recent years, it is still an active area of research and there is much work to be done to improve the accuracy of earthquake prediction.

One of the main challenges is the lack of data on historical earthquakes. Machine learning algorithms rely on large amounts of data to identify patterns and make predictions, but earthquake data is relatively sparse, and earthquakes are inherently unpredictable. Additionally, earthquakes are influenced by a wide range of factors, including geological features, tectonic activity, and even weather patterns, which can make it difficult to isolate the specific factors that contribute to earthquake occurrence.

Despite these challenges, there have been some promising results in recent years, particularly in the use of machine learning algorithms to identify patterns in seismic data that may indicate an impending earthquake. Some researchers have also explored the use of machine learning to model earthquake behavior and simulate seismic activity, which could help improve our understanding of earthquake dynamics.

Overall, while there is still much work to be done, the use of machine learning in earthquake prediction shows great potential for improving our ability to forecast seismic activity and mitigate the potentially devastating effects of earthquakes.

CHAPTER 9
REFERENCES

References:

1.	https://data-flair.training/blogs/earthquake-prediction-using-machine-learning/
2.	https://en.wikipedia.org/wiki/Secondary_data
3.	https://en.wikipedia.org/wiki/Primary_data
4.	https://www.w3schools.com/html/html_editors.asp
5.	https://youtu.be/oYRda7UtuhA
6.	https://en.wikipedia.org/wiki/Indian_Plate
7.	https://www.pmfias.com/interaction-of-tectonic-plates-indian-plate/
8.	https://en.wikipedia.org/wiki/List_of_earthquakes_in_India
9.	https://youtu.be/q-ng6YpxHxU
10.	https://www.freemaptools.com/elevation-finder.htm