Practical No.1

Title: Implementation of Logic programming

Aim: Understanding basics of prolog and implementation of prolog to solve tower of hanoi problem, water jug problem, tic-tac-toe problem and 8- Puzzle Problem.

Introduction:

What is Prolog :-

Prolog (programming in logic) is one of the most widely used programming languages in artificial intelligence research. As opposed to imperative languages such as C or Java (the latter of which also happens to be object-oriented) it is a declarative programming language. That means, when implementing the solution to a problem, instead of specifying how to achieve a certain goal in a certain situation, we specify what the situation (rules and facts) and the goal (query) are and let the Prolog interpreter derive the solution for us. Prolog is very useful in some problem areas, such as artificial intelligence, natural language processing, databases, . . . , but pretty useless in others, such as graphics or numerical algorithms.

Example : -

```
male(james1).

male(charles1).

male(charles2).

male(james2).

male(george1).
```

```
 female(catherine).

 female(elizabeth).

 female(sophia).

 parent(charles1, james1).

 parent(elizabeth, james1).

 parent(charles2, charles1).

 parent(catherine, charles1).

 parent(james2, charles1).

 parent(sophia, elizabeth).

 parent(george1, sophia).


mother(M,X):- parent(X,M),female(M),write(M),write(' is Mother of '),write(X),nl.


father(F,X):- parent(X,F),male(F).


sibling(S1,S2):- parent(S1,X), parent(S2,X).


grandfather(G,X) :- parent(Y,G),parent(X,Y).
```

Exercise -

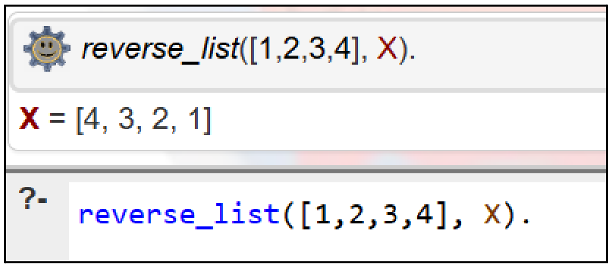
1. **Write a Prolog predicate reverse_list/2 that works like the built-in predicate reverse/2 (without using reverse/2).**

Implementation:

Program:

```
reverse_list([], []).

reverse_list([H|T], R) :-

   reverse_list(T, RT),

   append(RT, [H], R).
```

Output:



```
reverse_list([1,2,3,4], X).
X = [4, 3, 2, 1]
?- reverse_list([1,2,3,4], X).
```

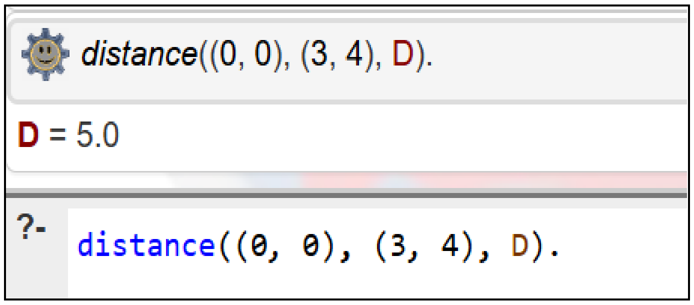
2. **Write a Prolog predicate distance/3 to calculate the distance between two points in the 2-dimensional plane. Points are given as pairs of coordinates.**

Implementation:

Program:

```
distance((X1, Y1), (X2, Y2), D) :-
 DX is X2 - X1,
 DY is Y2 - Y1,
D is sqrt(DX * DX + DY * DY).
```

OUTPUT:



```
distance((0, 0), (3, 4), D).
D = 5.0
?- distance((0, 0), (3, 4), D).
```

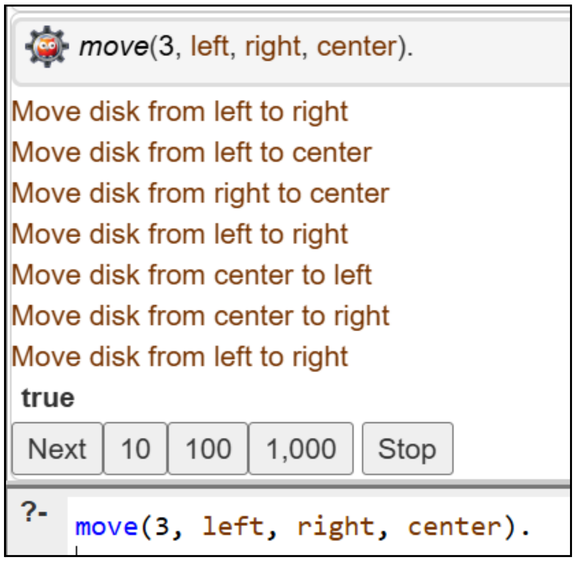
3. **Write a prolog program to solve Tower of hanoi Problem**

Implementation:

Program:

```
        move(1, Source, Destination, _) :-
  write('Move disk from '), write(Source), write(' to '), write(Destination), nl.
move(N, Source, Destination, Temp) :-
  N > 1,
  M is N - 1,
  move(M, Source, Temp, Destination),
  move(1, Source, Destination, _),
  move(M, Temp, Destination, Source).
```

OUTPUT:



```
move(3, left, right, center).
Move disk from left to right
Move disk from left to center
Move disk from right to center
Move disk from left to right
Move disk from center to left
Move disk from center to right
Move disk from left to right
true
```

```
?- move(3, left, right, center).
```

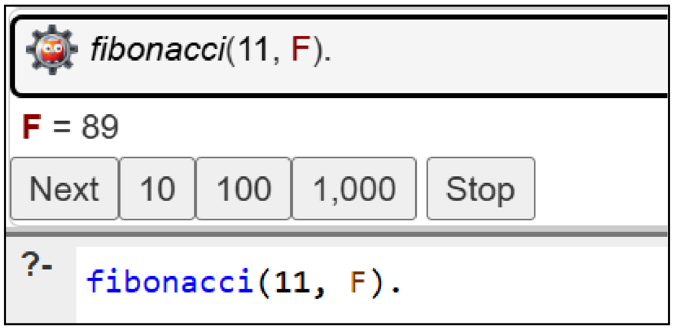
4. Write a Prolog predicate fibonacci/2 to compute the nth Fibonacci number.

Implementation:

Program:

```
fibonacci(0, 0).

fibonacci(1, 1).

fibonacci(N, F) :-

  N > 1,

  N1 is N - 1,

  N2 is N - 2,

  fibonacci(N1, F1),

  fibonacci(N2, F2),

  F is F1 + F2.
```

OUTPUT:



```
fibonacci(11, F).
F = 89
?- fibonacci(11, F).
```

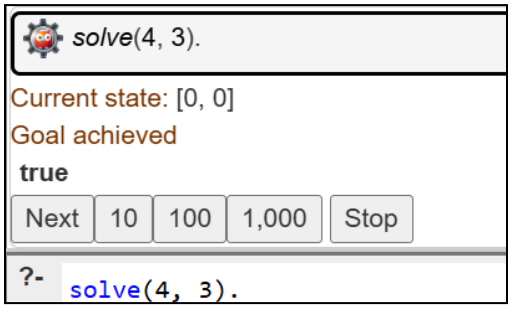
5. Write a prolog program to solve water jug problem.

Implementation:

Program:

```
solve(Jug4, Jug3) :- move([0, 0], Jug4, Jug3, []).

move([4, _], _, _, _) :- write('Goal achieved'), nl.

move([X, Y], Max4, Max3, Visited) :-

   member([X, Y], Visited) -> fail; % avoid loops

   write('Current state: '), write([X, Y]), nl,

   NextVisited = [[X, Y] | Visited],

   (

      move([4, Y], Max4, Max3, NextVisited);

      move([0, Y], Max4, Max3, NextVisited);

      move([X, 3], Max4, Max3, NextVisited);

      move([X, 0], Max4, Max3, NextVisited);

      Transfer is min(X, 3 - Y), NX is X - Transfer, NY is Y + Transfer, move([NX, NY], Max4, Max3,
NextVisited);

      Transfer is min(Y, 4 - X), NX is X + Transfer, NY is Y - Transfer, move([NX, NY], Max4, Max3,
NextVisited)

   ).
```

OUTPUT:



```
solve(4, 3).
Current state: [0, 0]
Goal achieved
true
```

```
?- solve(4, 3).
```

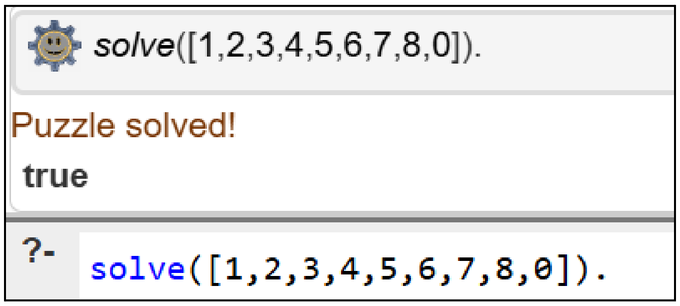
6. **Write a prolog program to solve 8- Puzzle Problem**

Implementation:

Program:

```
        goal([1,2,3,4,5,6,7,8,0]).

move([0,B,C,D,E,F,G,H,I], [B,0,C,D,E,F,G,H,I]).

% Define more moves...

solve(Puzzle) :- goal(Puzzle), write('Puzzle solved!').
```

OUTPUT:



```
solve([1,2,3,4,5,6,7,8,0]).
Puzzle solved!
true
?- solve([1,2,3,4,5,6,7,8,0]).
```

Practical No. 2

Title:Introduction to Python Programming and python libraries

Aim: Understanding basics of python programming and python libraries like numpy, pandas and matplotlib

Introduction:

What is Python :-

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python libraries for Machine Learning

Machine Learning, as the name suggests, is the science of programming a computer by which they are able to learn from different kinds of data. A more general definition given by Arthur Samuel is – "Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed." They are typically used to solve various types of life problems.

In the older days, people used to perform Machine Learning tasks by manually coding all the algorithms and mathematical and statistical formulas. This made the processing time consuming, tedious and inefficient. But in the modern days, it has become very much easy and efficient compared to the olden days with various python libraries, frameworks, and modules. Today, Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reasons is its vast collection of libraries. Python libraries that used in Machine Learning are:

Numpy

NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow use NumPy internally for manipulation of Tensors.

Pandas

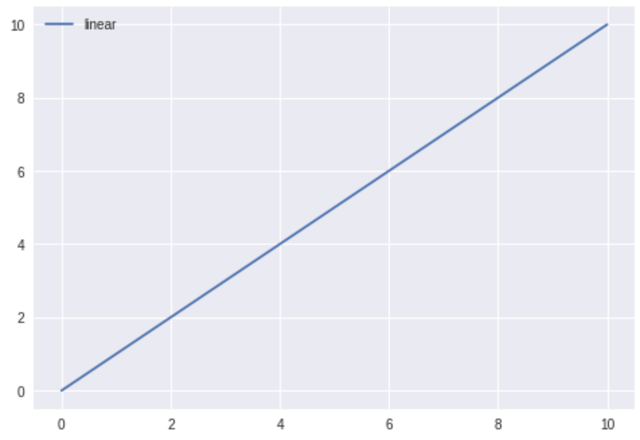
Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and a wide variety of tools for data analysis. It provides many inbuilt methods for grouping, combining and filtering data.

Matplotlib

Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar charts, etc,

Example : -

```
import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0, 10, 100)

plt.plot(x, x, label ='linear')

plt.legend()

plt.show()
```



Exercise -

1. **Create Pandas Dataframe to enter employee database which includes Sr No, Name, Mobile No, City**
2. **Use Iris Dataset from Github and perform all basic operations on Iris Dataset**
3. **Use matplotlib to plot bar chart using dictionary**
4. **Use matplotlib to scatter graph and histogram**

Implementation:
Program:

```
#Understanding basics of python programming
a1=True
type(a1)
```

```
bool
```

```
print("Hello this is first python prgm")
```

```
Hello this is first python prgm
```

```
student="Amey"
student
```

```
'Amey'
```

```
a1=10
a2=20
a1*a2
```

```
200
```

```
str3='''
amey is
MCA
student
from batch 2025
'''
str3
```

```
'\namey is\nMCA\nstudent\nfrom batch 2025\n'
```

```
student="Akshay"
len(student)
```

```
6
```

```
student[3:]
```

```
'hay'
```

```
student="Akshay"
student.replace('A','m')
```

```
'mkshay'
```

```
student="Akshay Akshay Akshay Akshay"
student.count('Akshay')
```

```
4
```

```
#Python Data Structures Tupple
tup1=(1,"b",True,2.2)
tup1
```

```
(1, 'b', True, 2.2)
```

```
tup1[:3]
```

```
(1, 'b', True)
```

```
len(tup1)
```

```
4
```

```
#concat operation
tup2=(3,"c",False,4)
tup1+tup2
```

```
(1, 'b', True, 2.2, 3, 'c', False, 4)
```

```
tup1*2+tup2
```

```
(1, 'b', True, 2.2, 1, 'b', True, 2.2, 3, 'c', False, 4)
```

```
tup2=(3,"c",False,4,3)
tup2
```

```
(3, 'c', False, 4, 3)
```

```
#Python Data Structures : List
list1=[1,2,True,2.2]
list1
```

```
[1, 2, True, 2.2]
```

```
list1[0]=2
list1
```

```
[2, 2, True, 2.2]
```

```
list1[:3]
```

```
[2, 2, True]
```

```
list1.append("amey")
list1
```

```
[2, 2, True, 2.2, 'amey']
```

```
list1.pop()
list1
```

```
[2, 2, True, 2.2]
```

```
list1.insert(3,"three")
list1
```

```
[2, 2, True, 'three', 2.2]
```

```
def msg():
  print("this is function")
```

```
msg()
```

```
this is function
```

```
def oddeven(X):
  if X%2==0:
    print("this is even number")
  else:
    print("this is odd number")
```

```
oddeven(100)
```

```
this is even number
```

```
g=lambda x:x*x*x
```

```
g(2)
```

```
8
```

```
#numpy library
#numpy array
```

```
import numpy as np
n1=np.array([1,2,3])
n1
```

```
array([1, 2, 3])
```

```
n2=np.array([[1,2,3],[4,5,6],[6,7,8]])
n2
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [6, 7, 8]])
```

```
n2.shape
```

```
(3, 3)
```

```
n3=np.zeros((5,5))
n3
```

```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

```
n3=np.full((4,4),4)
n3
```

```
array([[4, 4, 4, 4],
       [4, 4, 4, 4],
       [4, 4, 4, 4],
       [4, 4, 4, 4]])
```

```
n3=np.arange(10,200)
n3
```

```
array([ 10,  11,  12,  13,  14,  15,  16,  17,  18,  19,  20,  21,  22,
        23,  24,  25,  26,  27,  28,  29,  30,  31,  32,  33,  34,  35,
        36,  37,  38,  39,  40,  41,  42,  43,  44,  45,  46,  47,  48,
        49,  50,  51,  52,  53,  54,  55,  56,  57,  58,  59,  60,  61,
        62,  63,  64,  65,  66,  67,  68,  69,  70,  71,  72,  73,  74,
        75,  76,  77,  78,  79,  80,  81,  82,  83,  84,  85,  86,  87,
        88,  89,  90,  91,  92,  93,  94,  95,  96,  97,  98,  99, 100,
       101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113,
       114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126,
       127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139,
       140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152,
       153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165,
       166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178,
       179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191,
       192, 193, 194, 195, 196, 197, 198, 199])
```

```
n4=np.array([[1,2,3],[4,5,6]])
n4
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
n4.shape=(3,2)
n4
```

```
array([[1, 2],
       [3, 4],
         [5, 6]])
```

```
n1=np.array([1,2,3])
n2=np.array([4,5,6])
np.vstack((n1,n2))
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
np.hstack((n1,n2))
```

```
array([1, 2, 3, 4, 5, 6])
```

```
np.column_stack((n1,n2))
```

```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

```
n1=np.array([1,2,3,4,5])
n2=np.array([4,5,6,7,8])
np.intersect1d(n1,n2)
```

```
array([4, 5])
```

```
np.setdiff1d(n2,n1)
```

```
array([6, 7, 8])
```

```
import numpy as np

n1=np.array([1,2,3])
n2=np.array([4,5,4])
np.sum([n1,n2])
```

```
np.int64(19)
```

```
n1=np.array([1,2,3])
np.save('myarr',n1)
```

```
n2=np.load('myarr.npy')
n2
```

```
array([1, 2, 3])
```

```
#Python Pandas lib #series Object

import pandas as pd
s1=pd.Series([1,2,3,4])
s1
```

	0
0	1
1	2
2	3
3	4

dtype: int64

```
s1=pd.Series([1,2,3,4],index=['a','b','c','d'])
s1
```

	0
a	1
b	2
c	3
d	4

dtype: int64

```
s2=pd.Series({'a':1,'b':2,'c':3})
s2
```

	0
a	1
b	2
c	3

dtype: int64

```
#Pandas DataFrame
pd.DataFrame({'Name':['Amey','Akshay','Anil'],
              'Marks':[80,90,95],
              'MobileNo':[7972221200,7972221201,7972221202]})
```

	Name	Marks	MobileNo
0	Amey	80	7972221200
1	Akshay	90	7972221201
2	Anil	95	7972221202

```
url='https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/0e7a9b0a5d22642a06d3d5b9
iris=pd.read_csv(url)
```

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
iris.tail()
```

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

```
iris.shape
```

```
(150, 5)
```

```
iris.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
iris.loc[75:77,"petal_length"]
```

	petal_length
75	4.4
76	4.8
77	5.0

dtype: float64

```
iris.drop([0,1,2],axis=0)
```

	sepal_length	sepal_width	petal_length	petal_width	species
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

147 rows × 5 columns

```
iris.head()
```

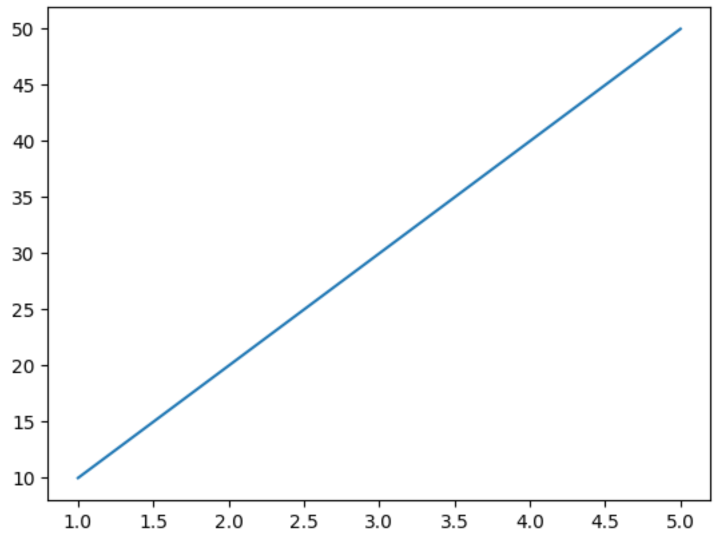
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
iris.sort_values(by='sepal_length')
```

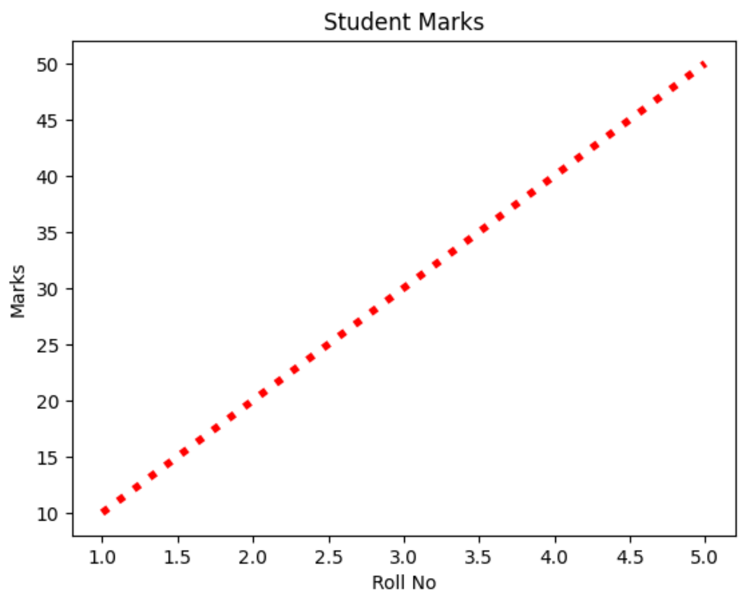
	sepal_length	sepal_width	petal_length	petal_width	species
13	4.3	3.0	1.1	0.1	setosa
8	4.4	2.9	1.4	0.2	setosa
42	4.4	3.2	1.3	0.2	setosa
38	4.4	3.0	1.3	0.2	setosa
41	4.5	2.3	1.3	0.3	setosa
...	...	...	...	...	...
122	7.7	2.8	6.7	2.0	virginica
117	7.7	3.8	6.7	2.2	virginica
118	7.7	2.6	6.9	2.3	virginica
135	7.7	3.0	6.1	2.3	virginica
131	7.9	3.8	6.4	2.0	virginica

150 rows × 5 columns

```
#python library matplotlib

import numpy as np
from matplotlib import pyplot as plt
x=np.array([1,2,3,4,5])
y=np.array([10,20,30,40,50])

plt.plot(x,y)
plt.show()
```

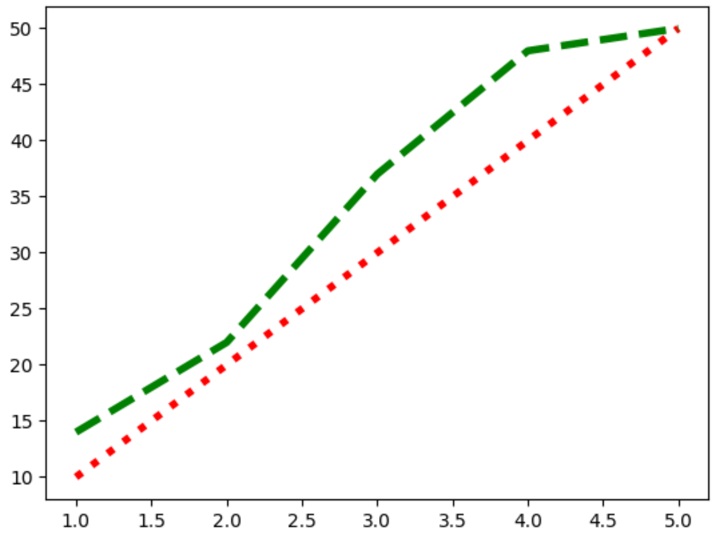


```
plt.plot(x,y,color="r",linestyle=":",linewidth=4)
plt.title("Student Marks")
plt.xlabel("Roll No")
plt.ylabel("Marks")
plt.show()
```

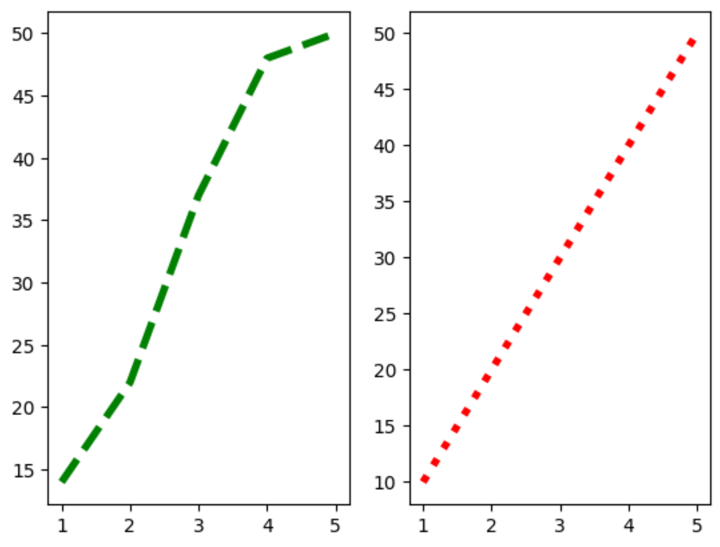


```
y2=np.array([14,22,37,48,50])
plt.plot(x,y,color="r",linestyle=":",linewidth=4)
plt.plot(x,y2,color="g",linestyle="--",linewidth=4)
```

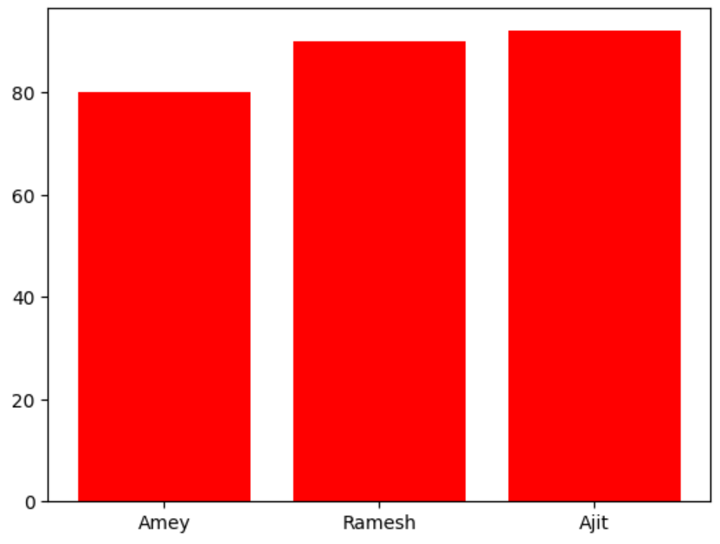
```
[<matplotlib.lines.Line2D at 0x7f7e42ef9d10>]
```



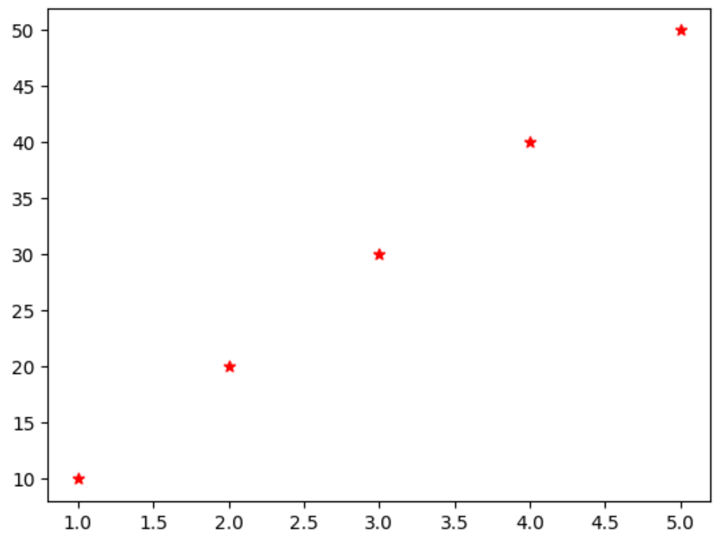
```
plt.subplot(1,2,2)
plt.plot(x,y,color="r",linestyle=":",linewidth=4)

plt.subplot(1,2,1)
plt.plot(x,y2,color="g",linestyle="--",linewidth=4)
plt.show()
```



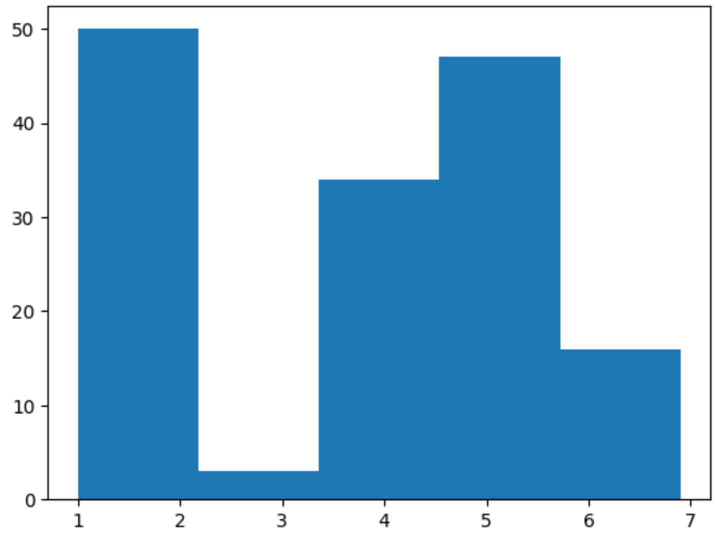
```
student={"Amey":80,"Ramesh":90,"Ajit":92}
name=list(student.keys())
marks=list(student.values())

plt.bar(name,marks,color="r")
plt.show()
```



```
plt.scatter(x,y,color="r",marker="*")
plt.show()
```



```
x=[1,2,3,4,5]
y=[4,5,6,7,8]
z=[7,8,9,10,11]

plt.hist(iris['petal_length'],bins=5)
plt.show()
```



```
x=[1,2,3,4,5]
```

Practical No. 3

Title: Implementation of Artificial Neural Network algorithms

Aim: To implement and evaluate Artificial Neural Network (ANN) algorithms for solving classification and regression problems using Python

Introduction:

Artificial Neural Networks (ANNs) are computational models inspired by the structure and functioning of the human brain. They consist of interconnected processing elements called neurons that work collectively to solve specific problems. ANNs are widely used in various applications such as classification, regression, pattern recognition, and signal processing due to their ability to learn from data and generalize well to unseen inputs.

Two foundational algorithms in the field of neural networks are the Perceptron and ADALINE (Adaptive Linear Neuron). These models are considered the building blocks of more complex neural networks and are essential for understanding the fundamentals of supervised learning.

- Perceptron Algorithm:
  The Perceptron is one of the earliest neural network models, introduced by Frank Rosenblatt in 1958. It is a binary classifier that updates its weights based on the prediction error using a simple update rule. It works well for linearly separable datasets and uses a step activation function.

- ADALINE Algorithm:
  ADALINE, developed by Bernard Widrow and Ted Hoff, is similar to the perceptron but differs in the way it updates weights. Instead of using a binary output, ADALINE uses a linear activation function and minimizes the mean squared error (MSE) between predicted and actual outputs. This allows smoother and more stable convergence during training.

Both algorithms utilize the Gradient Descent optimization technique to minimize the error. Gradient Descent updates the weights iteratively in the direction of the negative gradient of the loss function to reach the optimal solution.

In this practical, we implement the Perceptron and ADALINE algorithms and explore how they learn from training data through weight updates using Gradient Descent. This provides a foundational understanding of how learning occurs in neural networks.

Exercise -

1. Implement Perceptron algorithm for neural networks
2. Implement Perceptron algorithm for OR operation
3. Implement Perceptron algorithm for AND operation

Implementation:

Program: Implement Perceptron algorithm for neural networks

```
#implementing-the-perceptron-neural-network-with-python

x_input=[0.1,0.5,0.2]

w_weights = [0.4,0.3,0.6]

threshold = 0.5


def step(weighted_sum):

 if weighted_sum>threshold:

  return 1

 else:

  return 0
```

```
def perceptron():
 weighted_sum=0


 for x,w in zip(x_input,w_weights):
  weighted_sum += x*w
  print(weighted_sum)
 return step(weighted_sum)


output=perceptron()
print("output" + str(output))
```

Output:

```
0.04000000000000001
0.19
0.31
output0
```

Program: Implement Perceptron algorithm for OR operation

```
import numpy as np
class Perceptron:
  def __init__(self, learning_rate=0.01, n_iters=1000):
    self.lr = learning_rate
    self.n_iters = n_iters
    self.activation_func = self._unit_step_func
```

```
        self.weights = None
        self.bias = None


    def fit(self, X, y):
        n_samples, n_features = X.shape
        # init parameters
        self.weights = np.zeros(n_features)
        self.bias = 0
        y_ = np.array([1 if i > 0 else 0 for i in y])
        for _ in range(self.n_iters):
            for idx, x_i in enumerate(X):
                linear_output = np.dot(x_i, self.weights) + self.bias
                y_predicted = self.activation_func(linear_output)
                # Perceptron update rule
                update = self.lr * (y_[idx] - y_predicted)
                self.weights += update * x_i
                self.bias += update
    def predict(self, X):
        linear_output = np.dot(X, self.weights) + self.bias
        y_predicted = self.activation_func(linear_output)
        return y_predicted
    def _unit_step_func(self, x):
        return np.where(x >= 0, 1, 0)
# OR gate inputs and outputs
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([0, 1, 1, 1])
# Initialize and train the perceptron
perceptron = Perceptron(learning_rate=0.1, n_iters=10)
perceptron.fit(X, y)
# Test the perceptron
predictions = perceptron.predict(X)
predictions
```

Output:

```
([0, 1, 1, 1])
```

Program: Implement Perceptron algorithm for AND operation

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import StandardScaler

# Define AND operation dataset
```

```
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])  # Input features

y = np.array([0, 0, 0, 1])  # AND operation labels


# Standardize the data (important for Adaline)

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)


# Implement Adaline using scikit-learn's SGDClassifier with linear loss

adaline = SGDClassifier(loss="squared_loss", learning_rate="constant", eta0=0.01,
max_iter=1000)

adaline.fit(X_scaled, y)


# Predict on training data

predictions = adaline.predict(X_scaled)


# Display results

print("Predictions:", predictions)

print("Actual Labels:", y)
```

Practical No. 4

Title: Implementation of Feature Selection and Dimensionality Reduction Techniques

Aim: To understand and implement various techniques of feature extraction, feature selection, normalization, transformation, and principal component analysis (PCA) with visualizations for high-dimensional datasets.

Introduction:

In machine learning and data science, handling high-dimensional data is a critical step. Raw data often contains redundant, irrelevant, or noisy features that can degrade model performance. To build efficient and accurate models, it is important to reduce dimensionality by extracting relevant features, selecting important ones, and transforming them into usable formats. This manual covers essential steps including feature extraction, selection, normalization, transformation, and PCA with visual insights.

1. Feature Extraction

Definition:
Feature extraction involves transforming raw data into numerical features that can be processed while preserving the information in the original data.

Examples:

- Text: TF-IDF, word embeddings
- Image: Edge detection, color histograms
- Time Series: Statistical metrics like mean, std, etc.

2. Feature Selection

Definition:
Feature selection involves selecting the most relevant features from the dataset for model training to improve performance and reduce overfitting.

Techniques:

- Filter Methods (e.g., correlation, chi-square)
- Wrapper Methods (e.g., Recursive Feature Elimination)
- Embedded Methods (e.g., Lasso Regression)

Python Example:

```
from sklearn.feature_selection import SelectKBest, chi2

from sklearn.datasets import load_iris

iris = load_iris()

X, y = iris.data, iris.target

X_new = SelectKBest(score_func=chi2, k=2).fit_transform(X, y)

print(X_new[:5])
```

3. Normalization

Definition:
Normalization scales numerical data to a standard range (usually 0 to 1) to ensure uniformity and prevent bias towards features with large scales.

Python Example:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X_scaled = scaler.fit_transform(X)
```

```
print(X_scaled[:5])
```

4. Transformation

Definition:
Transformation changes the distribution of the data, often to approximate normality or stabilize variance.

Common Transformations:

- Log Transformation
- Box-Cox Transformation
- Power Transformation

Python Example:

```
import numpy as np

from sklearn.preprocessing import PowerTransformer


X_transformed = PowerTransformer().fit_transform(X)

print(X_transformed[:5])
```

5. Principal Component Analysis (PCA)

Definition:
PCA is a dimensionality reduction technique that transforms the data to a new coordinate system, selecting the directions (principal components) that maximize variance.

Python Example with Visualization:

```
from sklearn.decomposition import PCA
```

```
import matplotlib.pyplot as plt


pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)


plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Iris Dataset')
plt.colorbar()
plt.show()
```

Exercise -

1. Implementation of Features Extraction and Selection, Normalization, Transformation, Principal Components Analysis.

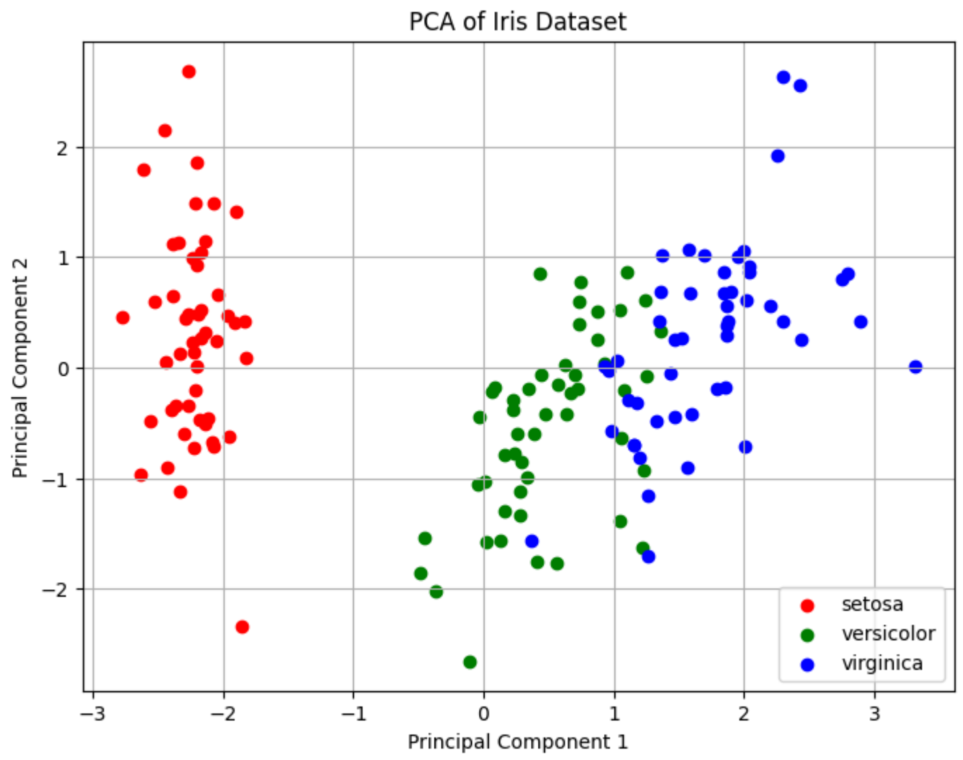
Implementation:

Program:

```
#Feature Selection: Features Extraction, Feature Selection, Normalization, Transformation,
#Principal Components Analysis-visualizations of complex datasets.


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA


# Step 1: Load the dataset

iris = load_iris()

X = iris.data

y = iris.target


# Step 2: Standardize the features

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)


# Step 3: Perform PCA

pca = PCA(n_components=2)

X_pca = pca.fit_transform(X_scaled)


# Step 4: Visualize the results

plt.figure(figsize=(8, 6))

targets = np.unique(y)

colors = ['r', 'g', 'b']
```

```
for target, color in zip(targets, colors):
 plt.scatter(X_pca[y == target, 0],X_pca[y == target, 1],color=color, label=target)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Iris Dataset')
plt.legend(labels=iris.target_names)
plt.grid()
plt.show()
```

Output:



Practical No. 5

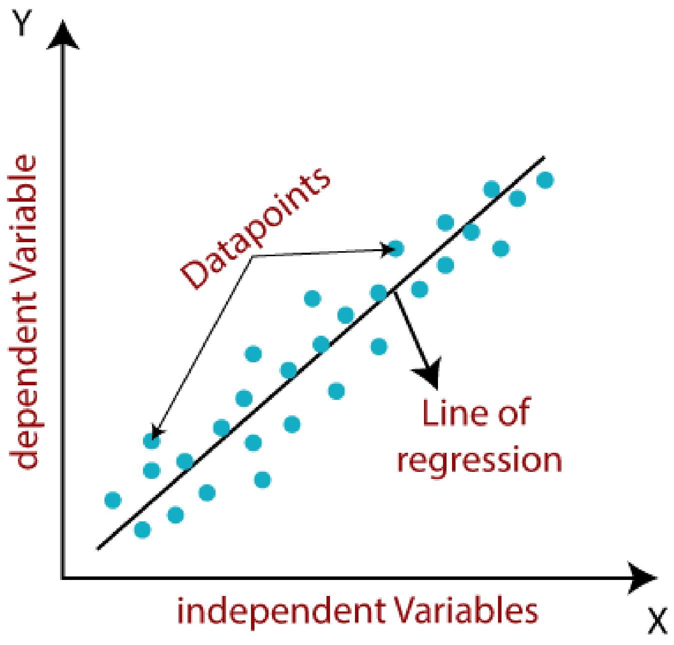
Title: Implementation Supervised Learning algorithms : Linear Regression, Logistic regression, Support Vector Machine or SVM

Aim: Understanding basics of Linear Regression, Logistic regression and Support Vector Machine or SVM

Introduction:

Linear Regression

Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.



Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables.

Mathematically, we can represent a linear regression as:

y= mx + b

Here,

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

b= intercept of the line (Gives an additional degree of freedom)

m= Linear regression coefficient (scale factor to each input value).

The values for x and y variables are training datasets for Linear Regression model representation.

Finding the best fit line:

When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

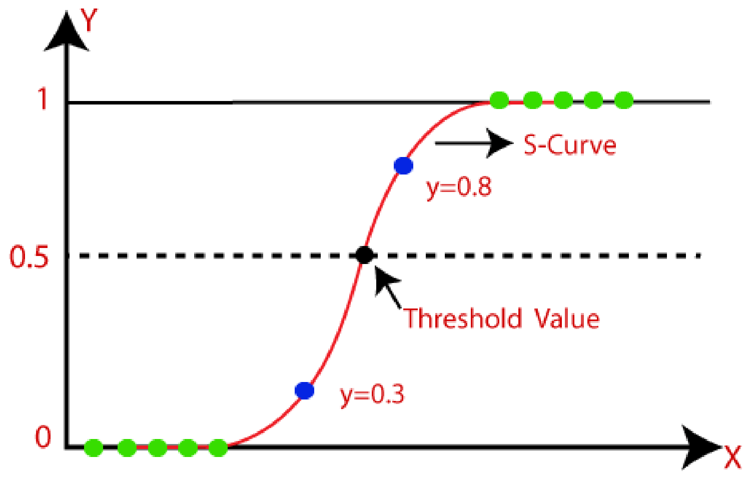
The different values for weights or the coefficient of lines (a_0, a_1) gives a different line of regression, so we need to calculate the best values for a_0 and a_1 to find the best fit line, so to calculate this we use cost function.

Logistic Regression in Machine Learning

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or

No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.

- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:

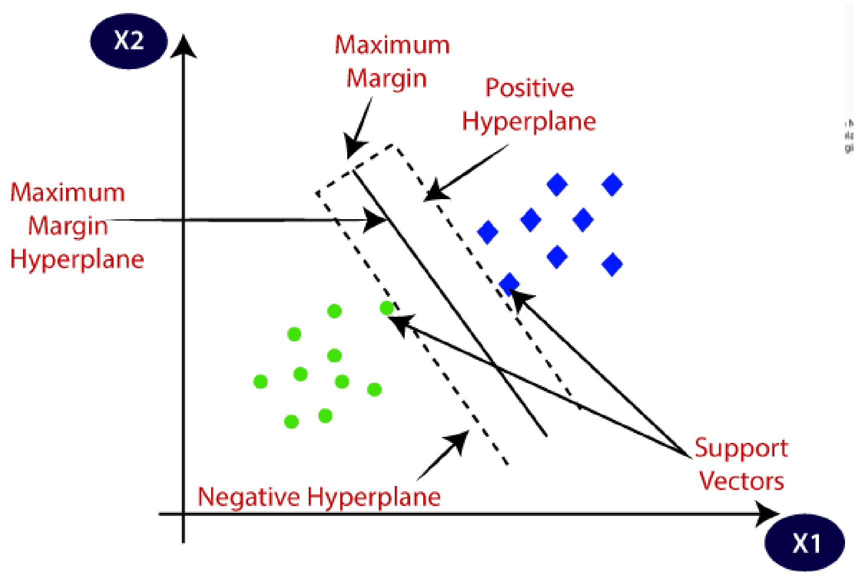


Support Vector Machine or SVM

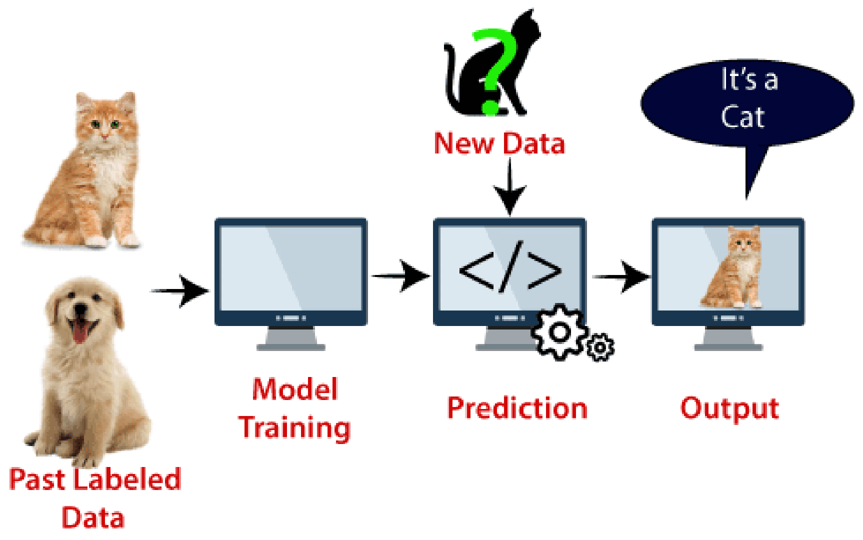
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

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 support vectors, and hence the 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:



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 the support vector creates a decision boundary between these two data (cat and dog) and chooses 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:



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.

Exercise -

1. Implementation of Linear regression
2. Implementation of Logistic regression
3. Support Vector Machine Tutorial Using Python Sklearn

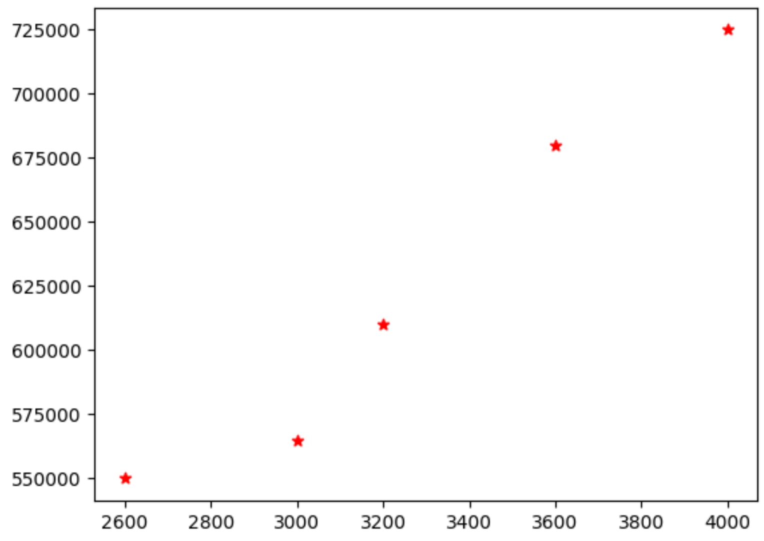
Implementation:

Program:

```
#Linear Regression : Supervised Machine Learning
import numpy as np
import pandas as pd
from sklearn import linear_model
from matplotlib import pyplot as plt
url="https://raw.githubusercontent.com/codebasics/py/master/ML/1_linear_reg/homeprices.csv"
hp=pd.read_csv(url)
hp
```

	area	price
0	2600	550000
1	3000	565000
2	3200	610000
3	3600	680000
4	4000	725000

```
plt.scatter(hp.area,hp.price,color="r",marker="*")
plt.show()
```



```
rg=linear_model.LinearRegression()
rg.fit(hp[['area']],hp.price)
```

```
▾ LinearRegression
LinearRegression()
```

```
rg.predict([[3300]])
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Lin
  warnings.warn(
array([628715.75342466])
```

```
rg.predict([[3900]])
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Lin
  warnings.warn(
array([710188.35616438])
```

```
rg.coef_
```

```
array([135.78767123])
```

```
rg.intercept_
```

```
np.float64(180616.43835616432)
```

```
135.78767123*3300+180616.43835616432
```

```
628715.7534151643
```

```
#Linear Regression : Supervised Machine Learning
url="https://raw.githubusercontent.com/codebasics/py/master/ML/1_linear_reg/areas.csv"
ar1=pd.read_csv(url)
ar1
```

	area
0	1000
1	1500
2	2300
3	3540
4	4120
5	4560
6	5490
7	3460
8	4750
9	2300
10	9000
11	8600
12	7100

```
p=rg.predict(ar1)
ar1['prices']=p
ar1
```

	area	prices
0	1000	3.164041e+05
1	1500	3.842979e+05
2	2300	4.929281e+05
3	3540	6.613048e+05
4	4120	7.400616e+05
5	4560	7.998082e+05
6	5490	9.260908e+05
7	3460	6.504418e+05
8	4750	8.256079e+05
9	2300	4.929281e+05
10	9000	1.402705e+06
11	8600	1.348390e+06
12	7100	1.144709e+06

```
ar1.to_csv('newareas.csv')
```

```
#Linear Regression : Supervised Machine Learning Multivariable
import numpy as np
import pandas as pd
from sklearn import linear_model
from matplotlib import pyplot as plt
url="https://raw.githubusercontent.com/codebasics/py/master/ML/2_linear_reg_multivariate/homepr
homep=pd.read_csv(url)
homep
```

	area	bedrooms	age	price
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	NaN	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000
5	4100	6.0	8	810000

```
homep.bedrooms=homep.bedrooms.fillna(homep.bedrooms.median())
homep
```

	area	bedrooms	age	price
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	4.0	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000
5	4100	6.0	8	810000

```
rg=linear_model.LinearRegression()
rg.fit(homep[['area','bedrooms','age']],homep.price)
```

```
▾ LinearRegression
LinearRegression()
```

```
rg.predict([[3000,3,40]])
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Lin
  warnings.warn(
array([498408.25158031])
```

```
rg.coef_
```

```
array([  112.06244194, 23388.88007794, -3231.71790863])
```

```
rg.intercept_
```

```
np.float64(221323.00186540396)
```

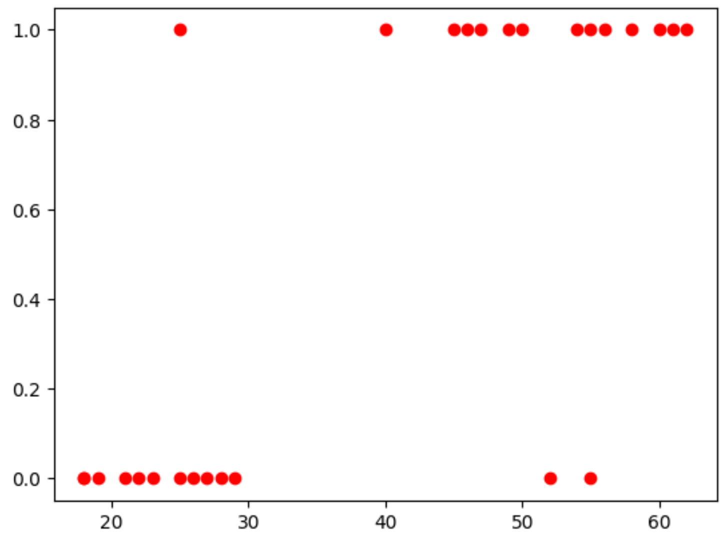
```
112.06244194*3000+23388.88007794*3+-3231.71790863*40+221323.00186540396
```

```
498408.251574024
```

```
#Logistic Regression : Supervised Machine Learning
import numpy as np
import pandas as pd
from sklearn import linear_model
from matplotlib import pyplot as plt
url="https://raw.githubusercontent.com/WamanParulekar/AIML/main/lic.csv
```

	age	lic_member
0	22	0
1	25	0
2	47	1
3	52	0
4	28	0
5	27	0
6	29	0
7	49	1
8	55	1
9	25	1
10	58	1
11	19	0
12	46	1
13	56	1
14	55	0
15	60	1
16	62	1
17	61	1
18	18	0
19	18	0
20	21	0
21	26	0
22	40	1
23	45	1
24	50	1
25	54	1
26	23	0

```
plt.scatter(lic.age,lic.lic_member,color="r")
plt.show()
```



```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(lic[['age']],lic.lic_member,train_size=0.9)
```

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
lr.predict([[54]])
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Log
  warnings.warn(
array([1])
```

```
X_test
```

	age
17	61
25	54
22	40

```
lr.score(X_test,y_test)
```

```
0.6666666666666666
```

```
#Support Vector Machine : Supervised Machine Learning
import numpy as np
import pandas as pd
from sklearn import linear_model
from matplotlib import pyplot as plt
url="https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/0e7a9b0a5d22642a06d3d5b
iris=pd.read_csv(url)
iris
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

```
iris1=iris.drop(['species'],axis='columns')
iris1.head()
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris1,iris[['species']],train_size=0.9)
```

```
X_test.head()
```

	sepal_length	sepal_width	petal_length	petal_width
52	6.9	3.1	4.9	1.5
145	6.7	3.0	5.2	2.3
85	6.0	3.4	4.5	1.6
149	5.9	3.0	5.1	1.8
45	4.8	3.0	1.4	0.3

```
from sklearn.svm import SVC
model=SVC()
model.fit(X_train,y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConv
  y = column_or_1d(y, warn=True)
▾ SVC
SVC()
```

```
model.predict([[5.4,3.7,1.5,0.2]])
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but SVC
  warnings.warn(
array(['setosa'], dtype=object)
```

Practical No. 6

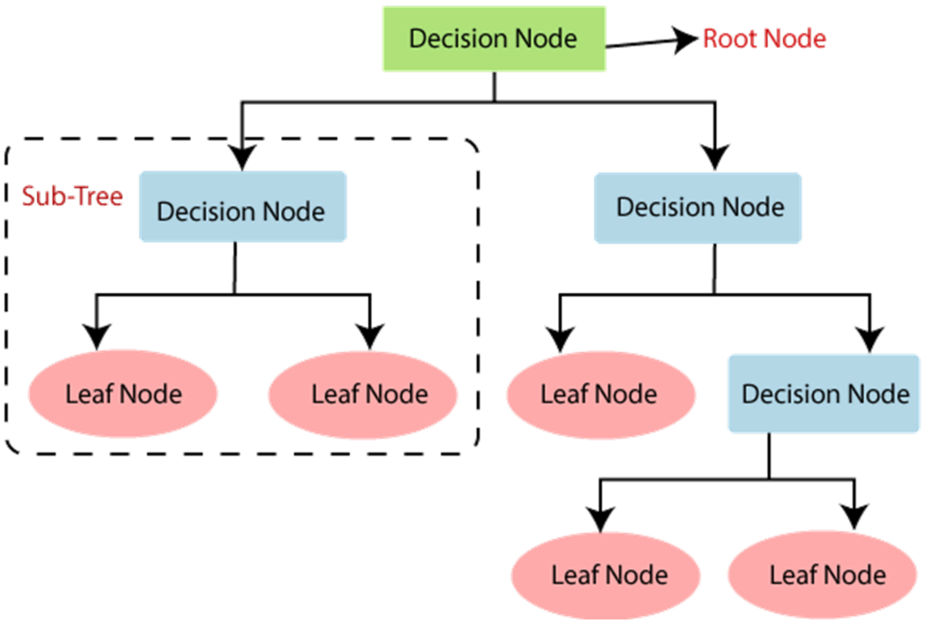
Title: Implementation of Bagging Algorithm: Decision Tree, Random Forest

Aim: Understanding basics of Bagging Algorithm: Decision Tree, Random Forest

Introduction:

Decision Tree

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

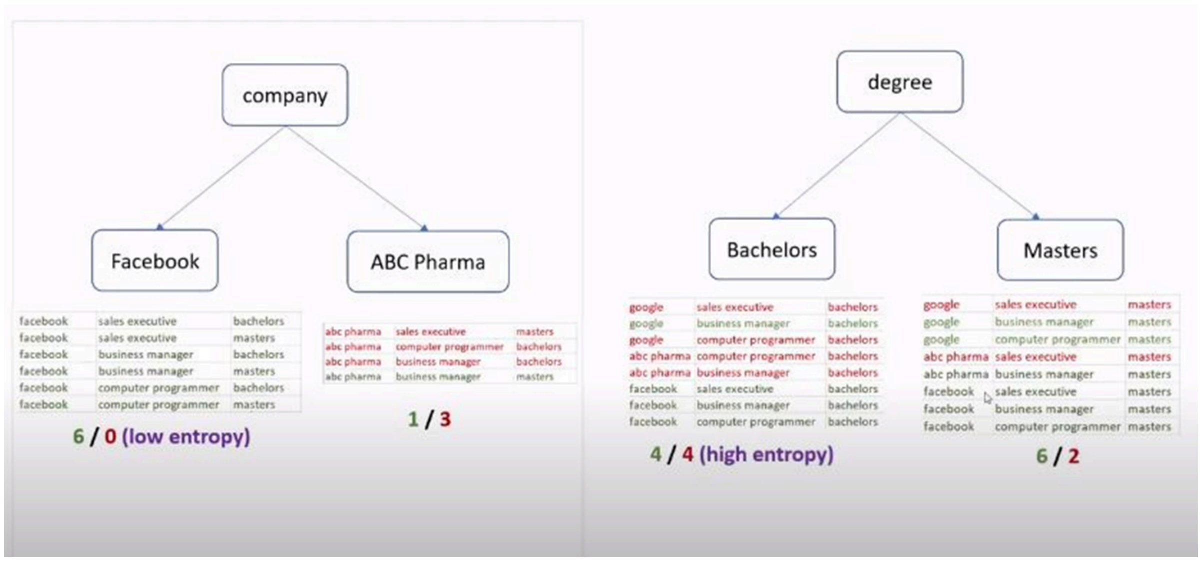


In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
Step-3: Divide the S into subsets that contain possible values for the best attributes.
Step-4: Generate the decision tree node, which contains the best attribute.
Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Attribute Selection Measures
While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**

By this measurement, we can easily select the best attribute for the nodes of the tree.
There are two popular techniques for ASM, which are:
Information Gain
Gini Index

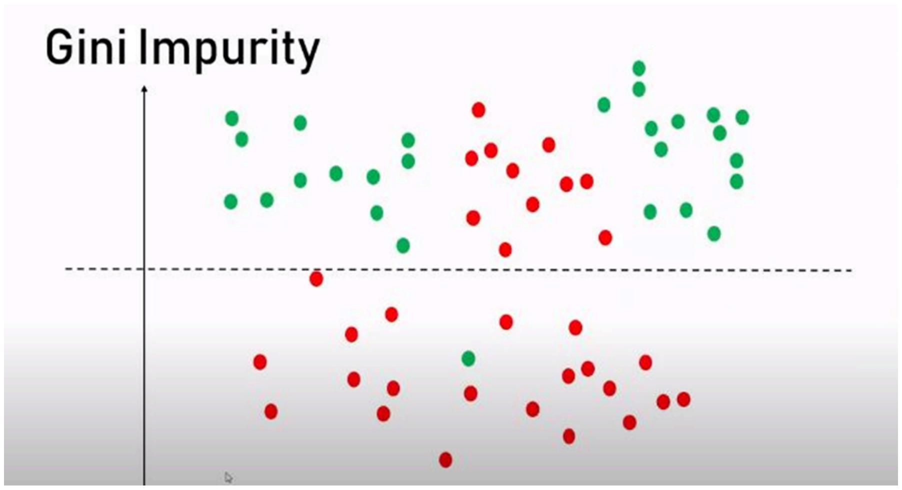


facebook	sales executive	bachelors
facebook	sales executive	masters
facebook	business manager	bachelors
facebook	business manager	masters
facebook	computer programmer	bachelors
facebook	computer programmer	masters

abc pharma	sales executive	masters
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
abc pharma	business manager	masters

google	sales executive	bachelors
google	business manager	bachelors
google	computer programmer	bachelors
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
facebook	sales executive	bachelors
facebook	business manager	bachelors
facebook	computer programmer	bachelors

google	sales executive	masters
google	business manager	masters
google	computer programmer	masters
abc pharma	sales executive	masters
abc pharma	business manager	masters
facebook	sales executive	masters
facebook	business manager	masters
facebook	computer programmer	masters

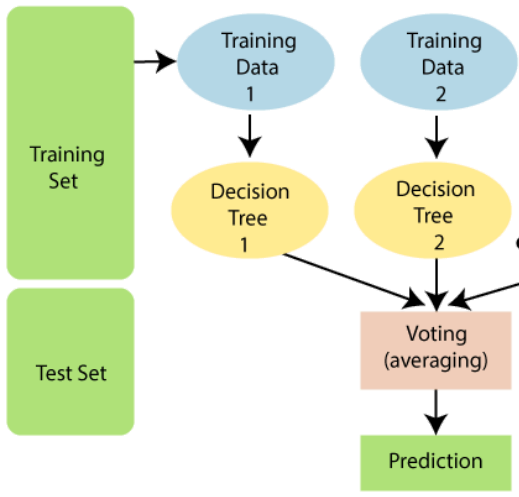


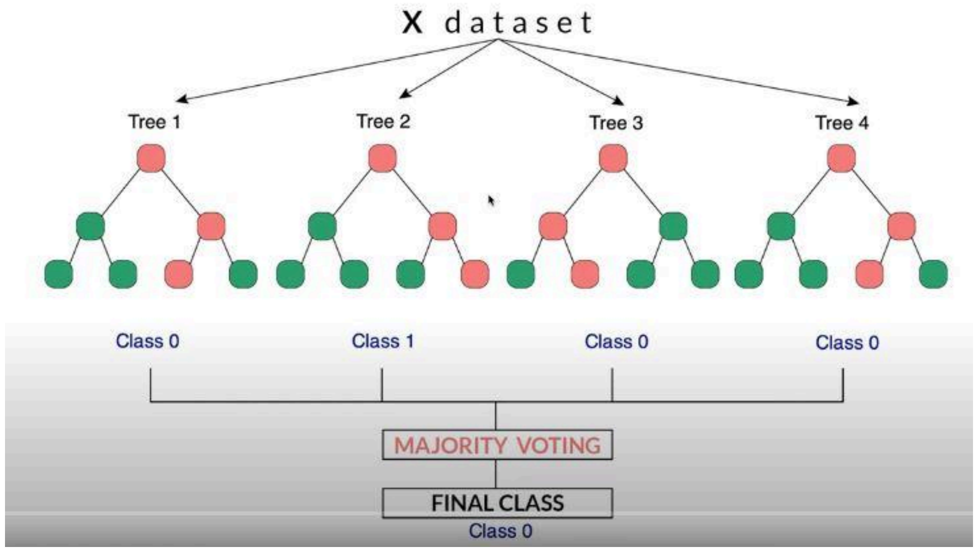
Introduction: Random Forest Algorithm

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:





Exercise -

1. Implement given dataset to predict salary range of computer programmer from google having bachelor degree / master's degree
2. Implement random forest algorithm.

```
Implementation:
Program:

#Bagging Algorithm:Decision Tree
import pandas as pd
url='https://raw.githubusercontent.com/codebasics/py/master/ML/9_decision_tree/salaries.csv'
df = pd.read_csv(url)
df
```

	company	job	degree	salary_more_then_100k
0	google	sales executive	bachelors	0
1	google	sales executive	masters	0
2	google	business manager	bachelors	1
3	google	business manager	masters	1
4	google	computer programmer	bachelors	0
5	google	computer programmer	masters	1
6	abc pharma	sales executive	masters	0
7	abc pharma	computer programmer	bachelors	0
8	abc pharma	business manager	bachelors	0
9	abc pharma	business manager	masters	1
10	facebook	sales executive	bachelors	1
11	facebook	sales executive	masters	1
12	facebook	business manager	bachelors	1
13	facebook	business manager	masters	1
14	facebook	computer programmer	bachelors	1
15	facebook	computer programmer	masters	1

```
inp = df.drop(['salary_more_then_100k'],axis="columns")
inp
```

	company	job	degree
0	google	sales executive	bachelors
1	google	sales executive	masters
2	google	business manager	bachelors
3	google	business manager	masters
4	google	computer programmer	bachelors
5	google	computer programmer	masters
6	abc pharma	sales executive	masters
7	abc pharma	computer programmer	bachelors
8	abc pharma	business manager	bachelors
9	abc pharma	business manager	masters
10	facebook	sales executive	bachelors
11	facebook	sales executive	masters
12	facebook	business manager	bachelors
13	facebook	business manager	masters
14	facebook	computer programmer	bachelors
15	facebook	computer programmer	masters

```
trgt = df['salary_more_then_100k']
trgt
```

```
0     0
1     0
2     1
3     1
4     0
```

```
5     1
6     0
7     0
8     0
9     1
10    1
11    1
12    1
13    1
14    1
15    1
Name: salary_more_then_100k, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
lbl_company = LabelEncoder()
lbl_job = LabelEncoder()
lbl_degree = LabelEncoder()

inp['company_new'] = lbl_company.fit_transform(inp['company'])
inp['job_new'] = lbl_company.fit_transform(inp['job'])
inp['degree_new'] = lbl_company.fit_transform(inp['degree'])
inp
```

	company	job	degree	company_new	job_new	degree_new
0	google	sales executive	bachelors	2	2	0
1	google	sales executive	masters	2	2	1
2	google	business manager	bachelors	2	0	0
3	google	business manager	masters	2	0	1
4	google	computer programmer	bachelors	2	1	0
5	google	computer programmer	masters	2	1	1
6	abc pharma	sales executive	masters	0	2	1
7	abc pharma	computer programmer	bachelors	0	1	0
8	abc pharma	business manager	bachelors	0	0	0
9	abc pharma	business manager	masters	0	0	1
10	facebook	sales executive	bachelors	1	2	0
11	facebook	sales executive	masters	1	2	1
12	facebook	business manager	bachelors	1	0	0
13	facebook	business manager	masters	1	0	1
14	facebook	computer programmer	bachelors	1	1	0
15	facebook	computer programmer	masters	1	1	1

```
inp_new = inp.drop(['company','job','degree'],axis='columns')
inp_new
```

	company_new	job_new	degree_new
0	2	2	0
1	2	2	1
2	2	0	0
3	2	0	1
4	2	1	0
5	2	1	1
6	0	2	1
7	0	1	0
8	0	0	0
9	0	0	1
10	1	2	0
11	1	2	1
12	1	0	0
13	1	0	1
14	1	1	0
15	1	1	1

```
from sklearn import tree
tree_model = tree.DecisionTreeClassifier()
tree_model.fit(inp_new,trgt)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
tree_model.predict([[2,2,1]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClas
  warnings.warn(
array([0])
```

```
#Random Forest Algorithm
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

df = pd.read_csv("https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/0e7a9b0a5d2
df
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df.drop(['species'],axis='columns'),df[['specie
from sklearn.ensemble import RandomForestClassifier
RFC=RandomForestClassifier()
RFC.fit(X_train,y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/base.py:1389: DataConversionWarning:
  return fit_method(estimator, *args, **kwargs)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
RFC.predict([[4.6,3.1,1.5,0.2]])
```

Practical No. 7

Title: Implementation of Boosting Algorithms

Aim: Understanding basics of Boosting Algorithms

Introduction:

What is Boosting?
Boosting is an ensemble learning technique that builds a strong learner by combining multiple weak learners (usually decision trees with shallow depth). Instead of training models independently (like in bagging), boosting trains them sequentially, where each new model tries to correct the mistakes made by the previous ones.

Key Ideas Behind Boosting:

1. Sequential Learning: Models are trained one after the other.
2. Focus on Errors: Each model tries to fix the errors of its predecessor.
3. Weighted Voting: Final prediction is a weighted vote (or sum) of all the models.
4. Boost Weak Learners: Even models that are only slightly better than random can be boosted into a strong predictor.

Types of Boosting Algorithms
1. AdaBoost (Adaptive Boosting)

- Starts with equal weights for all training instances.
- After each model, it increases the weights of misclassified points so that the next model focuses more on them.
- Final model is a weighted sum of all weak learners.

Base Learner: Usually decision stumps (1-level trees)
Sequence:

- Fit a model
- Compute error and adjust sample weights
- Fit next model on new weights

---

2. Gradient Boosting

- Works like gradient descent: each new model tries to minimize the error of the overall model using the gradient of a loss function.
- Models are trained on the residuals (errors) of the previous model.

Base Learner: Small decision trees
Sequence:

- Start with an initial prediction (e.g., mean)
- Compute residuals (errors)
- Fit a new model on the residuals
- Add new model to the ensemble

---

3. Stochastic Gradient Boosting

- Same as Gradient Boosting, but adds randomness:
    - Only a subset of data is used to train each new model.
- Helps reduce overfitting and improves generalization.

Exercise -

1. Implement Boosting Algorithms

```
#Boosting Algorithm: AdaBoost
import pandas as pd
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
import warnings
url='https://raw.githubusercontent.com/codebasics/py/master/ML/9_decision_tree/salaries.csv'
df = pd.read_csv(url)
df
```

```
       company                  job     degree  salary_more_then_100k
0       google      sales executive  bachelors                      0
1       google      sales executive    masters                      0
2       google     business manager  bachelors                      1
3       google     business manager    masters                      1
4       google  computer programmer  bachelors                      0
5       google  computer programmer    masters                      1
6   abc pharma      sales executive    masters                      0
7   abc pharma  computer programmer  bachelors                      0
8   abc pharma     business manager  bachelors                      0
9   abc pharma     business manager    masters                      1
10    facebook      sales executive  bachelors                      1
11    facebook      sales executive    masters                      1
12    facebook     business manager  bachelors                      1
13    facebook     business manager    masters                      1
14    facebook  computer programmer  bachelors                      1
15    facebook  computer programmer    masters                      1
```

```
inp = df.drop(['salary_more_then_100k'],axis="columns")
trgt = df['salary_more_then_100k']
trgt
```

```
    salary_more_then_100k
0                       0
1                       0
2                       1
3                       1
4                       0
5                       1
6                       0
7                       0
8                       0
9                       1
10                      1
11                      1
12                      1
13                      1
14                      1
15                      1

dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
lbl_company = LabelEncoder()
lbl_job = LabelEncoder()
lbl_degree = LabelEncoder()

inp['company_new'] = lbl_company.fit_transform(inp['company'])
inp['job_new'] = lbl_company.fit_transform(inp['job'])
inp['degree_new'] = lbl_company.fit_transform(inp['degree'])


inp_new = inp.drop(['company','job','degree'],axis='columns')
inp_new
```

	company_new	job_new	degree_new
0	2	2	0
1	2	2	1
2	2	0	0
3	2	0	1
4	2	1	0
5	2	1	1
6	0	2	1
7	0	1	0
8	0	0	0
9	0	0	1
10	1	2	0
11	1	2	1
12	1	0	0
13	1	0	1
14	1	1	0
15	1	1	1

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inp_new,trgt,train_size=0.8)

model=AdaBoostClassifier(n_estimators=100,learning_rate=1)
model.fit(X_train,y_train)
```

```
AdaBoostClassifier
AdaBoostClassifier(learning_rate=1, n_estimators=100)
```

```
model.predict([[2,0,1]])
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Ada
  warnings.warn(
array([1])
```