

ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Experiment-1 (A): Study of Prolog.

OBJECTIVE: Prolog programming basics

What is Prolog?

- Prolog stands for *programming in logic*.
- Prolog is a declarative language, which means that a program consists of data based on the facts and rules (Logical relationship) rather than computing how to find a solution.
- A logical relationship describes the relationships which hold for the given application.
- To obtain the solution, the user asks a question rather than running a program. When a user asks a question, then to determine the answer, the run time system searches through the database of facts and rules.
- Prolog is a declarative language that means we can specify what problem we want to solve rather than how to solve it.
- Prolog is used in some areas like database, natural language processing, artificial intelligence, but it is pretty useless in some areas like a numerical algorithm or instance graphics.
- In artificial intelligence applications, prolog is used. The artificial intelligence applications can be automated reasoning systems, natural language interfaces, and expert systems. The expert system consists of an interface engine and a database of facts. The prolog's run time system provides the service of an interface engine.

Applications of Prolog

- Specification Language
- Robot Planning
- Natural language understanding
- Machine Learning
- Problem Solving
- Intelligent Database retrieval
- Expert System
- Automated Reasoning

Starting Prolog

- Prolog system is straightforward.
- Prolog will produce a number of lines of headings in the starting, which is followed by a line. It contains just
- ?-
- The above symbol shows the system prompt. The prompt is used to show that the Prolog system is ready to specify one or more goals of sequence to the user.
- Using a full stop, we can terminate the sequence of goals.

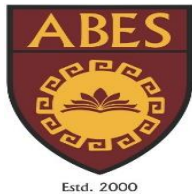
Subject Code: KCS-751A

Subject Name: Artificial Intelligence Lab

Name of Student: Ashish Gupta

Roll No.: 2000320100037

Page 1



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

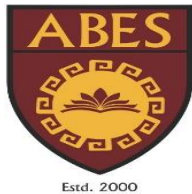
Year/Semester: 4th /VII

Section: A

- **For example:**
- **?- write('Welcome to AI lab Class'),nl,write('Example of Prolog'),nl.**
- **nl** indicates 'start a new line'. When we press 'return' key, the above line will show the effect like this:
Welcome to AI Lab Class
Example of Prolog
yes
- **?- prompt** shows the sequence of goal which is entered by the user. The user will not type the prompt.
- Prolog system will automatically generate this prompt. It means that it is ready to receive a sequence of goals.
- To show that the goals have succeeded, we will output yes.
- **'Query'** is a sequence of one or more goals.

Prolog Programs

- To write a Prolog program, firstly, the user has to write a program which is written in the Prolog language, load that program, and then specify a sequence of one or more goals at the prompt.
- To create a program in Prolog, the simple way is to type it into the text editor and then save it as a text file like prolog1.pl.
- The following example shows a simple program of Prolog. The program contains *three components, which are known as clauses. Each clause is terminated using a full stop.*
dog(rottweiler).
cat(munchkin).
animal(A) :- cat(A).
- Using the built-in predicate **'consult'**, the above program can be loaded in the Prolog system.
- **?-consult('prolog1.pl').**
- This shows that prolog1.pl file exists, and the prolog program is systemically correct, which means it has valid clauses, the goal will succeed, and to confirm that the program has been correctly read, it produces one or more lines of output. e.g.,
- **?-
0.00 seconds to consult prolog1.pl
?-**
- The alternative of 'consult' is 'Load', which will exist on the menu option if the Prolog system has a graphical user interface.
- When the program is loaded, the clause will be placed in a storage area, and that storage area is known as the Prolog database. In response to the system prompt, specify a sequence of goals, and it will cause Prolog to search for and use the clauses necessary to evaluate the goals.



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Terminology

In the following program, three lines show the clauses.

dog(rottweiler).

cat(munchkin).

animal(A) :- cat(A).

Using the full stop, each clause will be terminated. Prolog programs have a sequence of clauses. Facts or rules are described by these clauses.

Example of **facts** is **dog(rottweiler)** and **cat(munchkin)**. They mean that '**rottweiler**' is a dog' and '**munchkin**' is a cat'.

Dog is called a predicate. Dog contains one argument. Word '**rottweiler**' enclosed in bracket(). Rottweiler is called an atom.

The example of rule is the final line of the program.

animal(A) :- dog(A).

The colon(:-) character will be read as 'if'. Here A is a variable, and it represents any value. In a natural way, the rule can be read as "If A is an animal, then A is a dog".

The above clause shows that the **rottweiler** is an animal.

Such deduction can also make by Prolog:

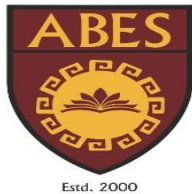
?- animal(rottweiler).

yes

To imply that munchkin is an animal, there is no evidence of this.

?- animal(munchkin).

no



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Experiment-1 (B)

OBJECTIVE: Write simple fact for the statements using PROLOG.

Facts

A fact is like a predicate expression. It is used to provide a declarative statement about the problem. In a Prolog expression, when a variable occurs, it is assumed to be universally quantified. Facts are specified in the form of the **head**. Head is known as the clause head. It will take in the same way as the goal entered at the prompt by the user.

```
cat(bengal).          /* bengal is a cat */
dog(rottweiler).      /* rottweiler is a dog */
likes(Jolie, Kevin).   /* Jolie likes Kevin */
likes(A, Kevin).       /* Everyone likes Kevin */
likes(Jolie, B).       /* Jolie likes everybody */
likes(B, Jolie), likes(Jolie, B). /* Everybody likes Jolie and Jolie likes everybody */
/
likes(Jolie, Kevin); likes(Jolie, Ray). /* Jolie likes Kevin or Jolie likes Ray */
not(likes(Jolie, pasta)). /* Jolie does not like pasta */
```

Queries

In Prolog, the query is the action of asking the program about the information which is available within its database. When a Prolog program is loaded, we will get the query prompt,

?-

After this, we can ask about the information to the run time system. Using the above simple database, we can ask a question to the program like

?- 'It is sunny'.

and it will give the answer

yes

?-

The system responds to the query with yes if the database information is consistent to answer the query. Using the available database information, we can also check that the program is capable of proving the query true. No indicates that the fact is not deducible based on the available information.

The system answers no to the query if the database does not have sufficient information.

?- 'It is cold'.

no

?-

Write simple fact for following:

- Ram likes mango. - likes(ram ,mango).
- Seema is a girl. – girl(seema).

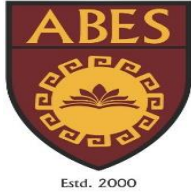
Subject Code: KCS-751A

Subject Name: Artificial Intelligence Lab

Name of Student: Ashish Gupta

Roll No.: 2000320100037

Page 4



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

- c. Bill likes Cindy. – likes(bill, cindy).
- d. Rose is red. – red(rose).
- e. John owns gold. – owns(john, gold).

?-likes(ram,What).

What= mango

?-likes(Who,cindy).

Who= cindy

?-red(What).

What= rose

?-owns(Who,What).

Who= john

What= gold.

Program:

sister(sita,gita).

Output:

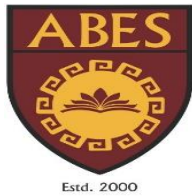
?- sister(gita,sita).
false.

?- sister(gita,sita).
true.

?- sister(what,gita).
What = sita.

Screenshot Output-

```
?- sister(gita, sita).  
false.  
  
?- sister(sita,gita).  
true.  
  
?- sister(What,gita).  
What = sita.
```



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Experiment-2 (A)

OBJECTIVE: Write predicates One converts centigrade temperatures to Fahrenheit, the other checks if a temperature is below freezing.

Formula for Centigrade (C) temperatures to Fahrenheit (F) -

$$F = C * 9 / 5 + 32$$

Rule

Centigrade to Fahrenheit (c_to_f)F is $C * 9 / 5 + 32$

Program- Goal to find Fahrenheit temperature and freezing point

c_to_f(C,F) :-

F is $C * 9 / 5 + 32$.

freezing(F) :-

$F \leq 32$.

Output:

?- c_to_f(100,X).

X = 212

?- freezing(15) .

Yes

?- freezing(45).

No

Screenshot Output-

```
?- c_to_f(100, F).
```

```
F = 212.
```

```
?- freezing(40).
```

```
false.
```

```
?- freezing(20).
```

```
true.
```

```
?- |
```

Experiment-2 (B)

OBJECTIVE: Write a program to solve the Monkey Banana problem.

Monkey wants the bananas but he can't reach them.

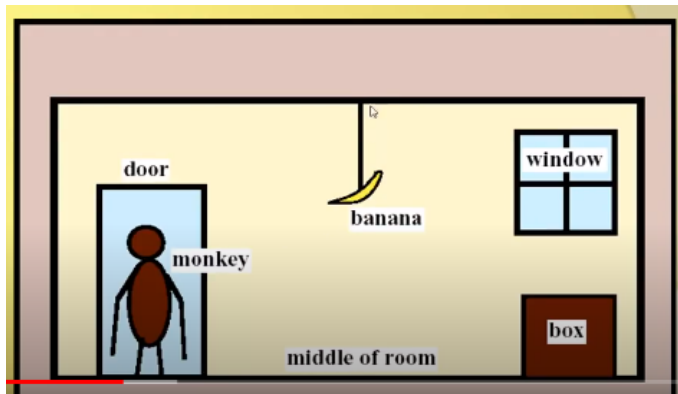
What shall he do? The monkey is in the room.

Suspended from the roof, just out of his reach, is a bunch of bananas.

In the corner of the room is a box. The monkey desperately wants to grasp bananas.

After several unsuccessful attempts to reach the bananas:

1. The monkey walks to the box.
2. Pushes it under the bananas.
3. Climb on the box.
4. Picks the banana & eats them.



Program:

on(floor,monkey).

on(floor,box).

in(room,monkey).

in(room,box).

at(ceiling,banana).

strong(monkey).

grasp(monkey).

climb(monkey,box).

push(monkey,box):-

strong(monkey).

under(banana,box):-

push(monkey,box).

canreach(banana,monkey):-

at(floor,banana);

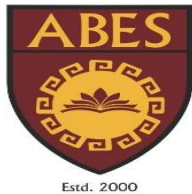
at(ceiling,banana);

under(banana,box).

canget(banana,monkey):-

canreach(banana,monkey),grasp(monkey).

OUTPUT:




ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

 SWI-Prolog (AMD64, Multi-threaded, version 8.2.1)

File Edit Settings Run Debug Help

Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.1)

SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.

Please run ?- license. for legal details.

For online help and background, visit <https://www.swi-prolog.org>

For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- canget(banana, monkey).

ERROR: Unknown procedure: canget/2 (DWIN could not correct goal)

?-

% c:/users/aditi/documents/prolog/monkey compiled 0.00 sec, -2 clauses

?- canget(banana, monkey).

true.

?- trace.

true.

[trace] ?- canget(banana, monkey).

Call: (10) canget(banana, monkey) ? creep

Call: (11) canreach(banana, monkey) ? creep

Call: (12) at(floor, banana) ? creep

Fail: (12) at(floor, banana) ? creep

Redo: (11) canreach(banana, monkey) ? creep

Call: (12) at(ceiling, banana) ? creep

Fail: (12) at(ceiling, banana) ? creep

Redo: (11) canreach(banana, monkey) ? creep

Call: (12) under(banana, box) ? creep

Call: (13) push(monkey, box) ? creep

Call: (14) strong(monkey) ? creep

Exit: (14) strong(monkey) ? creep

Exit: (13) push(monkey, box) ? creep

Exit: (12) under(banana, box) ? creep

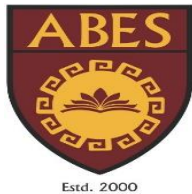
Exit: (11) canreach(banana, monkey) ? creep

Call: (11) grasp(monkey) ? creep

Exit: (11) grasp(monkey) ? creep

Exit: (10) canget(banana, monkey) ? creep

true.



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Experiment-3

OBJECTIVE: Write a program to design medical diagnosis expert system in SWI Prolog. The system must work for diagnosis of following diseases:-

- a) measles
- b) German measles
- c) Flu
- d) common cold
- e) mumps
- f). chickenpox

Program Code:

```
write('What is the patient's name? '),
read(Patient),get_single_char(Code),
hypothesis(Patient,Disease),
write_list([Patient,', probably has ',Disease,'],),nl.
```

```
go :-
write('Sorry, I don't seem to be able to'),nl,
write('diagnose the disease. '),nl.
```

```
symptom(Patient,fever) :-
verify(Patient," have a fever (y/n) ?").
symptom(Patient,rash) :-
verify(Patient," have a rash (y/n) ?").
symptom(Patient,headache) :-
verify(Patient," have a headache (y/n) ?").
symptom(Patient,runny_nose) :-
verify(Patient," have a runny_nose (y/n) ?").
symptom(Patient,conjunctivitis) :-
verify(Patient," have a conjunctivitis (y/n) ?").
symptom(Patient,cough) :-
verify(Patient," have a cough (y/n) ?").
symptom(Patient,body_ache) :-
verify(Patient," have a body_ache (y/n) ?").
symptom(Patient,chills) :-
verify(Patient," have a chills (y/n) ?").
symptom(Patient,sore_throat) :-
verify(Patient," have a sore_throat (y/n) ?").
symptom(Patient,sneezing) :-
verify(Patient," have a sneezing (y/n) ?").
symptom(Patient,swollen_glands) :-
```

Subject Code: KCS-751A

Subject Name: Artificial Intelligence Lab

Name of Student: Ashish Gupta

Roll No.: 2000320100037

Page 9



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

verify(Patient," have a swollen_glands (y/n) ?").

```
ask(Patient,Question) :-  
    write(Patient),write(', do you'),write(Question),  
    read(N),  
    ( (N == yes ; N == y)  
    ->  
    assert(yes(Question)) ;  
    assert(no(Question)), fail).
```

:- dynamic yes/1,no/1.

```
verify(P,S) :-  
    (yes(S) -> true ;  
    (no(S) -> fail ;  
    ask(P,S))).
```

```
undo :- retract(yes(_)),fail.  
undo :- retract(no(_)),fail.  
undo.
```

```
hypothesis(Patient,german_measles) :-  
    symptom(Patient,fever),  
    symptom(Patient,headache),  
    symptom(Patient,runny_nose),  
    symptom(Patient,rash).
```

```
hypothesis(Patient,common_cold) :-  
    symptom(Patient,headache),  
    symptom(Patient,sneezing),  
    symptom(Patient,sore_throat),  
    symptom(Patient,runny_nose),  
    symptom(Patient,chills).
```

```
hypothesis(Patient,measles) :-  
    symptom(Patient,cough),  
    symptom(Patient,sneezing),  
    symptom(Patient,runny_nose).
```

```
hypothesis(Patient,flu) :-  
    symptom(Patient,fever),  
    symptom(Patient,headache),  
    symptom(Patient,body_ache),  
    symptom(Patient,conjunctivitis),
```

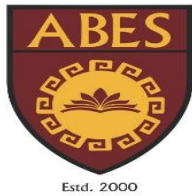
Subject Code: KCS-751A

Subject Name: Artificial Intelligence Lab

Name of Student: Ashish Gupta

Roll No.: 2000320100037

Page 10



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

```
symptom(Patient,chills),  
symptom(Patient,sore_throat),  
symptom(Patient,runny_nose),  
symptom(Patient,cough).
```

```
hypothesis(Patient,mumps) :-  
symptom(Patient,fever),  
symptom(Patient,swollen_glands).
```

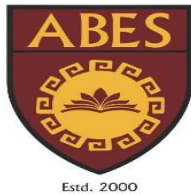
```
hypothesis(Patient,chicken_pox) :-  
symptom(Patient,fever),  
symptom(Patient,chills),  
symptom(Patient,body_ache),  
symptom(Patient,rash).
```

```
write_list([]).  
write_list([Term| Terms]) :-  
write(Term),  
write_list(Terms).
```

```
response(Reply) :-  
get_single_char(Code),  
put_code(Code), nl,  
char_code(Reply, Code).
```

OUTPUT:

```
What is the patient's name? Ashish.  
_17584, do you have a fever (y/n) ?y.  
_17584, do you have a headache (y/n) ?|: y.  
_17584, do you have a runny_nose (y/n) ?|: no.  
_17584, do you have a sneezing (y/n) ?|: no.  
_17584, do you have a cough (y/n) ?|: y.  
_17584, do you have a body_ache (y/n) ?|: y.  
_17584, do you have conjunctivitis (y/n) ?|: y.  
_17584, do you have a chills (y/n) ?|: y.  
_17584, do you have a sore_throat (y/n) ?|: y.  
_17584, do you have a swollen_glands (y/n) ?|: y.  
_17584, probably has mumps.  
true .
```



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th/VII

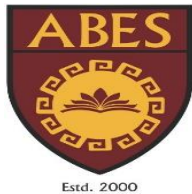
Section: A

Screenshot Output:

```
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
Warning: c:/users/adain/desktop/go.pl:1:
Warning: Singleton variables: [Code]
% c:/Users/adain/Desktop/go.pl compiled 0.00 sec, 27 clauses
?- go.
What is the patient's name? Ashish
_17584, do you have a fever (y/n) ?y.
_17584, do you have a headache (y/n) ?|: y.
_17584, do you have a runny_nose (y/n) ?|: no.
_17584, do you have a sneezing (y/n) ?|: no.
_17584, do you have a cough (y/n) ?|: y.
_17584, do you have a body_ache (y/n) ?|: y.
_17584, do you have a conjunctivitis (y/n) ?|: y.
_17584, do you have a chills (y/n) ?|: y.
_17584, do you have a sore_throat (y/n) ?|: y.
_17584, do you have a swollen_glands (y/n) ?|: y.
_17584, probably has mumps.
true .
```



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Experiment-4 (A)

OBJECTIVE: WAP to implement factorial, Fibonacci of a given number.

Program-

factorial(0,1).

factorial(N,F):-

N>0,

N1 is N-1,

factorial(N1,F1),

F is N * F1.

Goal- To find Factorial of the number.

Screenshot Output-

```
?- factorial(5, A).
```

```
A = 120 .
```

```
?- factorial(5, 120).
```

```
true |
```

The Fibonacci sequence is a sequence where the next term is the sum of the previous two terms. The first two terms of the Fibonacci sequence are 0 followed by 1.

The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21

Fibonacci-

Program-

fib(0, 1) :- !.

fib(1, 1) :- !.

fib(N, F) :-

N > 1,

N1 is N-1,

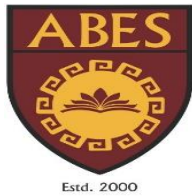
N2 is N-2,

fib(N1, F1),

fib(N2, F2),

F is F1+F2

Goal- To find Fibonacci number.



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Screenshot output:

?- fib(4, A).

A = 5.

?- fib(5,5).

false.

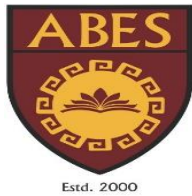
?- fib(0,A).

A = 1.

?- fib(1,A).

A = 1.

?- |



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

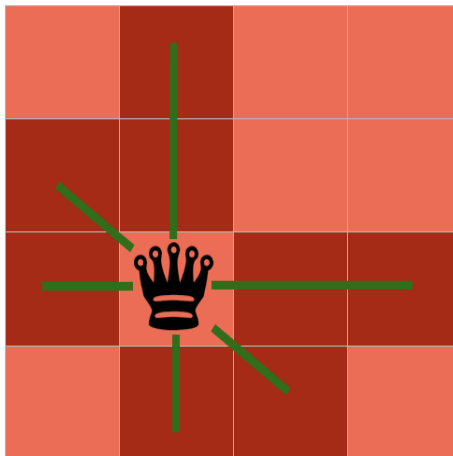
Year/Semester: 4th /VII

Section: A

Experiment-4 (B)

OBJECTIVE: Write a program to solve 4-Queen problem.

N queens problem is one of the most common examples of backtracking. Our goal is to arrange N queens on an NxN chessboard such that no queen can strike down any other queen. A queen can attack horizontally, vertically, or diagonally.

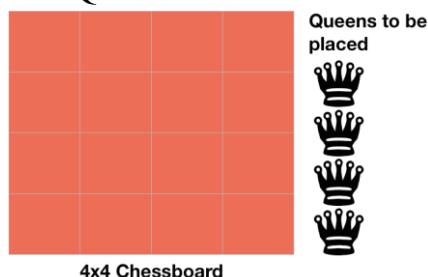


Cells attacked by the queen

So, we start by placing the first queen anywhere arbitrarily and then place the next queen in any of the safe places. We continue this process until the number of unplaced queens becomes zero (a solution is found) or no safe place is left. If no safe place is left, then we change the position of the previously placed queen.

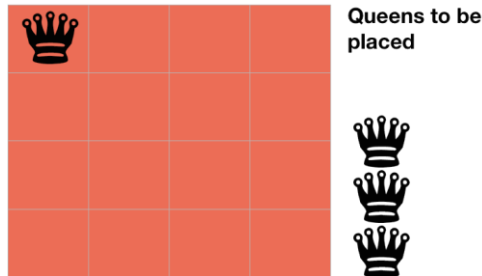
Let's test this algorithm on a 4x4 chessboard.

Using Backtracking to Solve N Queens

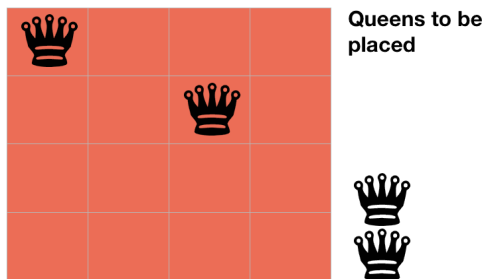


The above picture shows a 4x4 chessboard and we have to place 4 queens on it.

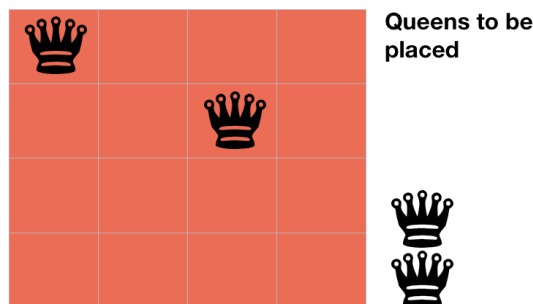
So, we will start by placing the first queen in the first row.



Now, the second step is to place the second queen in a safe position. Also, we can't place the queen in the first row, so we will try putting the queen in the second row this time.

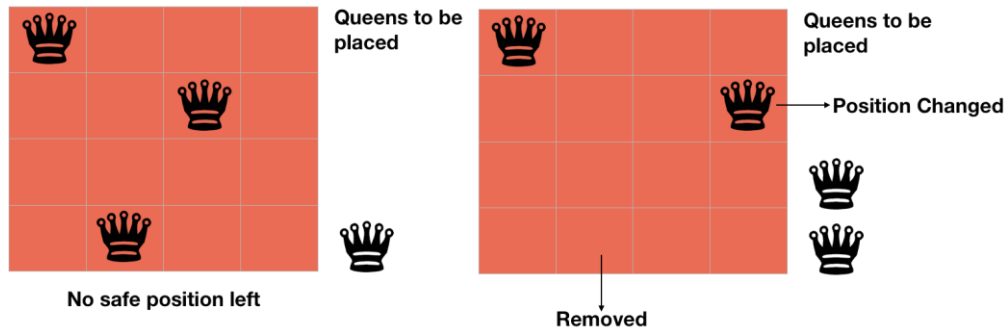


Let's place the third queen in a safe position, somewhere in the third row.

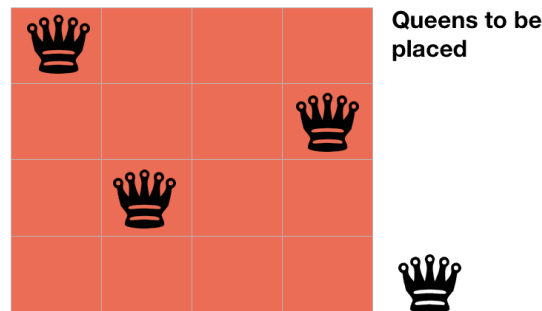


Now, we can see that there is no safe place where we can put the last queen. So, we will just change the position of the previous queen i.e., backtrack and change the previous decision.

Also, there is no other position where we can place the third queen, so we will go back one more step and change the position of the second queen.



And now we will place the third queen again in a safe position other than the previously placed position in the third row.



We will continue this process and finally, we will get the solution as shown below.



Program:

```
% render solutions nicely.  
:- use_rendering(chess).
```

```
%% queens(+N, -Queens) is nondet.  
%  
% @param Queens is a list of column numbers for placing the queens.  
% @author Richard A. O'Keefe (The Craft of Prolog)
```



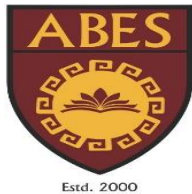
ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

```
queens(N, Queens) :-  
    length(Queens, N),  
    board(Queens, Board, 0, N, _, _),  
    queens(Board, 0, Queens).  
  
board([], [], N, N, _, _).  
board(_|Queens, [Col-Vars|Board], Col0, N, _|VR, VC) :-  
    Col is Col0+1,  
    functor(Vars, f, N),  
    constraints(N, Vars, VR, VC),  
    board(Queens, Board, Col, N, VR, _|VC)).  
  
constraints(0, _, _, _) :- !.  
constraints(N, Row, [R|Rs], [C|Cs]) :-  
    arg(N, Row, R-C),  
    M is N-1,  
    constraints(M, Row, Rs, Cs).  
  
queens([], _, []).  
queens([C|Cs], Row0, [Col|Solution]) :-  
    Row is Row0+1,  
    select(Col-Vars, [C|Cs], Board),  
    arg(Row, Vars, Row-Row),  
    queens(Board, Row, Solution).  
  
/** <examples>  
  
?- queens(8, Queens).  
  
*/
```




ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

OUTPUT:

 SWI-Prolog (AMD64, Multi-threaded, version 8.2.1)

File Edit Settings Run Debug Help

Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit <https://www.swi-prolog.org>
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-

% c:/users/aditi/documents/prolog/nqueenss compiled 0.00 sec, -2 clauses

?- queens(4,Queens).

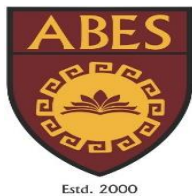
Queens = [2, 4, 1, 3] .

?- queens(8,Queens).

Queens = [1, 5, 8, 6, 3, 7, 2, 4] .

?- queens(10,Queens).

Queens = [1, 3, 6, 8, 10, 5, 9, 2, 4|...] ■



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Experiment-5

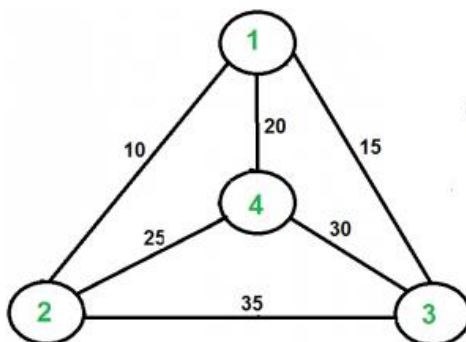
OBJECTIVE: Write a program to solve traveling salesman problem.

Given a set of cities and distances between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.

Note the difference between Hamiltonian Cycle and TSP. The Hamiltonian cycle problem is to find if there exists a tour that visits every city exactly once. Here we know that Hamiltonian Tour exists (because the graph is complete) and in fact, many such tours exist, the problem is to find a minimum weight Hamiltonian Cycle.

For example, consider the graph shown in the figure on the right side. A TSP tour in the graph is 1-2-4-3-1. The cost of the tour is $10+25+30+15$ which is 80.

The problem is a famous NP-hard problem. There is no polynomial-time known solution for this problem.



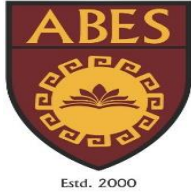
Output of Given Graph:

minimum weight Hamiltonian Cycle :

$$10 + 25 + 30 + 15 := 80$$

Steps for implementation:

1. Consider city 1 as the starting and ending point. Since the route is cyclic, we can consider any point as a starting point.
2. Generate all $(n-1)!$ Permutations of cities.
3. Calculate the cost of every permutation and keep track of the minimum cost permutation.
4. Return the permutation with minimum cost.



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Program: # Python3 program to implement traveling salesman
problem using naive approach.
from sys import maxsize
from itertools import permutations
V = 4

implementation of traveling Salesman Problem
def travellingSalesmanProblem(graph, s):

```
    # store all vertex apart from source vertex
    vertex = []
    for i in range(V):
        if i != s:
            vertex.append(i)
```

```
    # store minimum weight Hamiltonian Cycle
    min_path = maxsize
    next_permutation=permutations(vertex)
    for i in next_permutation:
```

```
        # store current Path weight(cost)
        current_pathweight = 0
```

```
        # compute current path weight
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]
```

```
        # update minimum
        min_path = min(min_path, current_pathweight)
```

```
    return min_path
```

Driver Code

```
if __name__ == "__main__":
```

```
    # matrix representation of graph
    graph = [[0, 10, 15, 20], [10, 0, 35, 25],
             [15, 35, 0, 30], [20, 25, 30, 0]]
    s = 0
    print(travellingSalesmanProblem(graph, s))
```

Subject Code: KCS-751A

Subject Name: Artificial Intelligence Lab

Name of Student: Ashish Gupta

Roll No.: 2000320100037

Page 21



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Output

```
current_pathweight += graph[k][s]

# update minimum
min_path = min(min_path, current_pathweight)

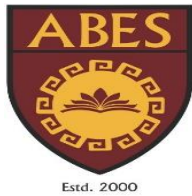
return min_path

# Driver Code
if __name__ == "__main__":

    # matrix representation of graph
    graph = [[0, 10, 15, 20], [10, 0, 35, 25],
             [15, 35, 0, 30], [20, 25, 30, 0]]
    s = 0
    print(travellingSalesmanProblem(graph, s))

80

In [ ]: |
```



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Experiment-6

OBJECTIVE: Write a program to solve water jug problem.

Problem: You are given two jugs, a 4-gallon one and a 3-gallon one. Neither has any measuring mark on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into the 4-gallon jug.

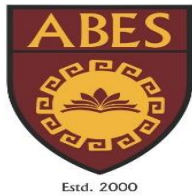
Solution:

- The state space for this problem can be described as the set of ordered pairs of integers **(x,y)**
- Where,
- X represents the quantity of water in the 4-gallon jug **X= 0,1,2,3,4**
- Y represents the quantity of water in 3-gallon jug **Y=0,1,2,3**
- **Start State: (0,0)**
- **Goal State: (2,0)**

Generate production rules for the water jug problem

Production Rules:

Rule	State	Process
1	$(X,Y \mid X < 4)$	$(4,Y)$ {Fill 4-gallon jug}
2	$(X,Y \mid Y < 3)$	$(X,3)$ {Fill 3-gallon jug}
3	$(X,Y \mid X > 0)$	$(0,Y)$ {Empty 4-gallon jug}
4	$(X,Y \mid Y > 0)$	$(X,0)$ {Empty 3-gallon jug}
5	$(X,Y \mid X+Y \geq 4 \wedge Y > 0)$	$(4,Y-(4-X))$ {Pour water from 3-gallon jug into 4-gallon jug until 4-gallon jug is full}
6	$(X,Y \mid X+Y \geq 3 \wedge X > 0)$	$(X-(3-Y),3)$ {Pour water from 4-gallon jug into 3-gallon jug until 3-gallon jug is full}
7	$(X,Y \mid X+Y \leq 4 \wedge Y > 0)$	$(X+Y,0)$ {Pour all water from 3-gallon jug into 4-gallon jug}
8	$(X,Y \mid X+Y \leq 3 \wedge X > 0)$	$(0,X+Y)$ {Pour all water from 4-gallon jug into 3-gallon jug}
9	$(0,2)$	$(2,0)$ {Pour 2 gallon water from 3 gallon jug into 4 gallon jug}



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Initialization:

Start State: (0,0)

Apply Rule 2:

$(X,Y \mid Y < 3) \rightarrow (X,3)$
{Fill 3-gallon jug}

Now the state is (X,3)

Iteration 1:

Current State: (X,3)

Apply Rule 7:

$(X,Y \mid X+Y \leq 4 \wedge Y > 0) \rightarrow (X+Y,0)$
{Pour all water from 3-gallon jug into 4-gallon jug}

Now the state is (3,0)

Iteration 2:

Current State : (3,0)

Apply Rule 2:

$(X,Y \mid Y < 3) \rightarrow (3,3)$
{Fill 3-gallon jug}

Now the state is (3,3)

Iteration 3:

Current State:(3,3)

Apply Rule 5:

$(X,Y \mid X+Y > 4 \wedge Y > 0) \rightarrow (4,Y-(4-X))$
{Pour water from 3-gallon jug into 4-gallon jug until 4-gallon jug is full}

Now the state is (4,2)

Iteration 4:

Current State : (4,2)

Apply Rule 3:

$(X,Y \mid X > 0) \rightarrow (0,Y)$
{Empty 4-gallon jug}

Now state is (0,2)

Iteration 5:

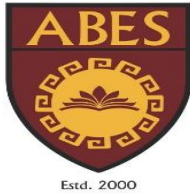
Current State : (0,2)

Apply Rule 9:

$(0,2) \rightarrow (2,0)$
{Pour 2 gallon water from 3 gallon jug into 4 gallon jug}

Now the state is (2,0)

Goal Achieved.



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

OUTPUT:

```
Exp_6_JAI GARG.py - C:\Users\gargy\OneDrive\Desktop\Exp_6_JAI GARG.py (3.8)
File Edit Format Run Options Window Help

x = 0
y = 0
m = 4
n = 3
print("Initial state = (0,0)")
print("Capacities = (4,3)")
print("Goal state = (2,0)")
while x != 2:
    r = int(input("Enter rule"))
    if (r == 1):
        x = m
    elif (r == 2):
        y = n
    elif (r == 3):
        x = 0
    elif (r == 4):
        y = 0
    elif (r == 5):
        t = n - y
        y = n
        x = t
    elif (r == 6):
        t = m - x
        x = m
        y = t
    elif (r == 7):
        y += x
        x = 0
    elif (r == 8):
        x += y
        y = 0
    print(x, y)
print("Goal state has reached")
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:6c08332, May 13 2020, 22:20:19) [MSC v.1919 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\gargy\OneDrive\Desktop\Exp_6_JAI GARG.py =====
Initial state = (0,0)
Capacities = (4,3)
Goal state = (2,0)
Enter rule 1
1 0
Enter rule 2
1 3
Enter rule 4
1 0
Enter rule 7
0 1
Enter rule 1
4 1
Enter rule 5
2 3
Goal state has reached
>>>
===== RESTART: C:\Users\gargy\OneDrive\Desktop\Exp_6_JAI GARG.py =====
Initial state = (0,0)
Capacities = (4,3)
Goal state = (2,0)
Enter rule 2
0 3
Enter rule 8
3 0
Enter rule 2
3 3
Enter rule 6
4 2
Enter rule 3
0 2
Enter rule 8
2 0
Goal state has reached
>>>
```

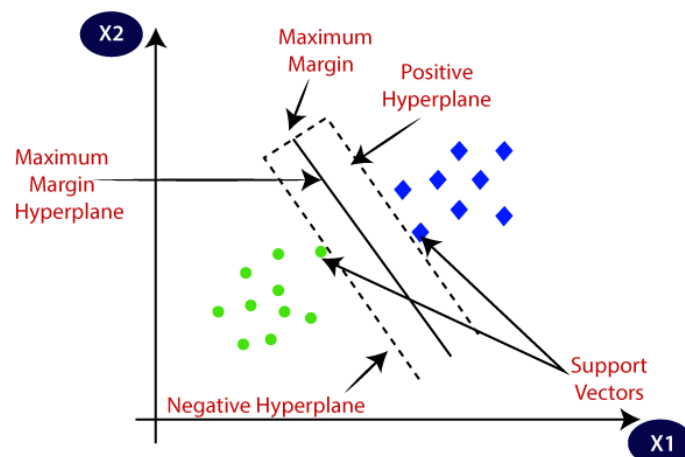
Experiment-7

OBJECTIVE: Write a Python program to Support vector Machine algorithm.

SVM is one of the most popular Supervised Learning algorithms, 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:



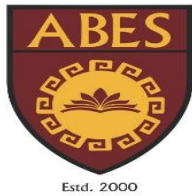
Types of SVM

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.



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

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.

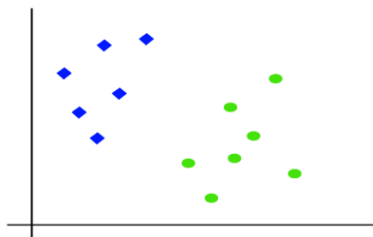
- **Support Vectors:**

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.

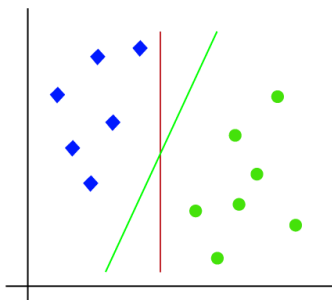
How does SVM works?

Linear SVM:

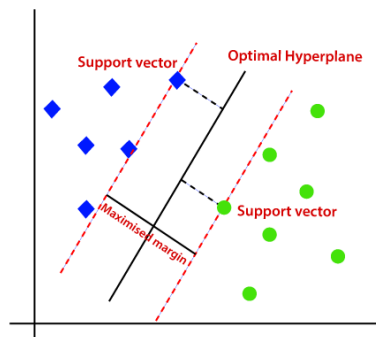
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x_1 and x_2 . We want a classifier that can classify the pair(x_1 , x_2) of coordinates in either green or blue. Consider the below image:



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:

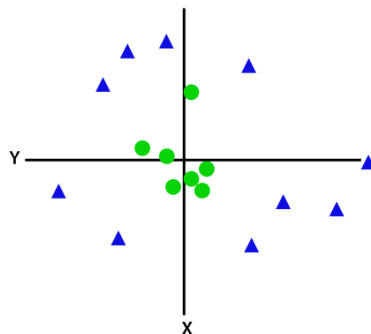


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is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



Non-Linear SVM:

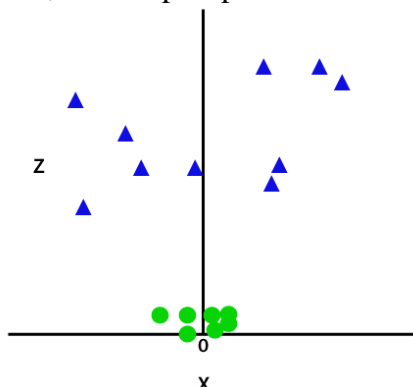
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



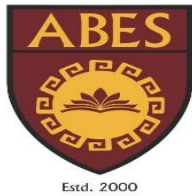
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$z = x^2 + y^2$$

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



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

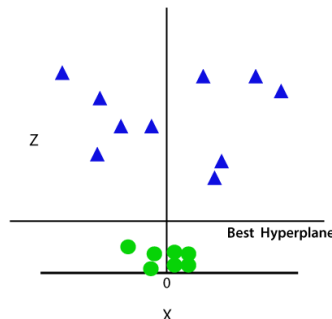


ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A



Program:

In the model the building part, you can use the cancer dataset, which is a very famous multi-class classification problem. This dataset is computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

This data has two types of cancer classes: **malignant (harmful)** and **benign (not harmful)**.

Here, you can build a model to classify the type of cancer. The dataset is available in the scikit-learn library or you can also download it from the UCI Machine Learning Library.

Loading Data

Let's first load the required dataset you will use.

Exploring Data

After you have loaded the dataset, you might want to know a little bit more about it. You can check feature and target names.

You can also check the shape of the dataset using shape.

```
jupyter Untitled Last Checkpoint: 12/02/2020 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

In [1]:
Out[1]: 10

In [2]: from sklearn import datasets
cancer = datasets.load_breast_cancer()
print("Features: ", cancer.feature_names)
print("Labels: ", cancer.target_names)

Features: ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
'mean smoothness' 'mean compactness' 'mean concavity'
'mean concave points' 'mean symmetry' 'mean fractal dimension'
'radius error' 'texture error' 'perimeter error' 'area error'
'smoothness error' 'compactness error' 'concavity error'
'concave points error' 'symmetry error' 'fractal dimension error'
'worst radius' 'worst texture' 'worst perimeter' 'worst area'
'worst smoothness' 'worst compactness' 'worst concavity'
'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels: ['malignant' 'benign']

In [3]: cancer.data.shape
Out[3]: (569, 30)
```

Subject Code: KCS-751A

Subject Name: Artificial Intelligence Lab

Name of Student: Ashish Gupta

Roll No.: 2000320100037

Page 29



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Let's check top 5 records of the feature set.

```
In [4]: print(cancer.data[0:5])

[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
[2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
 7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
 5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
 2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
 2.750e-01 8.902e-02]
[1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
 1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
 6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
 2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
 3.613e-01 8.758e-02]
[1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
 1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
 9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
 2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
 6.638e-01 1.730e-01]
[2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
 1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
 1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
 1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
 2.364e-01 7.678e-02]]
```

Let's take a look at the target set.

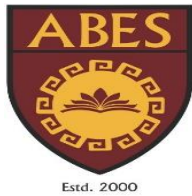
```
In [5]: print(cancer.target)

[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 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 0 1 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Split the dataset by using the function `train_test_split()`. you need to pass 3 parameters features, target, and test_set size. Additionally, you can use `random_state` to select records randomly.



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Generating Model

Let's build support vector machine model. First, import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in SVC() function.

Then, fit your model on train set **using fit()** and perform prediction on the test set using predict().

Evaluating the Model

Let's estimate how accurately the classifier or model can predict the breast cancer of patients.

Accuracy can be computed by comparing actual test set values and predicted values.

```

In [6]: # Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3, random_state=100) # 70% training ar

In [7]: #Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

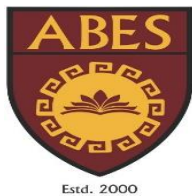
#Predict the response for test dataset
y_pred = clf.predict(X_test)

In [8]: #Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.9649122807017544

```



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

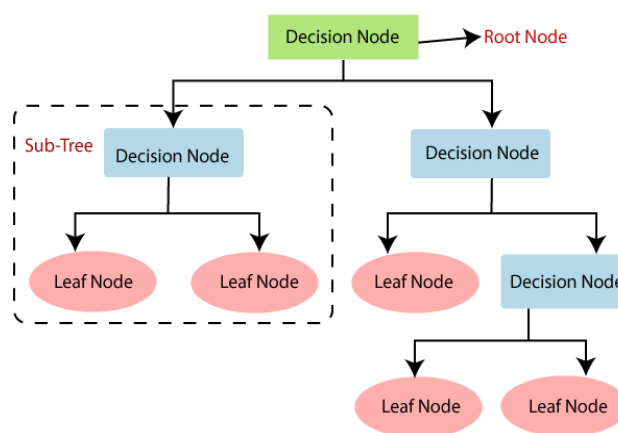
Year/Semester: 4th /VII

Section: A

Experiment-8

OBJECTIVE: Write a python program for Decision tree algorithm.

Introduction: 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. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome**.



Below are the two reasons for using the Decision tree:

1. Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
2. The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

- ☐ **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- ☐ **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- ☐ **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- ☐ **Branch/Sub Tree:** A tree formed by splitting the tree.
- ☐ **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- ☐ **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

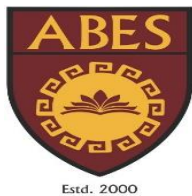
Subject Code: KCS-751A

Subject Name: Artificial Intelligence Lab

Name of Student: Ashish Gupta

Roll No.: 2000320100037

Page 32



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **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 contains 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**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1. $\text{Information Gain} = \text{Entropy}(S) - [(\text{Weighted Avg}) * \text{Entropy}(\text{each feature})]$

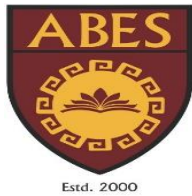
Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$\text{Entropy}(s) = -P(\text{yes}) \log_2 P(\text{yes}) - P(\text{no}) \log_2 P(\text{no})$$

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. Gini Index:



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

PROGRAM:

First, start with importing necessary python packages –

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

Next, download the iris dataset from its weblink as follows –

```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
pima = pd.read_csv(r"C:\pima-indians-diabetes.csv", header = None, names =
col_names)
pima.head()
```

	Pregnant	Glucose	BP	Skin	Insulin	Bmi	Pedigree	Age	Label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	33.6	0.627	50	1
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Now, split the dataset into features and target variable as follows –

```
feature_cols = ['pregnant', 'insulin', 'bmi', 'age', 'glucose', 'bp', 'pedigree']
X = pima[feature_cols] # Features
```

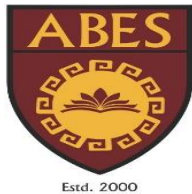
Subject Code: KCS-751A

Subject Name: Artificial Intelligence Lab

Name of Student: Ashish Gupta

Roll No.: 2000320100037

Page 34



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

`y = pima.label # Target variable`

Next, we will divide the data into train and test split. The following code will split the dataset into 70% training data and 30% of testing data –

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 1)
```

Next, train the model with the help of DecisionTreeClassifier class of sklearn as follows –

```
clf = DecisionTreeClassifier()  
clf = clf.fit(X_train,y_train)
```

At last we need to make prediction. It can be done with the help of following script –

```
y_pred = clf.predict(X_test)
```

Next, we can get the accuracy score, confusion matrix and classification report as follows –

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score  
result = confusion_matrix(y_test, y_pred)  
print("Confusion Matrix:")  
print(result)  
result1 = classification_report(y_test, y_pred)  
print("Classification Report:",)  
print(result1)  
result2 = accuracy_score(y_test,y_pred)  
print("Accuracy:",result2)
```

Output

Confusion Matrix:

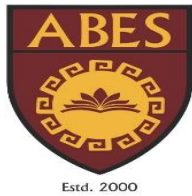
```
[[116 30]
```

```
[ 46 39]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.72	0.79	0.75	146
1	0.57	0.46	0.51	85
micro avg	0.67	0.67	0.67	231
macro avg	0.64	0.63	0.63	231
weighted avg	0.66	0.67	0.66	231

Accuracy: 0.670995670995671



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Experiment-9

OBJECTIVE: Write a python program for Gaussian Naïve Bayes Classifier.

It is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data.

Naive Bayes are a group of supervised machine learning classification algorithms based on the **Bayes theorem**. It is a simple classification technique, but has high functionality. They find use when the dimensionality of the inputs is high.

Bayes Theorem

Bayes Theorem can be used to calculate conditional probability. Being a powerful tool in the study of probability, it is also applied in Machine Learning.

The Formula For Bayes' Theorem Is

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A) \cdot P(B|A)}{P(B)}$$

where:

$P(A)$ = The probability of A occurring

$P(B)$ = The probability of B occurring

$P(A|B)$ = The probability of A given B

$P(B|A)$ = The probability of B given A

$P(A \cap B)$ = The probability of both A and B occurring

Naive Bayes Classifier

Naive Bayes Classifiers are based on the Bayes Theorem. These classifiers assume that the value of a particular feature is independent of the value of any other feature. In a supervised learning situation, Naive Bayes Classifiers are trained very efficiently. Naive Bayed classifiers need a small training data to estimate the parameters needed for classification

Gaussian Naive Bayes

When working with continuous data, an assumption often taken is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. The likelihood of the features is assumed to be-

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Sometimes assume variance

Subject Code: KCS-751A

Subject Name: Artificial Intelligence Lab

Name of Student: Ashish Gupta

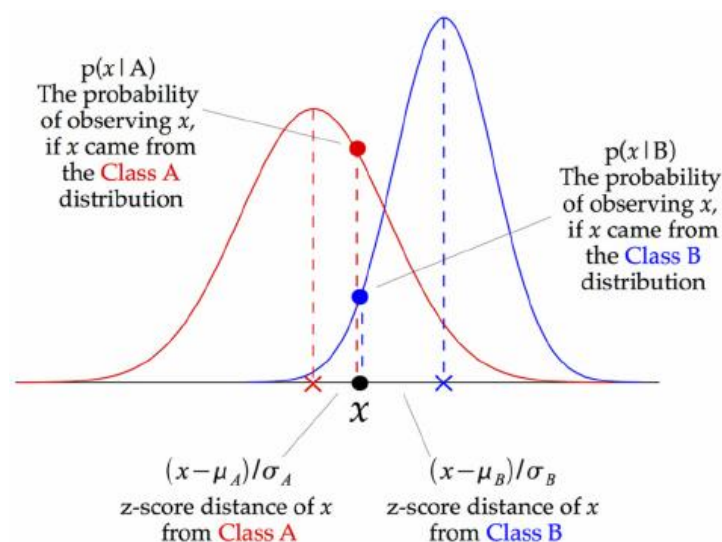
Roll No.: 2000320100037

Page 36

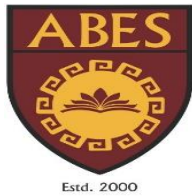
- is independent of Y (i.e., σ_i),
- or independent of X_i (i.e., σ_k)
- or both (i.e., σ)

Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution.

An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co-variance (independent dimensions) between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution.



The above illustration indicates how a Gaussian Naive Bayes (GNB) classifier works. At every data point, the z-score distance between that point and each class-mean is calculated, namely the distance from the class mean divided by the standard deviation of that class.



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Program:

```
# Gaussian Naive Bayes
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
# load the iris datasets
dataset = datasets.load_iris()
# fit a Naive Bayes model to the data
model = GaussianNB()

model.fit(dataset.data, dataset.target)
print(model)
# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

OUTPUT:

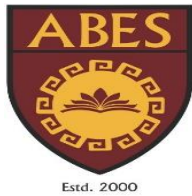
```
GaussianNB()

              precision    recall  f1-score   support

0               1.00        1.00        1.00         50
1               0.94        0.94        0.94         50
2               0.94        0.94        0.94         50

avg / total         0.96        0.96        0.96        150

[[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]
```



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

Experiment-10

OBJECTIVE: Write a python program to Predicting Cardiovascular Disease Using K Nearest Neighbors Algorithm.

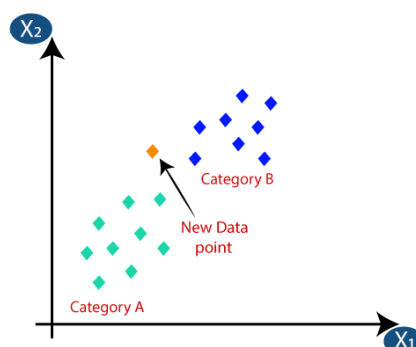
K Nearest Neighborhood Algorithm also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

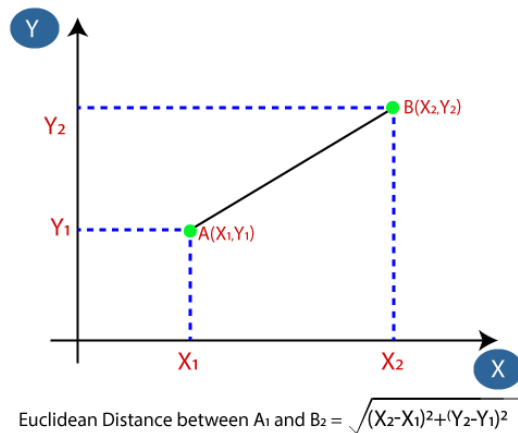
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

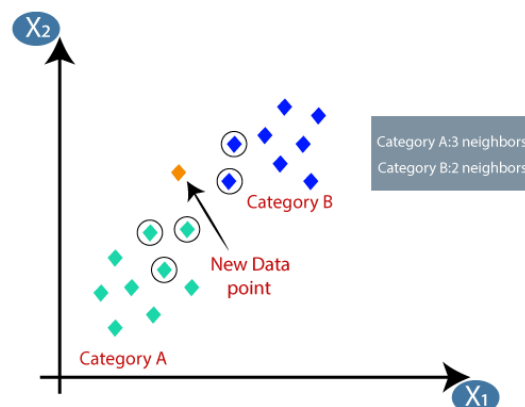


- Firstly, we will choose the number of neighbors, so we will choose the $k=5$.

- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



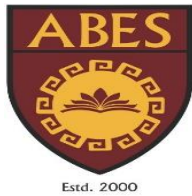
- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

Program:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

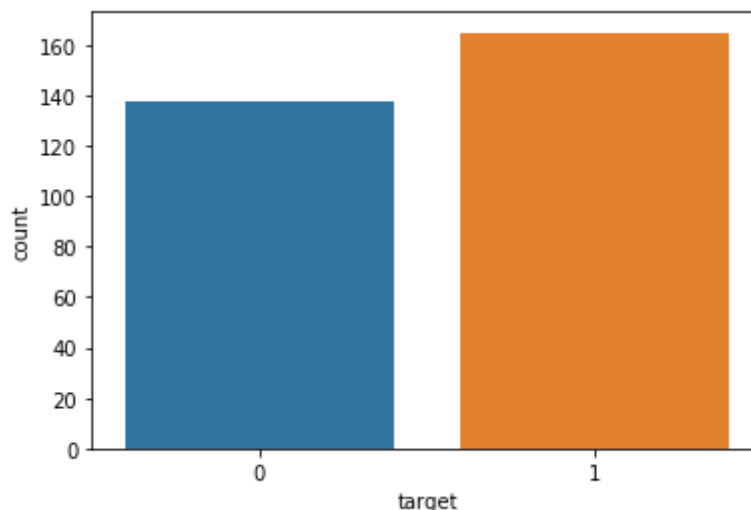
Year/Semester: 4th /VII

Section: A

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
df = pd.read_csv('heart.csv')
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
sns.countplot(df['target'])
```



```
x= df.iloc[:,0:13].values
y= df['target'].values
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)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
error = []
# Calculating error for K values between 1 and 30
for i in range(1, 30):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    pred_i = knn.predict(x_test)
    error.append(np.mean(pred_i != y_test))
plt.figure(figsize=(12, 6))
```

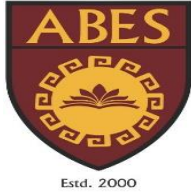
Subject Code: KCS-751A

Subject Name: Artificial Intelligence Lab

Name of Student: Ashish Gupta

Roll No.: 2000320100037

Page 41



ABES Engineering College, Ghaziabad

Department of Computer Science & Engineering

Year/Semester: 4th /VII

Section: A

```
plt.plot(range(1, 30), error, color='red', linestyle='dashed', marker='o',  
         markerfacecolor='blue', markersize=10)  
plt.title('Error Rate K Value')  
plt.xlabel('K Value')  
plt.ylabel('Mean Error')  
print("Minimum error:-",min(error),"at K =",error.index(min(error))+1)
```

```
Minimum error:- 0.13157894736842105 at K = 7
```

```
classifier= KNeighborsClassifier(n_neighbors=7)  
classifier.fit(x_train, y_train)  
y_pred= classifier.predict(x_test)  
from sklearn.metrics import confusion_matrix  
cm= confusion_matrix(y_test, y_pred)
```

```
# Output =>array([[26,  7],  
                  [ 3, 40]], dtype=int64)
```

```
accuracy_score(y_test, y_pred)
```

```
# Output => 0.868421052631579
```