MCA A.Y. 2022-2023

Journals of Artificial Intelligence and Machine Learning

ASM's Institute of Management & Computer Studies, Thane

Candidate Full Name	**OMKAR BHARAT KHANDARE**
Roll No.	MC2223044
Course	Masters of Computer Applications (MCA)
Semester	II
Subject Faculty Incharge	Prof. Abhay More

Audyogik Shikshan Mandal's

Institute of Management & Computer Studies, Thane

Certificate

This is to certify that Mr. **OMKAR BHARAT KHANDARE** Student of **MCA** Course, first year, **Semester 2**, Roll No **MC2223044** has successfully complete the required number of practical in subject of **Artificial Intelligence and Machine Learning** as prescribed by the **University of Mumbai** under our supervision during the academicyear 2022-2023.

Practical In-Charge **Internal Examiner**

Date: **Date:**

External Examiner **Director**

Date: **IMCOST**

College Seal:

INDEX

Sr. No.	PROBLEM STATEMENT	SIGN
1	Implementation of Logic programming using LISP /PROLOG	
1.1	DFS for water jug Problem	
1.2	BFS for tic-tac-toe problem	
1.3	Hill-climbing to solve 8 - Puzzle Problem	
2	Introduction to Python Programming: Learn the different libraries –	
2.1	NumPy	
2.2	Pandas	
2.3	SciPy	
2.4	Matplotlib	
2.5	Scikit Learn	
3	Implementation of Algorithm	
3.1	Linear Regression	
3.2	Logistic regression	
3.3	KNN - classification	
4	Implementation of dimensionality reduction techniques :	
4.1	Features Extraction and Selection	
4.2	Normalization	
4.3	Transformation	
4.4	Principal Components Analysis	
5	Implementation of clustering Algorithm	
5.1	K-Means	
5.2	K-Medoid	
6	Implementation of Classifying data using Support Vector Machines (SVMs).	
7	Implementation of Bagging Algorithm :	
7.1	Decision Tree	
7.2	Random Forest	
8	Implementation of Boosting Algorithm.	
9	Deployment of Machine Learning Models.	

Practical No.1

Aim :- Implementation of Logic programming using LISP /PROLOG.

PROLOG:

Programming Logic language was designed in the 1970s by Alain Colmerauer and a team of researchers .

It was possible to use logic to represent knowledge and to write programs.

It uses a subset of predicate logic and draws its structure from theoretical works of earlier logicians such as Herbrand (1930) and Robinson (1965) on the automation of theorem proving.

PROLOG supports:

- Natural Language Understanding
- Artificial Intelligence Lab
- Formal logic and associated forms of programming
- Reasoning modeling
- Database programming
- Expert System Development Real time AI programs

PROLOG programs are often described as declarative, although they unavoidably also have a procedural element. Programs are based on the techniques developed by logicians to form valid conclusions from available evidence. There are only two components to any program: facts and rules. The PROLOG system reads in the program and simply stores it. The user gives the queries which can be answered by the system using the facts and rules available to it. A simple example, is given below to illustrate the same.

The relationship between the objects and the particular relationship among the objects are explained through the following example.

Each family has three components: husband, wife and children are objects of the family. As the number of children varies from family to family the children are represented by a list that is capable of accommodating any number of items. Each person is, in turn, represented by a structure of four components: name or it specifies the working organization and salary. The family of can be stored in the database by the clause.

1.1 Water jug problem:

In the water jug problem in Artificial Intelligence, we are provided with two jugs: one having the capacity to hold 3 gallons of water and the other has the capacity to hold 4 gallons of water.

There is no other measuring equipment available and the jugs also do not have any kind of marking on them. So, the agent's task here is to fill the 4-gallon jug with 2 gallons of water by using only these two jugs and no other material. Initially, both our jugs are empty.

So, to solve this problem, following set of rules were proposed:

Production rules for solving the water jug problem

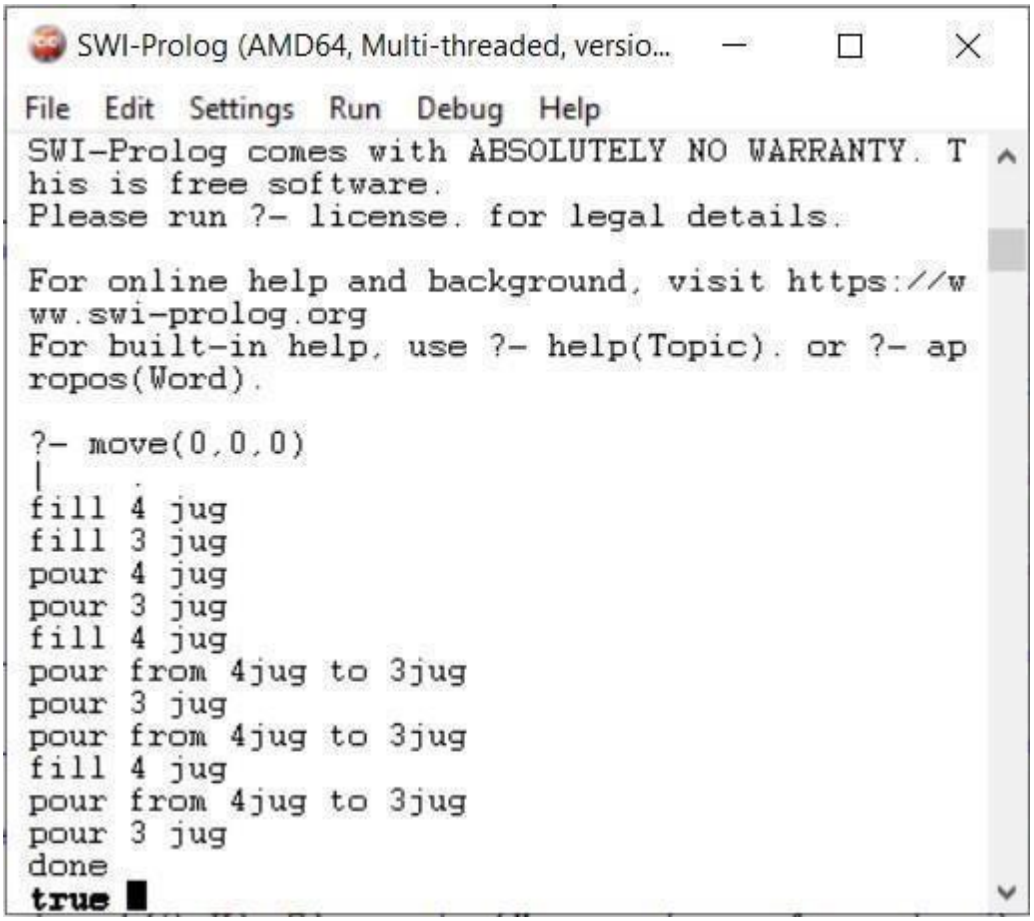
Here, let x denote the 4-gallon jug and y denote the 3-gallon jug.

S.No. Initial State Condition Final state Description of action taken

1. (x,y) If x<4 (4,y) Fill the 4 gallon jug completely
2. (x,y) if y<3 (x,3) Fill the 3 gallon jug completely
3. (x,y) If x>0 (x-d,y) Pour some part from the 4 gallon jug
4. (x,y) If y>0 (x,y-d) Pour some part from the 3 gallon jug
5. (x,y) If x>0 (0,y) Empty the 4 gallon jug
6. (x,y) If y>0 (x,0) Empty the 3 gallon jug
7. (x,y) If (x+y)<7 (4, y-[4-x]) Pour some water from the 3 gallon jug to fill the four gallon jug
8. (x,y) If (x+y)<7 (x-[3-y],y) Pour some water from the 4 gallon jug to fill the 3 gallon jug.
9. (x,y) If (x+y)<4 (x+y,0) Pour all water from 3 gallon jug to the 4 gallon jug
10. (x,y) if (x+y)<3 (0, x+y) Pour all water from the 4 gallon jug to the 3 gallon jug To solve the water jug problem in a minimum number of moves, following set of rules in the given sequence should be performed:

Code:- move(X,Y,_):-X=:=2,Y=:=0,
write('done'),
!. move(X,Y,Z):-X<4,
\+member((4,Y),Z),write("fill 4 jug"),nl,move(4,Y,[(4,Y)|Z]).
/*jug 4 filled*/ move(X,Y,Z):-Y<3,
\+member((X,3),Z),write("fill 3 jug"),
nl,move(X,3,[(X,3)|z]). /*jug 3 filled*/ move(X,Y,Z):-
X>0,\+member((0,Y),Z),write("pour 4 jug"),
nl,move(0,Y,[(0,Y)|Z]). /*jug 4 empty*/ move(X,Y,Z):-
Y>0,\+member((X,0),Z),write("pour 3 jug"),
nl,move(X,0,[(X,0)|Z]). /*jug 3 empty*/ move(X,Y,Z):-P
is X+Y,P>=4,Y>0,
K is 4-X,M is Y- K,
\+member((4,M),Z),
write("pour from 3jug to 4jug"),
nl,move(4,M,[(4,M)|Z]). /* 3L
is poured to 4L jug*/
move(X,Y,Z):-P is X+Y,
P>=3,X>0,K is 3-Y,M is X-K,
\+member((M,3),Z),
write("pour from 4jug to 3jug"),
nl,move(M,3,[(M,3)|Z]).
move(X,Y,Z):-K is X+Y,
K<4,Y>0,\+member((K,0),Z),
write("pour from 3jug to 4jug"),
nl,move(K,0,[(K,0)|Z]).
move(X,Y,Z):-K is X+Y,
K<3,X>0,\+member((0,K),Z),
write("pour from 4jug to 3jug"),
nl,move(0,K,[(0,K)|Z]).

O/P:-


SWI-Prolog (AMD64, Multi-threaded, versio...
File Edit Settings Run Debug Help
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. T
his is free software.
Please run ?- license. for legal details.
For online help and background, visit https://w
ww.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- ap
ropos(Word).
?- move(0,0,0)
fill 4 jug
fill 3 jug
pour 4 jug
pour 3 jug
fill 4 jug
pour from 4jug to 3jug
pour 3 jug
pour from 4jug to 3jug
fill 4 jug
pour from 4jug to 3jug
pour 3 jug
done
true

1.2 Tic-Tac-Toe Problem:

A board game (such as tic-tac-toe) is usually programmed as a state machine. Looking on the current-state and therefore the player's move, the game goes into the next-state.

tit-tat-toe (or Noughts and crosses, Xs and Os) could be a paper and pencil for 2 players, X and O, who take turns marking the areas in an exceedingly 3×3 grid.

The player who succeeds in putting 3 individual marks in an exceedingly horizontal, vertical or diagonal row wins the game. Players shortly discover that best play from each party ends up in a draw.

The game is generalized to an m,n,k-game during which 2 players alternate putting stones of their own colour on an m×n board, with the goal of obtaining k of their own colour in a row. Tit-Tat-Toe is the (3,3,3)-game.

Code :- play :- my_turn([]).

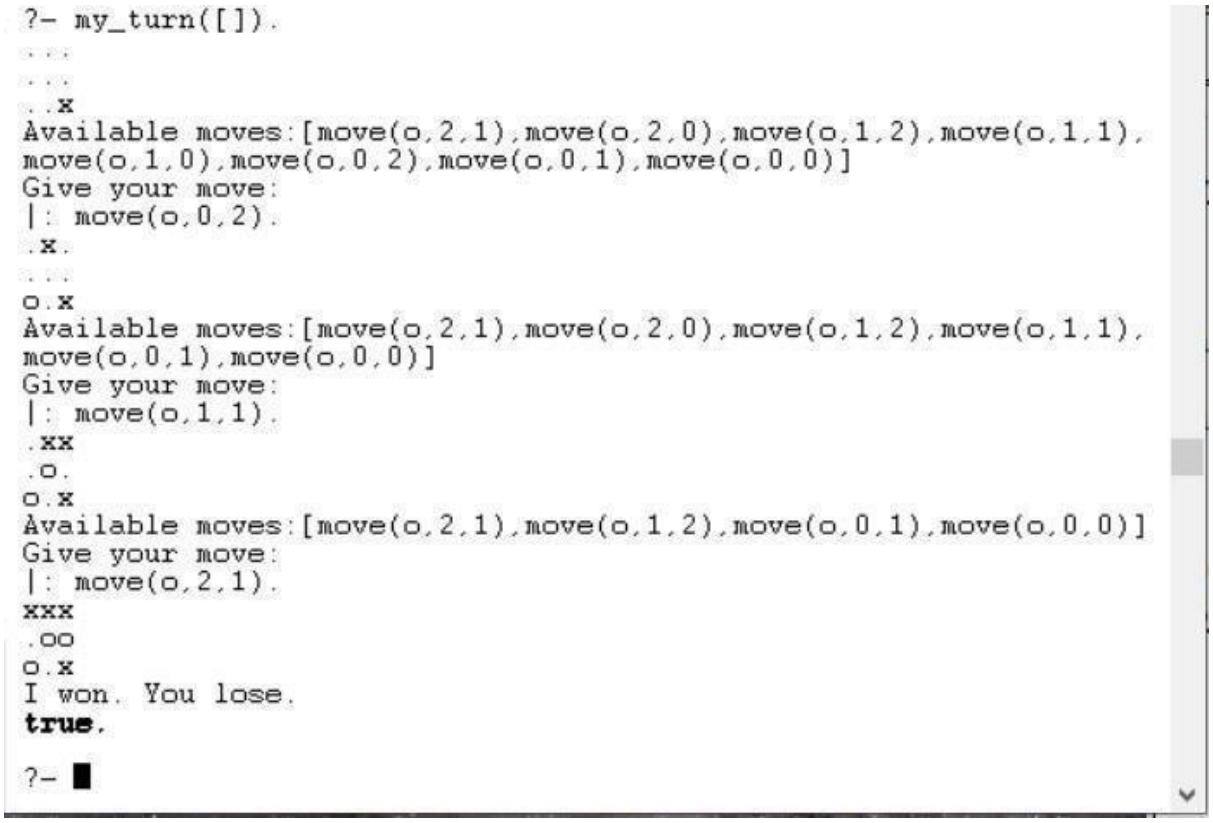
```
my_turn(Game) :-
valid_moves(ValidMoves, Game, x),
any_valid_moves(ValidMoves, Game).
any_valid_moves([], _) :-  write('It
is a tie'), nl.
any_valid_moves([_|_], Game) :-  findall(NextMove,
game_analysis(x, Game, NextMove), MyMoves),
do_a_decision(MyMoves, Game).
% This can only fail in the beginning. do_a_decision(MyMoves, Game) :-

not(MyMoves = []), length(MyMoves,  MaxMove),
random(0, MaxMove,
ChosenMove), nth0(ChosenMove,
MyMoves, X), NextGame = [X |
Game], print_game(NextGame),
(victory_condition(x, NextGame) ->  (write('I
won. You lose.'), nl);  your_turn(NextGame),
!).
your_turn(Game) :-  valid_moves(ValidMoves,
Game, o),
(ValidMoves = [] -> (write('It is a tie'), nl);
(write('Available moves:'), write(ValidMoves), nl, ask_move(Y, ValidMoves),
NextGame = [Y | Game],
(victory_condition(o, NextGame) ->
(write('I lose. You win.'), nl);
my_turn(NextGame), !))).  ask_move(Move,
ValidMoves) :-  write('Give your move:'), nl,
read(Move), member(Move, ValidMoves), !.
```

```
ask_move(Y, ValidMoves) :-  write('not
a move'), nl,  ask_move(Y,
ValidMoves).

movement_prompt(X, Y, ValidMoves) :-
write('Give your X:'), nl, read(X), member(move(o, X, Y), ValidMoves), !, write('Give your
Y:'), nl, read(Y), member(move(o, X, Y),  ValidMoves).
% A routine for printing games.. Well you can use it.

print_game(Game) :- plot_row(0, Game), plot_row(1, Game), plot_row(2, Game).
plot_row(Y, Game) :- plot(Game, 0, Y), plot(Game, 1, Y), plot(Game, 2, Y), nl.
plot(Game, X, Y) :-
(member(move(P, X, Y), Game), ground(P)) -> write(P) ; write('.').
% This system determines whether there's a perfect play available.

game_analysis(_, Game, _) :- victory_condition(Winner, Game), Winner = x. % We do not
want to lose.
% Winner = o. % We do not want to win. (egostroking mode).

% true. % If you remove this constraint entirely, it may let you win.

game_analysis(Turn, Game, NextMove) :- not(victory_condition(_, Game)),
game_analysis_continue(Turn, Game, NextMove).
game_analysis_continue(Turn, Game, NextMove) :-  valid_moves(Moves,
Game, Turn),  game_analysis_search(Moves, Turn, Game, NextMove).
% Comment these away and the system refuses to play,

% because there are no ways to play this without a possibility of tie.
game_analysis_search([], o, _, _). % Tie on opponent's turn. game_analysis_search([], x, _,
_). % Tie on our turn. game_analysis_search([X|Z], o, Game, NextMove) :- % Whatever
opponent does,

NextGame = [X | Game], % we desire not to lose.  game_analysis_search(Z,
o, Game, NextMove), game_analysis(x,  NextGame, _), !.
```

```
game_analysis_search(Moves, x, Game, NextMove) :-game_analysis_search_x(Moves,
Game, NextMove).  game_analysis_search_x([X|_], Game, X) :-
NextGame = [X | Game], game_analysis(o, NextGame, _). game_analysis_search_x([_|Z],
Game, NextMove) :- game_analysis_search_x(Z, Game, NextMove).
% This thing describes all kinds of valid games. valid_game(Turn, Game, LastGame, Result)
:-

victory_condition(Winner, Game) -> (Game =

LastGame, Result = win(Winner)) ;  valid_continuing_game(Turn,
Game, LastGame, Result).  valid_continuing_game(Turn, Game,
LastGame, Result) :-
valid_moves(Moves, Game, Turn), tie_or_next_game(Moves, Turn, Game, LastGame,
Result).
tie_or_next_game([], _, Game, Game, tie).
tie_or_next_game(Moves, Turn, Game, LastGame, Result) :-valid_gameplay_move(Moves,
NextGame, Game),  opponent(Turn, NextTurn), valid_game(NextTurn, NextGame,
LastGame, Result).  % Victory conditions for tic tac toe. victory(P, Game, Begin) :-
valid_gameplay(Game, Begin), victory_condition(P, Game). victory_condition(P, Game) :-

(X=0;X=1;X=2), member(move(P, X, 0), Game), member(move(P, X, 1), Game),
member(move(P, X, 2), Game). victory_condition(P, Game) :-(Y=0;Y=1;Y=2),
member(move(P, 0, Y), Game), member(move(P, 1, Y), Game), member(move(P, 2, Y),
Game). victory_condition(P, Game) :-member(move(P, 0, 2), Game), member(move(P, 1, 1),
Game), member(move(P, 2, 0), Game).
victory_condition(P, Game) :-  member(move(P,
0, 0), Game),  member(move(P, 1, 1), Game),
member(move(P, 2, 2), Game).
% This describes a valid form of gameplay. % Which player did the move is disregarded.
valid_gameplay(Start, Start).  valid_gameplay(Game, Start) :-
valid_gameplay(PreviousGame, Start),  valid_moves(Moves,
PreviousGame, _),  valid_gameplay_move(Moves, Game,
PreviousGame).
```

```
valid_gameplay_move([X|_], [X|PreviousGame],  PreviousGame).
valid_gameplay_move([_|Z], Game, PreviousGame) :-valid_gameplay_move(Z, Game,
PreviousGame).
% The set of valid moves must not be affected by the decision making

% of the prolog interpreter.

% Therefore we have to retrieve them like this.
% This is equivalent to the (∀x∈0..2)(∀y∈0..2)(....

% uh wait.. There's no way to represent this using those quantifiers. valid_moves(Moves,
Game, Turn) :-
valid_moves_column(0, M1, [], Game, Turn), valid_moves_column(1, M2, M1, Game, Turn),
valid_moves_column(2, Moves, M2, Game, Turn). valid_moves_column(X, M3, M0,
Game, Turn) :-valid_moves_cell(X, 0, M1, M0, Game, Turn), valid_moves_cell(X, 1, M2,
M1, Game, Turn), valid_moves_cell(X, 2, M3, M2, Game, Turn). valid_moves_cell(X, Y,
M1, M0, Game, Turn) :-Game) -> M0 = M1 ; M1 = [move(Turn,X,Y) | M0].
member(move(_, X, Y),  %
valid_move(X, Y, Game) :-
% (X=0;X=1;X=2),
% (Y=0;Y=1;Y=2),
% not(member(move(_, X, Y), Game)).
opponent(x,
o). opponent(o,
x).
```

O/P :-


?- my_turn([]).
...
...
..x
Available moves:[move(o,2,1),move(o,2,0),move(o,1,2),move(o,1,1),
move(o,1,0),move(o,0,2),move(o,0,1),move(o,0,0)]
Give your move:
|: move(o,0,2).
.x.
...
o.x
Available moves:[move(o,2,1),move(o,2,0),move(o,1,2),move(o,1,1),
move(o,0,1),move(o,0,0)]
Give your move:
|: move(o,1,1).
.xx
.o.
o.x
Available moves:[move(o,2,1),move(o,1,2),move(o,0,1),move(o,0,0)]
Give your move:
|: move(o,2,1).
xxx
.oo
o.x
I won. You lose.
true.
?-

1.3 8-Puzzle Problem:

/* This predicate initialises the problem states. The first argument of solve/3 is the initial state, the 2nd the goal state, and the third the plan that will be produced. */ **Code:-**

```
test(Plan):- write('Initial state:'),nl,
Init= [at(tile4,1), at(tile3,2), at(tile8,3), at(empty,4), at(tile2,5), at(tile6,6), at(tile5,7),
at(tile1,8), at(tile7,9)], write_sol(Init),
Goal= [at(tile1,1), at(tile2,2), at(tile3,3), at(tile4,4), at(empty,5), at(tile5,6), at(tile6,7),
at(tile7,8), at(tile8,9)], nl,write('Goal state:'),nl, write(Goal),nl,nl,
solve(Init,Goal,Plan).
solve(State, Goal, Plan):- solve(State,
Goal, [], Plan).
/*Determines whether Current and Destination tiles are a valid move. */
is_movable(X1,Y1) :- (1 is X1 - Y1) ; (-1 is X1 - Y1) ; (3 is X1 - Y1) ; (-3 is X1 - Y1).
/*This predicate produces the plan. Once the Goal list is a subset of the current State the plan
is complete and it is written to the screen using write_sol */ solve(State, Goal, Plan, Plan):-
is_subset(Goal, State), nl, write_sol(Plan).
solve(State, Goal, Sofar, Plan):- act(Action,
Preconditions, Delete, Add),
is_subset(Preconditions, State), \+
member(Action, Sofar), delete_list(Delete,
State, Remainder), append(Add,
Remainder, NewState), solve(NewState,
Goal, [Action|Sofar], Plan).
act(move(X,Y,Z),
[at(X,Y), at(empty,Z), is_movable(Y,Z)],
[at(X,Y), at(empty,Z)],
[at(X,Z), at(empty,Y)]).
/*Check is first list is a subset of the second */
is_subset([H|T], Set):- member(H, Set),
is_subset(T, Set). is_subset([], _).
```

```
/* Remove all elements of 1st list from second to create third. */
delete_list([H|T], Curstate, Newstate):- remove(H, Curstate,
Remainder), delete_list(T, Remainder, Newstate).
delete_list([], Curstate, Curstate). remove(X, [X|T], T).
remove(X, [H|T], [H|R]):- remove(X, T, R). write_sol([]).
write_sol([H|T]):- write_sol(T), write(H), nl.
append([H|T], L1, [H|L2]):-
append(T, L1, L2).
append([], L, L). member(X,
[X|_]). member(X, [_|T]):-
member(X, T).
```

Output :-

```
1 2 3
5 6 0
7 8 4

1 2 3
5 0 6
7 8 4

1 2 3
```

5 8 6
7 0 4

1 2 3
5 8 6
0 7 4 **Practical No.2**

Aim :-Introduction to Python Programming: Learn the different libraries 2.1

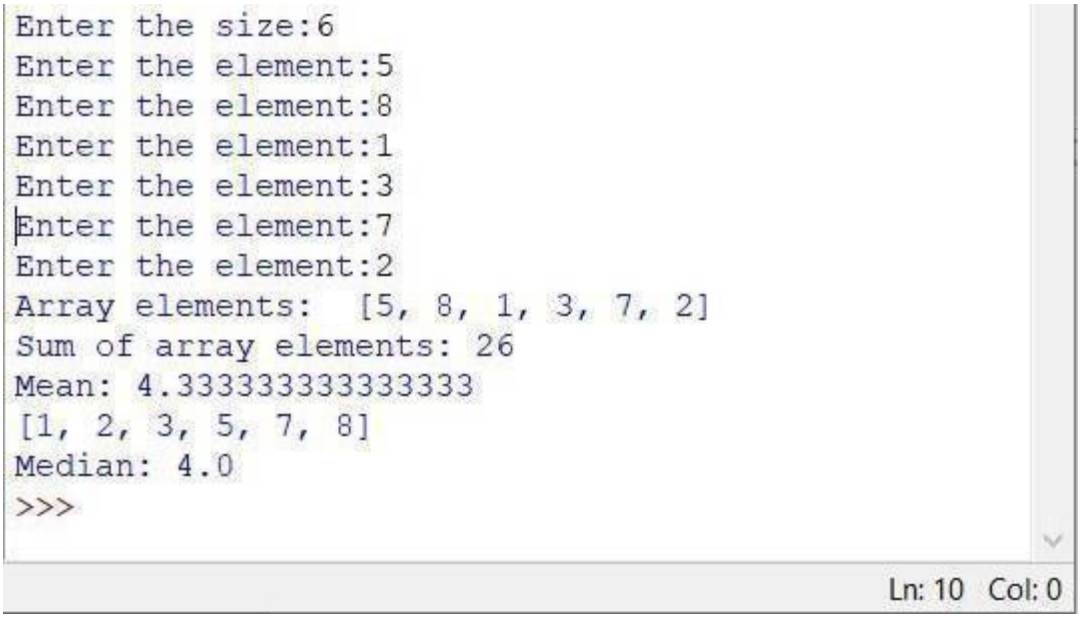
Numpy

- Python library is nothing but a ready made moule.
- This library can be used whenever we want.
- If we are writing a code and if a particular requirement arises then instead of sitting and writing the whole code we can just use the ready made code available in the library.
- Thus by using the library our time is getting saved in a very wonderful manner.
- We can relate the Python library with the real world book library too. So if you imagine a book library it has a whole set of books with it. We can choose the book according to our requirements. Similarly in the python library we can choose a particular set of code which is needed.
- The extension of library files are ".dll"
- Full form of dll is Dynamic Load Libraries
- So whenever we add a library in our program during the execution phase it searches it and loads the particular module which is needed.
- Now in this module we are studying about numpy which is one of the libraries in python.
- NumPy stands for Numerical Python.
- It is one of the most widely used libray.
- As it contains the code related to numerical details it is most popular around data science and machine learning as both these fields need a lot of numerical logic getting applied in it.
- It is used whenever the situation in coding arises in working with an array.
- It does have methods that is made up for algebra related logics.
- This Numpy was made in the year 2005

Code :-

```
import numpy  arr=[]
s=int(input("Enter the size:"))  for i in
range(0,s):  elem=int(input("Enter the
element:"))  arr.append(elem)
print("Array elements: ",arr)
#mean=sum/number of elements arr_
sum=sum(arr)  print("Sum of array
elements:",arr_sum)
mean_arr=arr_sum/s
print("Mean:",mean_arr)  #median
arr.sort()  print(arr)  if s%2==0:
        m1=arr[s//2]
m2=arr[s//2-1]
med=(m1+m2)/2  else:
        med=arr[s//2]
        print("Median:",med )
```

O/P :-


Enter the size:6
Enter the element:5
Enter the element:8
Enter the element:1
Enter the element:3
Enter the element:7
Enter the element:2
Array elements: [5, 8, 1, 3, 7, 2]
Sum of array elements: 26
Mean: 4.333333333333333
[1, 2, 3, 5, 7, 8]
Median: 4.0
>>>
Ln: 10 Col: 0

2.2 Pandas

- The main role of the pandas library is to analyze the data.
- It is open source in nature
- It is used in relational data
- On the top of Numpy library Pandas library is present.
- It is very quick in nature. ● It was made in the year 2008
- It is very efficient in datas.
- When it comes to pandas it is not necessary that the data should or should belong to a kind of category but instead it allows many.
- By using pandas you can reshape, analyze, and change your data very easily.

1. Series:

- It is an array.
- It can hold any kind of data types like integer, float, character etc.
- It points to the column.

2. Data Frame:

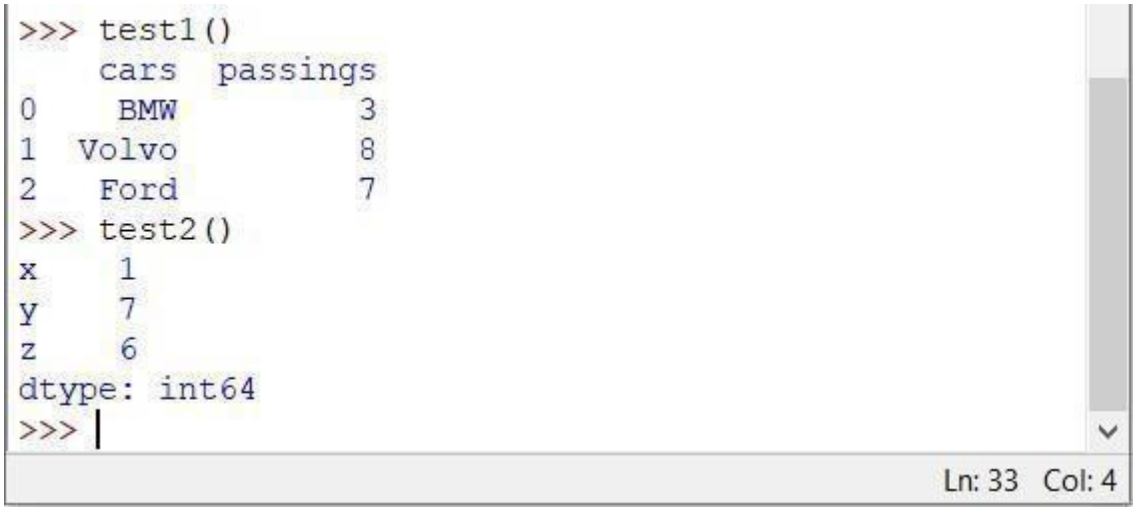
It handles 3 parts, mainly data, columns and rows.

Code :- import

pandas as pd

```
def test1():
        mydataset={'cars':["BMW","Volvo","Ford"],'passings':[3,8,7]}
        myvar=pd.DataFrame(mydataset)
        print(myvar)
def test2():
        a=[1,7,6]
        myvar2=pd.Series(a,index=["x","y","z"])  print(myvar2)
```

O/P :-



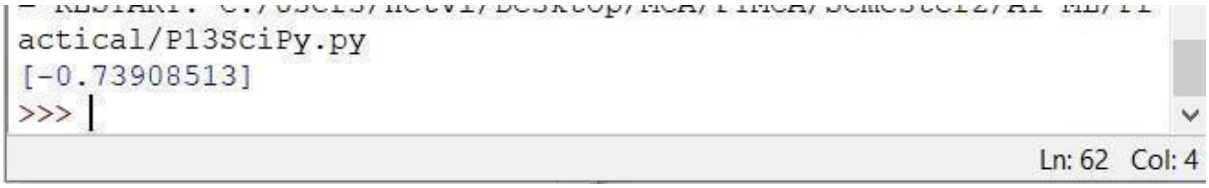
2.3 Scipy

It falls under NumPy:

- It uses scientific and mathematical logic.
- It makes the python very effective as it allows user interaction too.
- It stands for “Scientific Python”
- It is open source
- Manipulating N-dimension array is done through SciPy.
- Some sub packages of SciPy are as follows:
- **Scipy.clusetr:** K mean algorithm and such similar algorithms can be done using this library.
- **Scipy.io:** Inputs and outputs are handled here

Code :- from scipy.optimize
import root from math import
cos def eqn(x):
    return x + cos(x)
myroot = root(eqn, 0) print(myroot.x)

O/P:-



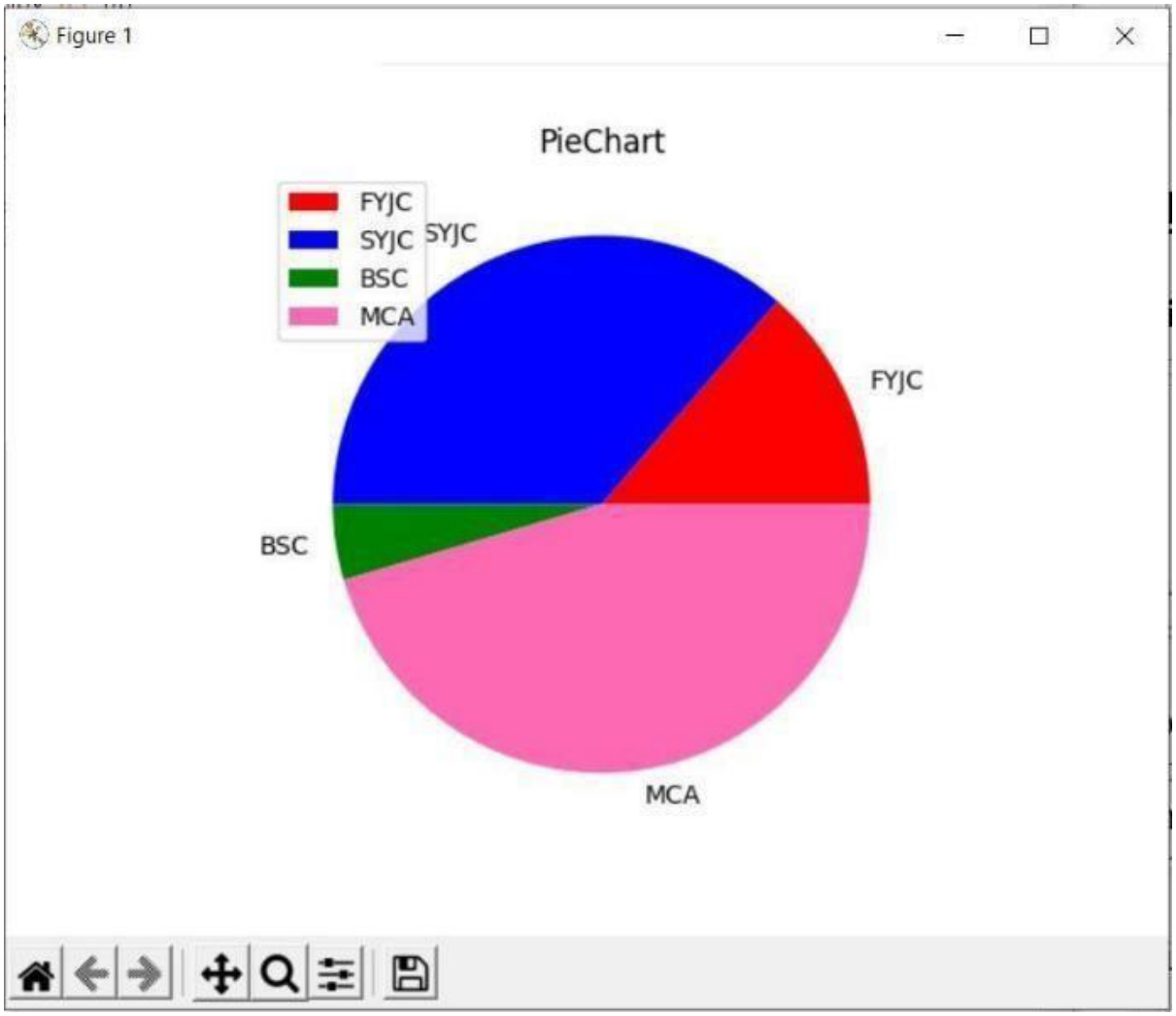
2.4 Matplotlib

It is used to plot graphs

Code :-

```
import matplotlib.pyplot as plt  import
numpy as np
x=np.array(["FYJC", "SYJC", "BSC", "MCA"])
y=np.array([3,8,1,10])
mycolors=np.array(["red", "blue", "green", "hotpink"])  plt.pie(y,labels=x,colors=mycolors)
plt.title("PieChart") plt.legend() plt.show()
```

O/P :-



2.5 Scikit Learn

● It is mainly used in machine learning

● It has lot of statistics related tools ● It is open source.

● By using the Scikit library the efficiency will improve tremendously as it is quite accurate.

- It is very useful in algorithms which are very famous in machine learning like K-mean, Knearest, clustering etc.
- It is available to everybody so any programmer if he or she feels like utilizing it then can use it.
- Scikit requires Numpy
- Installation of scikit is must to make the program run, this can be done in the following manner.

pip install -U scikit-learn

Code:-

```
from sklearn.datasets import load_iris
iris = load_iris()  A= iris.data  y =
iris.target feature_names =
iris.feature_names  target_names =
iris.target_names  print("Feature
names:", feature_names)  print("Target
names:", target_names)  print("\nFirst
10 rows of A:\n", A[:10]) O/P:-
```

Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

Target names: ['setosa' 'versicolor' 'virginica'] First 10 rows of X:

=

[

[5.1 3.5 1.4 0.2]

[4.9 3. 1.4 0.2]

[4.7 3.2 1.3 0.2]

[4.6 3.1 1.5 0.2]

[5. 3.6 1.4 0.2]

[5.4 3.9 1.7 0.4]

[4.6 3.4 1.4 0.3]

[5. 3.4 1.5 0.2]

[4.4 2.9 1.4 0.2]

[4.9 3.1 1.5 0.1]

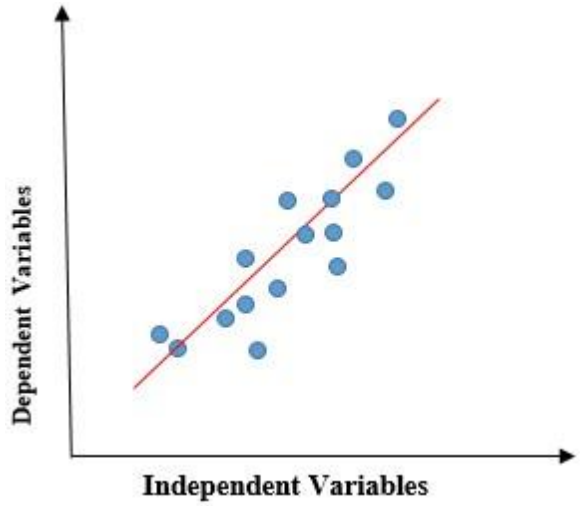
]

Practical No.3 Aim

:- Implementation of Algorithm.

3.1 Linear Regression

Linear regression is a quiet and simple statistical regression method used for predictive analysis and shows the relationship between the continuous variables. Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Yaxis), consequently called linear regression. If there is a single input variable (x), such linear regression is called **simple linear regression**. And if there is more than one input variable, such linear regression is called **multiple linear regression.** The linear regression model gives a sloped straight line describing the relationship within the variables.



The above graph presents the linear relationship between the dependent variable and independent variables. When the value of x (**independent variable**) increases, the value of y (**dependent variable**) is likewise increasing. The red line is referred to as the best fit straight line. Based on the given data points, we try to plot a line that models the points the best.

To calculate best-fit line linear regression uses a traditional slope-intercept form.

$$y = mx+b \quad \Longrightarrow \quad y = a_0+a_1x$$

y= Dependent Variable. x= Independent Variable. a0= intercept of the line. a1 = Linear regression coefficient.

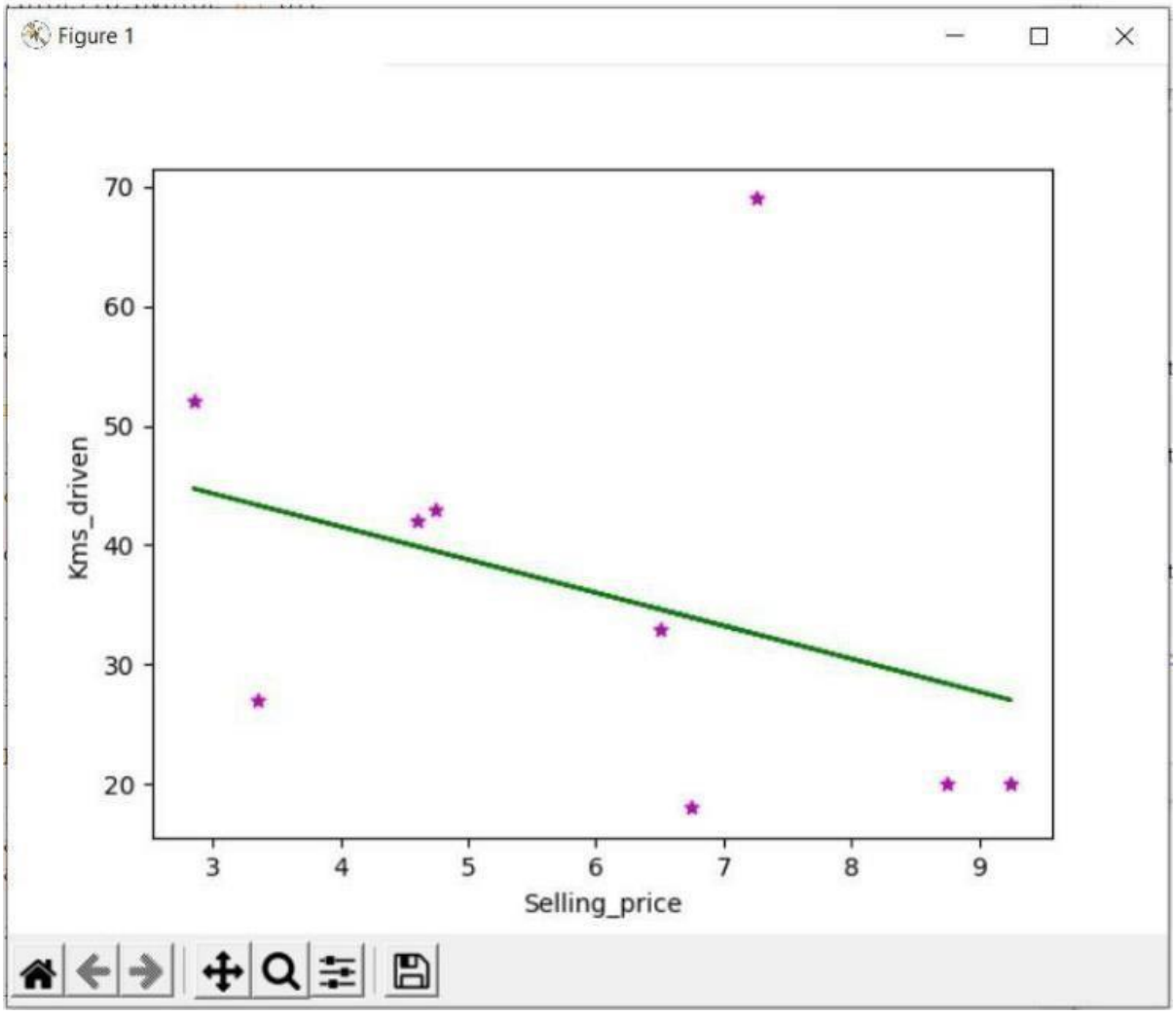
Code :- import numpy as np import matplotlib.pyplot as plt def estimate_coef(x,y):

```
n=np.size(x)                    mean_x=np.mean(x)
mean_y=np.mean(y)               ss_xy=np.sum(y*x)-
n*mean_y*mean_x                 ss_xx=np.sum(x*x)-
n*mean_x*mean_x
        b1=ss_xy/ss_xx  b0=mean_y-
        b1*mean_x
        return(b0, b1)
def plotting_regLine(x,y,b):  plt.scatter(x,y,color="m",
        marker="*",s=30)
        y_pred=b[0]+b[1]*x
        plt.plot(x,y_pred,color="g")
        plt.xlabel("Selling_price")
        plt.ylabel("Kms_driven")
        plt.show()
def main():
        x=np.array([3.35,4.75,7.25,2.85,4.6,9.25,6.75,6.5,8.75])
        y=np.array([27,43,69,52,42,20,18,33,20])
        b=estimate_coef(x,y)
print("Estimated Coefficients:\nb0={}\nb1={}".format(b[0],b[1])) plotting_regLine(x,y,b)
```

O/P :-

```
>>> main()
Estimated Coefficients:
b0=52.63110752498607
b1=-2.769287099442639
>>>
```

Ln: 68 Col: 4



3.2 Logistic regression

Logistic regression is a supervised learning algorithm that outputs values between zero and one.

Hypothesis: The objective of a logistic regression is to learn a function that outputs the probability that the dependent variable is one for each training sample. To achieve that, a sigmoid / logistic function is required for the transformation.

A sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

This hypothesis is typically represented by the following function:

$$\mathrm{h}_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Where,

θ is a vector of parameters that corresponds to each independent variable

x is a vector of independent variables

COST FUNCTION

The cost function for logistic regression is derived from statistics using the principle of maximum likelyhood estimation, which allows efficient identification of parameters. In addition the covex property of the cost function allow gradient descent to work effectively.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$$

$$(h_\theta(x^i) - y^i)^2 = \begin{cases} -log(h_\theta(x^i)) \text{ if } y^i = 1 \\ -log(1 - h_\theta(x^i)) \text{ if } y^i = 0 \end{cases}$$

Where,

- i is one of the mth training samples
- hΘ(xi) is the predicted value for the training sample

● yi is the actual value for the training sample

Code :- import pandas as pd import
numpy as np import matplotlib.pyplot
as plt dataset =
pd.read_csv('User_Data.csv')
#Now, to predict whether a user will purchase the product or not, #one
needs to find out the relationship between Age and Estimated Salary.
#Here User ID and Gender are not important factors for finding out this.
input x = dataset.iloc[:, [2,
3]].values
output y = dataset.iloc[:,
4].values
#Splitting the dataset to train and test.
#75% of data is used for training the model and 25% of it #is used to test the performance of our model.
#understand train_test_split from
sklearn.model_selection import train_test_split xtrain,
xtest, ytrain, ytest = train_test_split(x, y, test_size =
0.25, random_state = 0)
#Now, it is very important to perform feature scaling here #because Age and Estimated Salary values lie in different ranges.
#If we don't scale the features then Estimated Salary feature will #dominate Age feature when the model finds the nearest neighbor to a data point #in data space.
#Here once see that Age and Estimated salary features values are sacled #and now there in the -1 to 1. Hence, each feature will contribute #equally in decision making i.e. finalizing the hypothesis.
from sklearn.preprocessing import StandardScaler sc_x = StandardScaler() xtrain =
sc_x.fit_transform(xtrain) xtest = sc_x.transform(xtest) print (xtrain[0:10, :])

#Finally, we are training our Logistic Regression model. from sklearn.linear_model import LogisticRegression classifier = LogisticRegression(random_state = 0) classifier.fit(xtrain, ytrain)

#After training the model, it time to use it to do prediction on testing data. y_pred = classifier.predict(xtest)

#Let's test the performance of our model – Confusion Matrix from sklearn.metrics import confusion_matrix cm = confusion_matrix(ytest, y_pred) print ("Confusion Matrix : \n", cm) from sklearn.metrics import accuracy_score print ("Accuracy : ", accuracy_score(ytest, y_pred)) #Visualizing the performance of our model. from matplotlib.colors import ListedColormap X_set, y_set = xtest, ytest

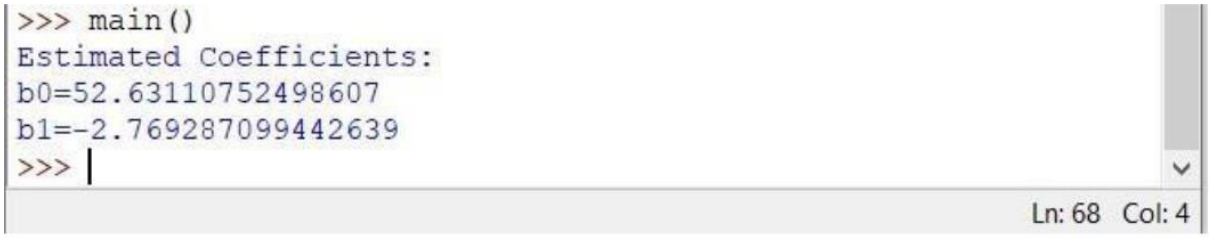
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01), np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01)) plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green'))) plt.xlim(X1.min(), X1.max())

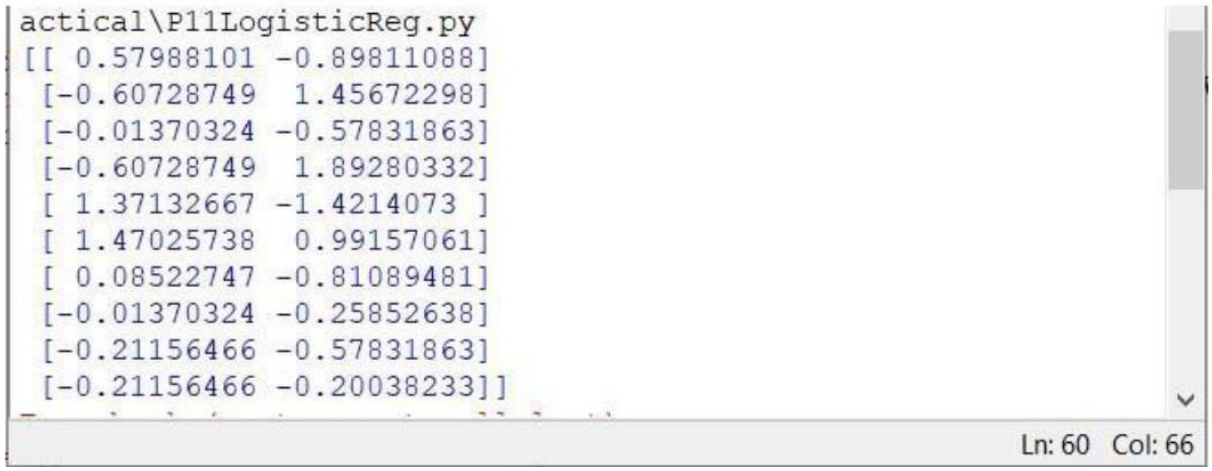
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):

plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j) plt.title('Classifier (Test set)') plt.xlabel('Age') plt.ylabel('Estimated Salary') plt.legend() plt.show()

O/P :-


>>> main()
Estimated Coefficients:
b0=52.63110752498607
b1=-2.769287099442639
>>>
Ln: 68 Col: 4


actical\P11LogisticReg.py
[[0.57988101 -0.89811088]
[-0.60728749 1.45672298]
[-0.01370324 -0.57831863]
[-0.60728749 1.89280332]
[1.37132667 -1.4214073]
[1.47025738 0.99157061]
[0.08522747 -0.81089481]
[-0.01370324 -0.25852638]
[-0.21156466 -0.57831863]
[-0.21156466 -0.20038233]]
Ln: 60 Col: 66

3.3 KNN – Classifier:

1. Evaluation procedure 1 - Train and test on the entire dataset

1. Train the model on the **entire dataset**.
2. Test the model on the **same dataset**, and evaluate how well we did by comparing the **predicted** response values with the **true** response values.

KNN model:

1. Pick a value for K.
2. Search for the K observations in the training data that are "nearest" to the measurements of the unknown iris
3. Use the most popular response value from the K nearest neighbors as the predicted response value for the unknown iris

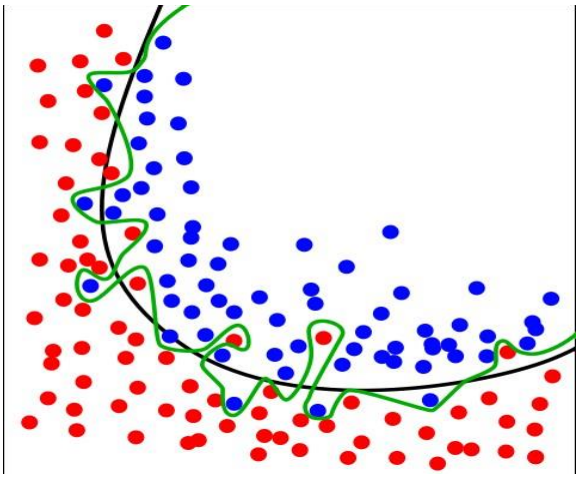
This would always have 100% accuracy, because we are testing on the exact same data, it would always make correct predictions

KNN would search for one nearest observation and find that exact same observation
KNN has memorized the training set

Because we testing on the exact same data, it would always make the same prediction **1d.**

Problems with training and testing on the same data:

- Goal is to estimate likely performance of a model on **out-of-sample data**
- But, maximizing training accuracy rewards **overly complex models** that won't necessarily generalize
- Unnecessarily complex models **overfit** the training data



Code :-

```
import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  path
= "https://archive.ics.uci.edu/ml/machine-learningdatabases/iris/iris.data"
headernames = ['sepal-length', 'sepal-width', 'petal-length', 'petalwidth', 'Class']
dataset = pd.read_csv(path, names = headernames)  dataset.head()
X = dataset.iloc[:, :-1].values  y
= dataset.iloc[:, 4].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40)
from sklearn.preprocessing import StandardScaler  scaler =
StandardScaler()  scaler.fit(X_train)
X_train = scaler.transform(X_train) X_test = scaler.transform(X_test)  from
sklearn.neighbors import KNeighborsClassifier  classifier =
KNeighborsClassifier(n_neighbors = 8)  classifier.fit(X_train, y_train)  y_pred =
classifier.predict(X_test)  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)
```

O/P :-

```
Confusion Matrix:
[[17  0  0]
 [ 0 19  0]
 [ 0  3 21]]
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        17
Iris-versicolor       0.86      1.00      0.93        19
 Iris-virginica       1.00      0.88      0.93        24

       accuracy                           0.95        60
      macro avg       0.95      0.96      0.95        60
   weighted avg       0.96      0.95      0.95        60

Accuracy: 0.95
>>>
```

Ln: 122 Col: 4

Practical No.4

AIM: Implementation of dimensionality reduction techniques

4.1 Features Extraction and Selection

DIMENSIONALITY REDUCTION

Dimensionality reduction eliminates some features of the dataset and creates a restricted set of features that contains all of the information needed to predict the target variables more efficiently and accurately.

Reducing the number of features normally also reduces the output variability and complexity of the learning process. The covariance matrix is an important step in the dimensionality reduction process. It is a critical process to check the correlation between different features.

Correlation and its Measurement:

There is a concept of correlation in machine learning that is called multicollinearity. Multicollinearity exists when one or more independent variables highly correlate with each other. Multicollinearity makes variables highly correlated to one another, which makes the variables' coefficients highly unstable.

The coefficient is a significant part of regression, and if this is unstable, then there will be a poor outcome of the regression result. Multicollinearity is confirmed by using Variance Inflation Factors (VIF). Therefore, if multicollinearity is suspected, it can be checked using the variance inflation factor (VIF).

$$VIF = 1/(1-R^2)$$

Feature Extraction:

In feature extraction, a set of new features are found. That is found through some mapping from the existing features. Moreover, mapping can be either linear or non-linear.

Linear Feature Extraction:

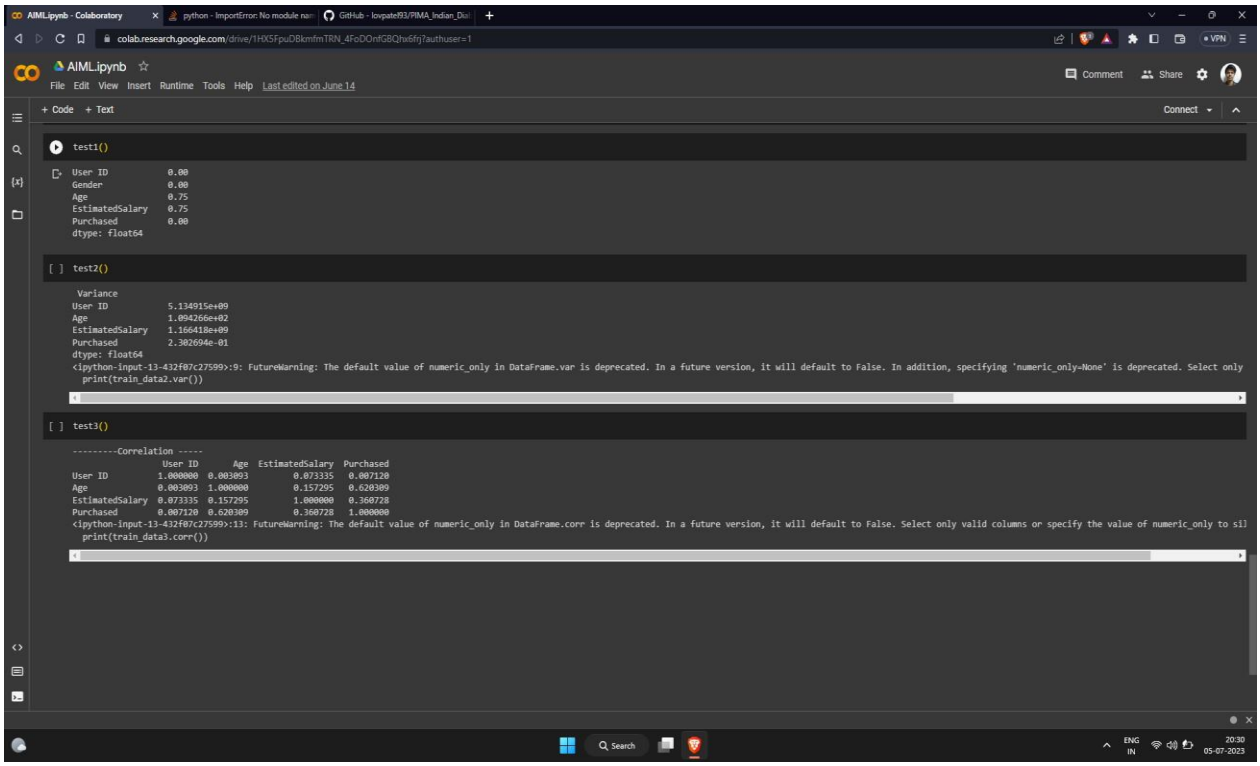
Linear feature extraction is straightforward to compute and analytically traceable.

Widespread linear feature extraction methods:

- **Principal Component Analysis (PCA)**: It seeks a projection that preserves as much information as possible in the data.
- **Linear Discriminant Analysis (LDA)**:- It seeks a projection that best discriminates the data.

Code :- import
pandas as pd import
numpy as np def
test1():
train_data=pd.read_csv("/content/User_Data.csv")
print(train_data.isnull().sum()/len(train_data)*100) def
test2():
train_data2=pd.read_csv("User_Data.csv")
print(" Variance ") print(train_data2.var())
def test3():
train_data3=pd.read_csv("User_Data.csv") print("----
-----Correlation ----- ") print(train_data3.corr())

O/P :-


AIML.ipynb - Colaboratory
colab.research.google.com/drive/1HX5FpuDBkmfmTRN_4FoDOnfGBQhx6frj?authuser=1
AIML.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 14
Comment
Share
+ Code + Text
Connect
test1()
User ID 0.00
Gender 0.00
Age 0.75
EstimatedSalary 0.75
Purchased 0.00
dtype: float64
test2()
Variance
User ID 5.134915e+09
Age 1.094266e+02
EstimatedSalary 1.166418e+09
Purchased 2.302694e-01
dtype: float64
<ipython-input-13-432f07c27599>:9: FutureWarning: The default value of numeric_only in DataFrame.var is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is deprecated. Select only
print(train_data2.var())
test3()
---------Correlation -----
User ID Age EstimatedSalary Purchased
User ID 1.000000 0.003093 0.073335 0.007120
Age 0.003093 1.000000 0.157295 0.620309
EstimatedSalary 0.073335 0.157295 1.000000 0.360728
Purchased 0.007120 0.620309 0.360728 1.000000
<ipython-input-13-432f07c27599>:13: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to sil
print(train_data3.corr())
Search
ENG
IN
20:30
05-07-2023

4.2 Normalisation:

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Normalization is also required for some algorithms to model the data correctly.

For example, assume your input dataset contains one column with values ranging from 0 to 1, and another column with values ranging from 10,000 to 100,000. The great difference in the *scale* of the numbers could cause problems when you attempt to combine the values as features during modelling.

Normalization avoids these problems by creating new values that maintain the general distribution and ratios in the source data, while keeping values within a scale applied across all numeric columns used in the model.

This component offers several options for transforming numeric data:

- You can change all values to a 0-1 scale, or transform the values by representing them as percentile ranks rather than absolute values.
- You can apply normalization to a single column, or to multiple columns in the same dataset. If you need to repeat the pipeline, or apply the same normalization steps to other data, you can save the steps as a normalization transform, and apply it to other datasets that have the same schema.

Normalization Techniques at a Glance:

Four common normalization techniques may be useful:

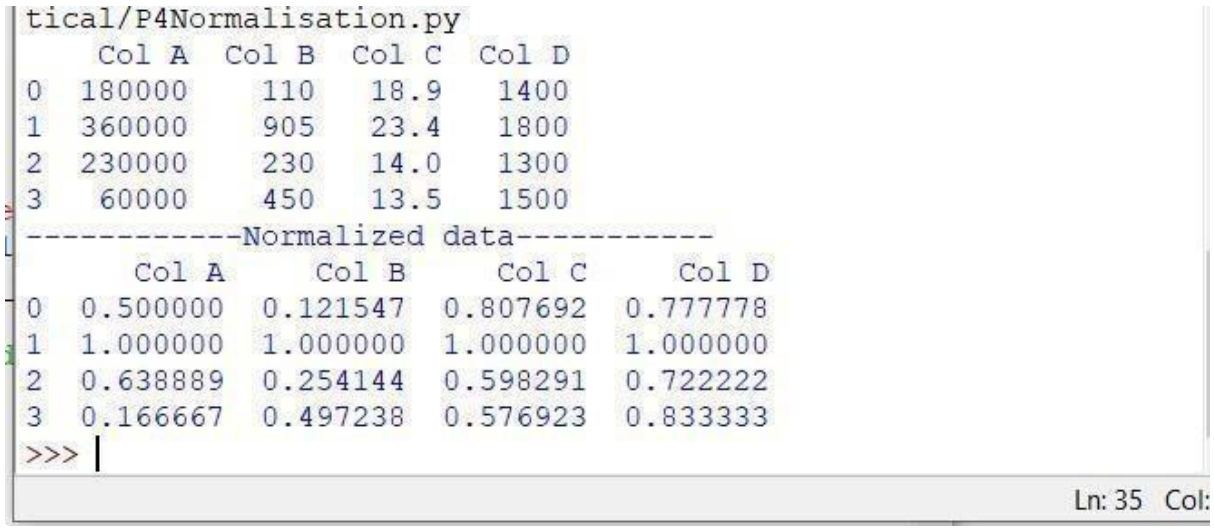
- scaling to a range
- clipping
- log scaling
- z-score

The following charts show the effect of each normalization technique on the distribution of the raw feature (price) on the left. The charts are based on the data set from 1985 Ward's Automotive Yearbook that is part of the UCI Machine Learning Repository under Automobile Data Set.

Code :-

```
# importing packages  import
pandas as pd  import
matplotlib.pyplot as plt
# create data df =
pd.DataFrame([ [180000,
110, 18.9, 1400],
[360000, 905, 23.4, 1800],
[230000, 230, 14.0, 1300],  [60000, 450,
13.5, 1500]],  columns=['Col A', 'Col B',
'Col C', 'Col D'])  print(df)  # copy the data
df_max_scaled = df.copy()
# apply normalization techniques for column in  df_max_scaled.columns:
df_max_scaled[column] = df_max_scaled[column] / df_max_scaled[column].abs().max()
print(" Normalized data ")  # view normalized data  print(df_max_scaled)
```

O/P:-



```
tical/P4Normalisation.py
    Col A  Col B  Col C  Col D
0  180000    110   18.9   1400
1  360000    905   23.4   1800
2  230000    230   14.0   1300
3   60000    450   13.5   1500
------------Normalized data-----------
      Col A     Col B     Col C     Col D
0  0.500000  0.121547  0.807692  0.777778
1  1.000000  1.000000  1.000000  1.000000
2  0.638889  0.254144  0.598291  0.722222
3  0.166667  0.497238  0.576923  0.833333
>>>
                                    Ln: 35  Col:
```

4.3 Transformation:

What is AI Transformation?:

AI transformation is the next step after digital transformation. After a company adopts digital processes, the next step is to improve the intelligence of those processes. This would increase the level of automation as well as the effectiveness of those processes.

AI transformation touches all aspects of the modern enterprise including both commercial and operational activities. Tech giants are integrating AI into their processes and products. For example, Google is calling itself an "AI-first" organization. Besides tech giants, IDC estimates that at least 90% of new organizations will insert AI technology into their processes and products by 2025.

What are the steps to AI transformation?:

We have listed below a set of the top 6 steps for Fortune 500 firms. Smaller firms could skip having in-house teams and strive for less risky and less investment heavy approaches such as relying on consultants for targeted projects.

1. Outline your company's AI strategy:

An AI strategy should include initiatives which will be uncovered as a result of these exercises:

- Identify your company‟s most valuable unique data sources
- Identify the most important processes which can benefit from automation
- Identify internal resources to drive the AI transformation
- Set ambitious, time-bound business targets

2. Execute pilot projects to gain momentum:

First few projects should create measurable business value while being attainable. This is important for the transformation to gain trust across the organization with achieved projects and it creates momentum that will lead to AI projects with greater success.

3. Build an in-house AI transformation team:

Outsourcing the AI work eases the start of the AI transformation process but building an inhouse AI transformation team can be more advantageous in the long run. If necessary, outsourced partners can help train your staff for upcoming projects.

4. Provide broad AI training:

Organizations should not expect adequate knowledge about AI technologies from their staff. In order to have a successful AI transformation, training each employee in accordance with their role can be beneficial to achieve objectives.

Code :- import pandas as pd # Creating the DataFrame df = pd.DataFrame({"A":[12, 4, 5, None, 1], "B":[7, 2, 54, 3, None], "C":[20, 16, 11, 3, 8], "D":[14, 3, None, 2, 6]}) # Create the index index_ = ['Row_1', 'Row_2', 'Row_3', 'Row_4', 'Row_5'] # Set the index df.index = index_ # Print the DataFrame print(df) # add 10 to each element of the dataframe result = df.transform(func = lambda x : x + 10) # Print the result print(result)

O/P:-

```
          A     B   C     D
Row_1  12.0   7.0  20  14.0
Row_2   4.0   2.0  16   3.0
Row_3   5.0  54.0  11   NaN
Row_4   NaN   3.0   3   2.0
Row_5   1.0   NaN   8   6.0
          A     B   C     D
Row_1  22.0  17.0  30  24.0
Row_2  14.0  12.0  26  13.0
Row_3  15.0  64.0  21   NaN
Row_4   NaN  13.0  13  12.0
Row_5  11.0   NaN  18  16.0
>>>
```

Ln: 79 Col: 4

4.4 PCA (Principal Components Analysis)

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modelling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data. PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the***

power allocation in various communication channels. It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

Variance and Covariance

Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

Dimensionality: It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.

Correlation: It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.

Orthogonal: It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.

Eigenvectors: If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.

Covariance Matrix: A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Code :-

```
# importing required libraries

import numpy as np import

matplotlib.pyplot as plt import

pandas as pd


# importing or loading the dataset dataset

= pd.read_csv("Wine.csv")
```

```
# distributing the dataset into two components X and Y

X = dataset.iloc[:, 0:13].values y = dataset.iloc[:,

13].values


# Splitting the X and Y into the #Training set and

Testing set from sklearn.model_selection import

train_test_split


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


# performing preprocessing part from

sklearn.preprocessing import StandardScaler sc=

StandardScaler()


X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

# Applying PCA function on training

#and testing set of X component from

sklearn.decomposition import PCA


pca = PCA(n_components = 2)

X_train = pca.fit_transform(X_train)

X_test = pca.transform(X_test)
```

```
explained_variance = pca.explained_variance_ratio_

# Fitting Logistic Regression To the training set from
sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train,y_train)

# Predicting the test set result using #predict
function under LogisticRegression y_pred =
classifier.predict(X_test)

# making confusion matrix between
# test set of Y and predicted value. from
sklearn.metrics import confusion_matrix cm
= confusion_matrix(y_test, y_pred)

# Predicting the training set # result through scatter plot from
matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,stop = X_set[:, 0].max() + 1,
step = 0.01),np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
```

```
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75,cmap = ListedColormap(('yellow', 'white','aquamarine')))


plt.xlim(X1.min(),X1.max()) plt.ylim(X2.min(),X2.max())


for i, j in enumerate(np.unique(y_set)):
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green',
'blue'))(i), label = j)


plt.title('Logistic Regression (Training set)')
plt.xlabel('PC1') # for Xlabel plt.ylabel('PC2') # for Ylabel
plt.legend() # to show legend
```

O/P:-

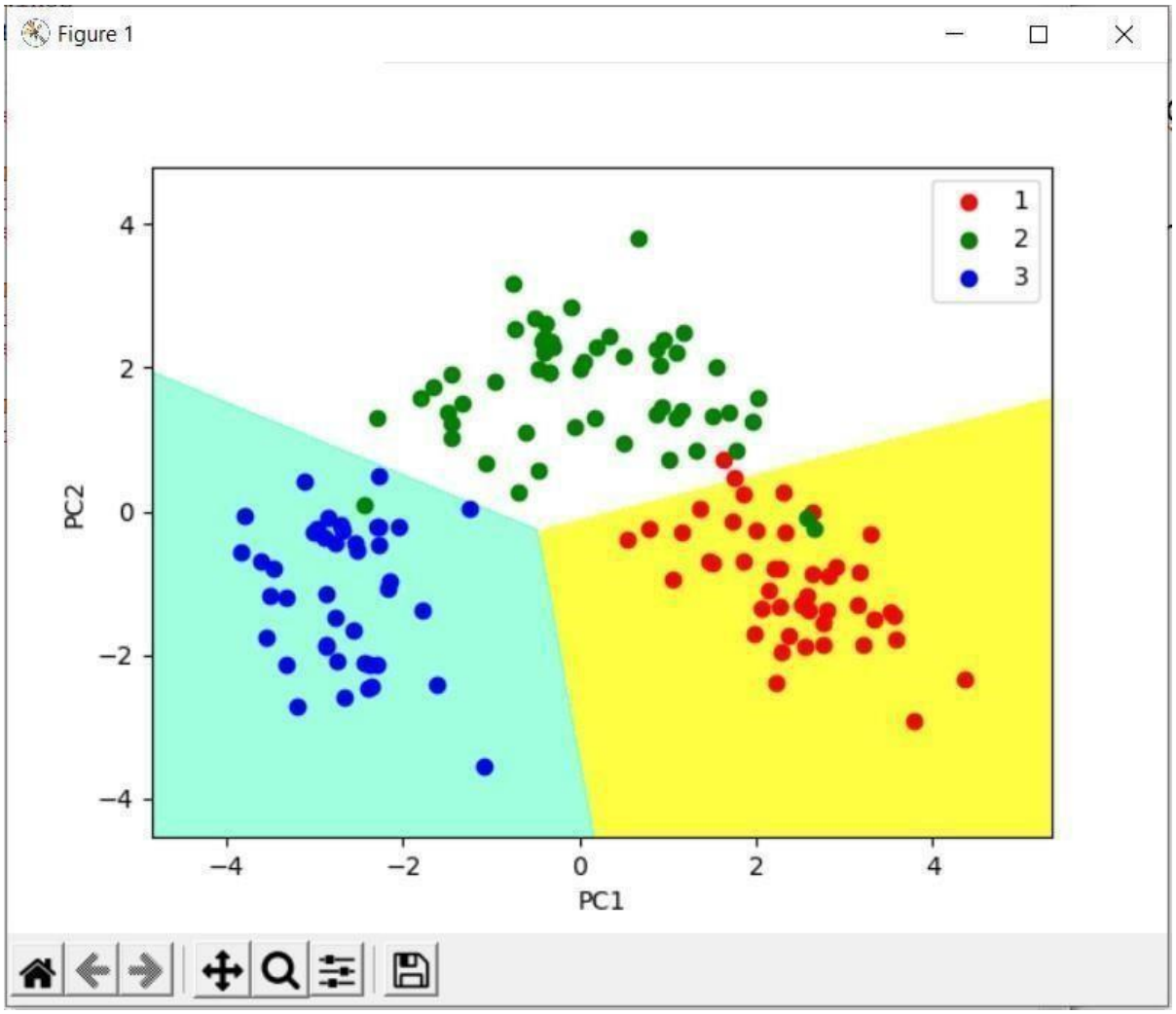


Figure 1
1
2
3
PC2
PC1
4
2
0
−2
−4
−4
−2
0
2
4

Practical No.5

AIM: Implementation of clustering algorithm.

5.1 KMean

A prototypical unsupervised learning algorithm is K-means, which is clustering algorithm. Given X = {x1,...,xm} the goal of K-means is to partition it into k clusters such that each point in a cluster is similar to points from its own cluster than with points from some other cluster. Towards this end, define prototype vectors μ1,...,μk and an indicator vector rij which is 1 if, and only if, xi is assigned to cluster j. To cluster our dataset we will minimize the following distortion measure, which minimizes the distance of each point from the prototype vector:

where r = {}, μ = {μj}, and denotes the usual Euclidean square norm. **K-Means clustering:**

The computation is to be performed in an unsupervised manner. In this section, we describe a solution to this problem that is rooted in clustering, by which we mean the following: Clustering is a form of unsupervised learning whereby a set of observations (i.e., data points) is partitioned into natural groupings or clusters of patterns in such a way that the measure of similarity between any pair of observations assigned to each cluster minimizes a specified cost function.

We have chosen to focus on the so-called *K-means algorithm*, because it is simple to implement, yet effective in performance, two features that have made it highly popular. Let {Xi }Ni=1 denote a set of multidimensional observations that is to be partitioned into a proposed set of K clusters, where K is smaller than the number of observations, N. Let the relationship.

denote a many-to-one mapper, called the encoder, which assigns the ith observation xi to the jth cluster according to a rule yet to be defined. To do this encoding, we need a measure of similarity between every pair of vectors xi and xi' which is denoted by d(xi, xi').When the measure d(xi, xi') is small enough, both xi and xi' are assigned to the same cluster; otherwise, they are assigned to different clusters.

To optimize the clustering process, we introduce the following cost function (Hastie et al.,2001):

$$J(C) = \frac{1}{2}\sum_{j=1}^{K}\sum_{C(i)=j}\sum_{C(i')=j} d(\mathbf{x}_i, \mathbf{x}_{i'})$$

Code :- import numpy as np import
pandas as pd import
matplotlib.pyplot as plt from
sklearn import datasets import
seaborn as sns from sklearn.cluster
import KMeans

iris = datasets.load_iris()

#Creating data frame
Data = pd.DataFrame(iris.data, columns = iris.feature_names)

#Top values of Dataset print(Data.head())

#Bottom Values of Dataset print(Data.tail())

Settin the data x=Data.iloc[:,0:3].values

css=[]

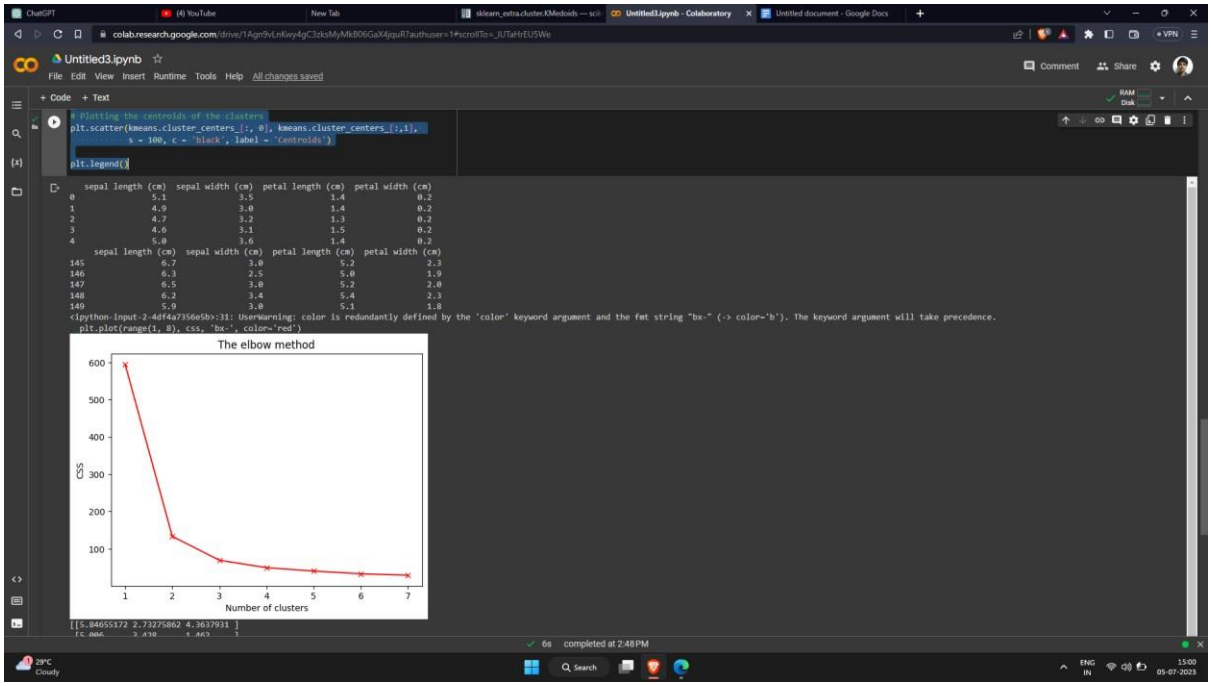
Finding inertia on various k values for
i in range(1,8):
    kmeans=KMeans(n_clusters = i, init = 'k-means++',
max_iter = 100, n_init = 10, random_state = 0).fit(x)
css.append(kmeans.inertia_)

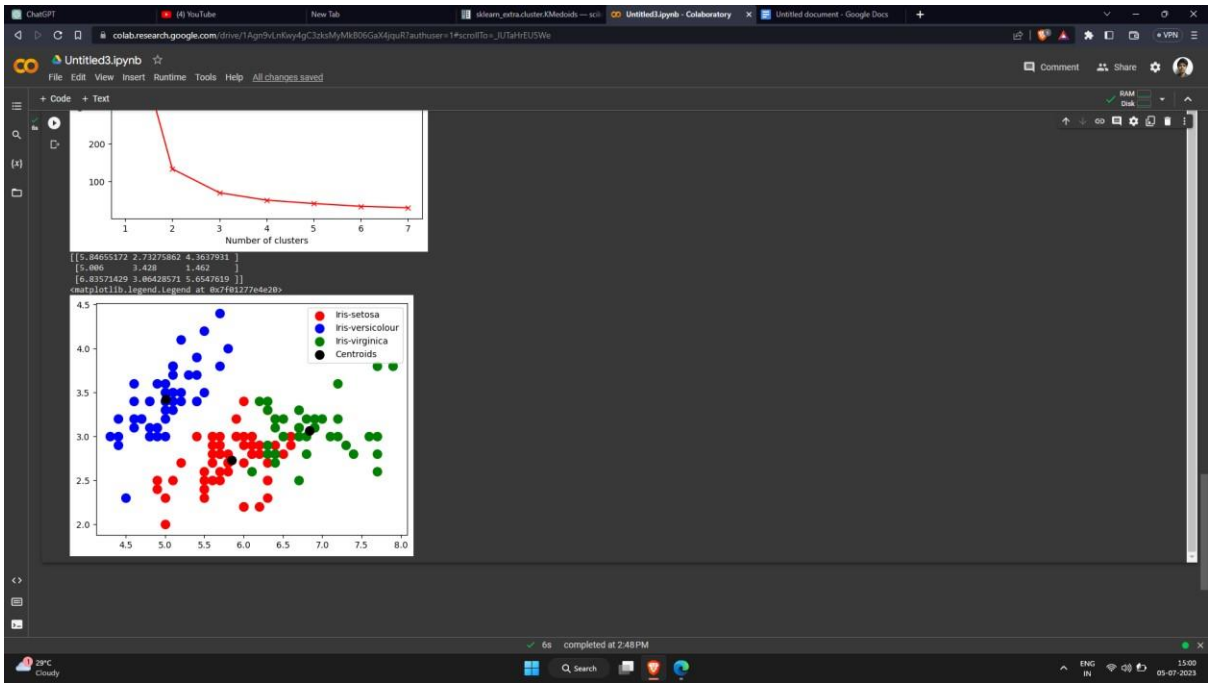
plt.plot(range(1, 8), css, 'bx-', color='red')
plt.title('The elbow method') plt.xlabel('Number
of clusters') plt.ylabel('CSS') plt.show()

#Applying Kmeans classifier kmeans = KMeans(n_clusters=3,init = 'k-means++', max_iter =
100, n_init = 10, random_state = 0)

```
y_kmeans = kmeans.fit_predict(x)

print(kmeans.cluster_centers_)

plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1],
s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1],
s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],
s = 100, c = 'green', label = 'Iris-virginica')

# Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],
s = 100, c = 'black', label = 'Centroids')

plt.legend()
```

O/P :-


Untitled3.ipynb
Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],
s = 100, c = 'black', label = 'Centroids')
plt.legend()
sepal length (cm) sepal width (cm) petal length (cm) petal width (cm)
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
sepal length (cm) sepal width (cm) petal length (cm) petal width (cm)
145 6.7 3.0 5.2 2.3
146 6.3 2.5 5.0 1.9
147 6.5 3.0 5.2 2.0
148 6.2 3.4 5.4 2.3
149 5.9 3.0 5.1 1.8
<ipython-input-2-4df4a7356e5b>:31: UserWarning: color is redundantly defined by the 'color' keyword argument and the fmt string "bx-" (-> color='b'). The keyword argument will take precedence.
plt.plot(range(1, 8), css, 'bx-', color='red')
The elbow method
CSS
Number of clusters
[[5.84655172 2.73275862 4.3637931]
6s completed at 2:48 PM

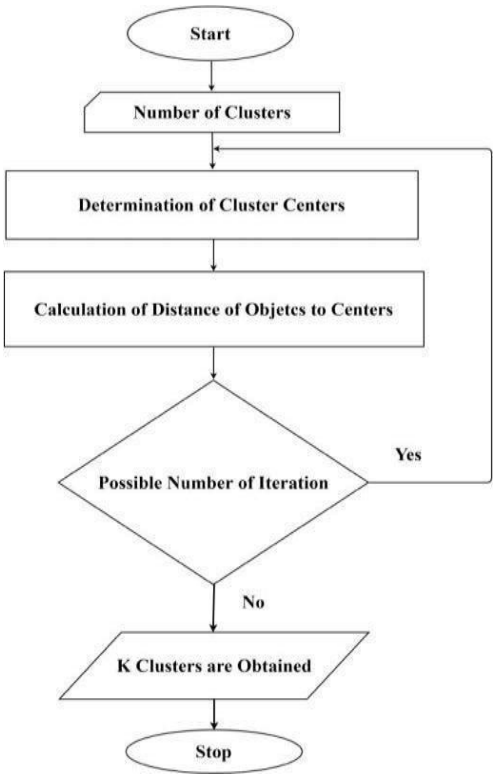

Untitled3.ipynb
Number of clusters
[[5.84655172 2.73275862 4.3637931]
[5.006 3.428 1.462]
[6.83571429 3.06428571 5.6547619]]
<matplotlib.legend.Legend at 0x7f01277e4e20>
iris-setosa
iris-versicolour
iris-virginica
Centroids
6s completed at 2:48 PM

5.2 KMedoid

K-Medoids is a clustering algorithm resembling the K-Means clustering technique. It falls under the category of unsupervised machine learning.

It majorly differs from the K-Means algorithm in terms of the way it selects the clusters‟ centres. The former selects the average of a cluster‟s points as its centre (which may or may not be one of the data points) while the latter always picks the actual data points from the clusters as their centres (also known as „**exemplars**‟ or „**medoids**‟). K-Medoids also differs in this respect from the K-Medians algorithm which is the same as K-means, except that it chooses the medians (instead of means) of the clusters as centres.

The mean in k-means clustering is sensitive to outliers. Since an object with an extremely high value may substantially distort the distribution of data. Hence we move to k-medoids. Instead of taking mean of cluster we take the most centrally located point in cluster as it‟s center. These are called medoids.



Advantages:

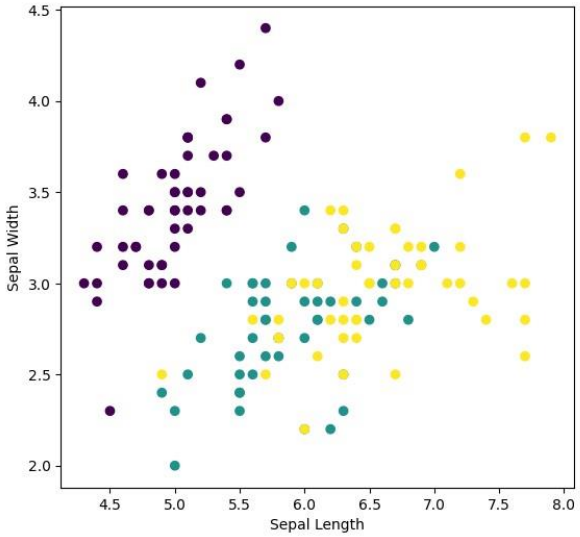
- PAM is more flexible as it can use any similarity measure.
- PAM is more robust than k-means as it handles noise better.

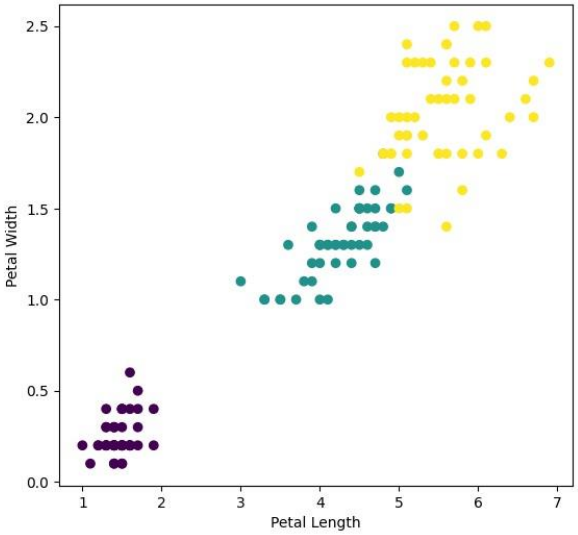
Disadvantages:

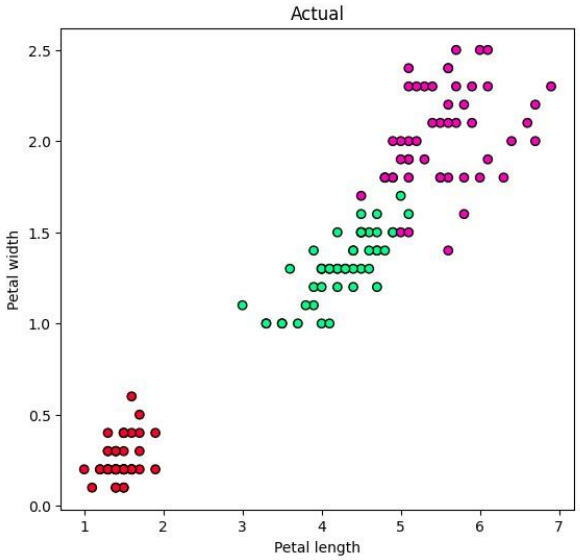
PAM algorithm for K-medoid clustering works well for dataset but cannot scale well for large data set due to high computational overhead.

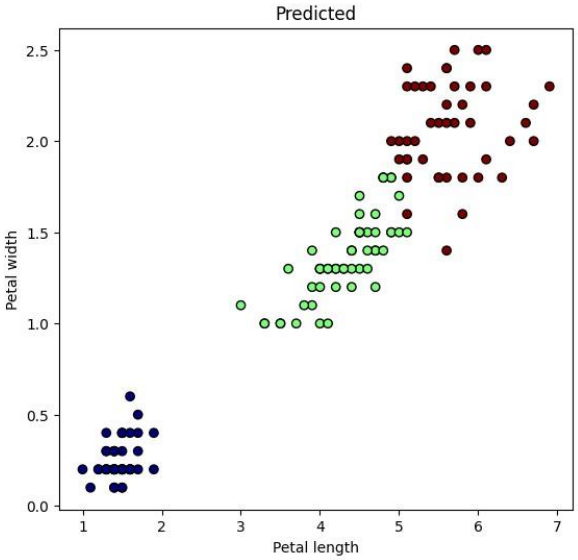
Code :- import numpy as np import
matplotlib.pyplot as plt from
sklearn_extra.cluster import KMedoids from
sklearn import datasets iris =
datasets.load_iris()
X = [] X = iris.data y = iris.target fig, axes =
plt.subplots(1, 2, figsize=(14,6),dpi=100)
axes[0].scatter(X[:,0], X[:,1], c=y)
axes[0].set_xlabel('Sepal Length')
axes[0].set_ylabel('Sepal Width')
axes[1].scatter(X[:,2], X[:,3], c=y)
axes[1].set_xlabel('Petal Length')
axes[1].set_ylabel('Petal Width') D=X[:,2:]
cluster_num=3 model = KMedoids(cluster_num)
centers = model.fit(D) print(centers) new_labels =
model.predict(D)
#Plot the identified clusters and compare with our result
fig, axes = plt.subplots(1, 2, figsize=(14,6),dpi=100)
axes[0].scatter(D[:, 0], D[:, 1], c=y, cmap='gist_rainbow',
edgecolor='k') axes[1].scatter(D[:, 0], D[:, 1],
c=new_labels, cmap='jet', edgecolor='k')
axes[0].set_xlabel('Petal length')
axes[0].set_ylabel('Petal width') axes[1].set_xlabel('Petal
length') axes[1].set_ylabel('Petal width')
axes[0].set_title('Actual') axes[1].set_title('Predicted')

O/P :-


Sepal Width
Sepal Length


Petal Width
Petal Length


Actual
Petal width
Petal length


Predicted
Petal width
Petal length

Practical No.6

AIM: Implementation of Classifying data using Support Vector Machines (SVMs).

An SVM model is a representation of the examples as points in space, mapped so that the examples of the different categories are separated by as wide a gap as possible. SVMs may do non-linear classification, implicitly translating their inputs into high-dimensional feature spaces, in addition to linear classification.

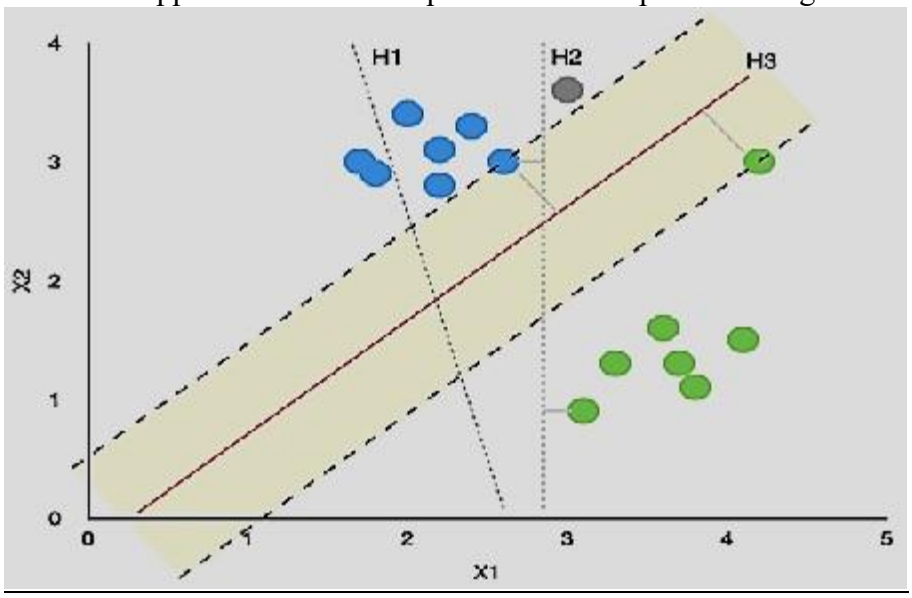
An SVM training algorithm creates a model that assigns new examples to one of two categories, making it a non-probabilistic binary linear classifier, given a series of training examples that are individually designated as belonging to one of two categories.

Before you go any further, make sure you have a basic knowledge of this topic. In this article, I'll show you how to use machine learning techniques like scikit-learn to classify cancer UCI datasets using SVM.

Assume we have a set of points that are divided into two classes. We want to split those two classes so that we can accurately assign any new points to one or the other in the future.

The SVM algorithm seeks out a hyperplane that separates these two classes by the greatest margin possible. A hard margin can be utilized if classes are entirely linearly separable. Otherwise, a soft margin is required.

Note that support vectors are the points that end up on the margins.



Code :-

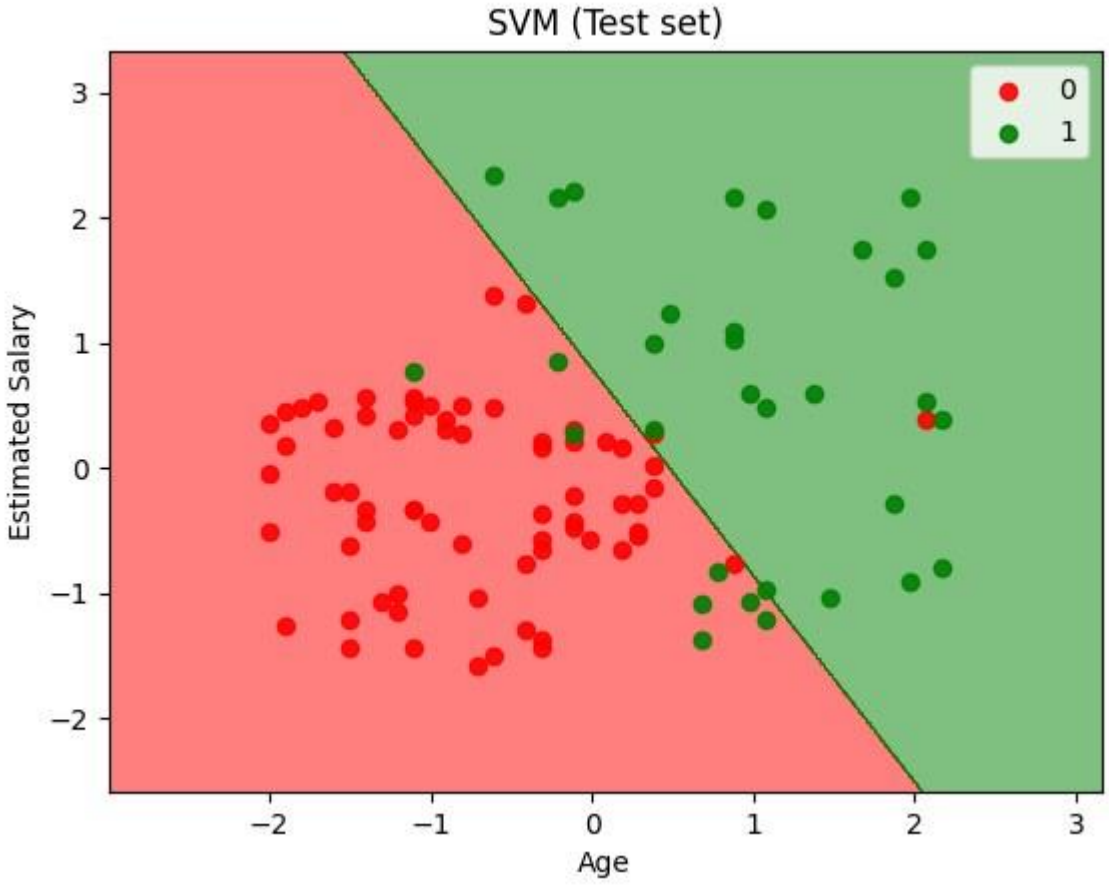
```
# Support Vector Machine (SVM) with 'linear' kernel


# Importing the libraries

import numpy as np import

matplotlib.pyplot as plt import

pandas as pd


# Importing the dataset dataset =

pd.read_csv('Social_Network_Ads.csv') X =

dataset.iloc[:, [2, 3]].values y = dataset.iloc[:,

4].values


# Splitting the dataset into the Training set and Test set from

sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)


# Feature Scaling from sklearn.preprocessing

import StandardScaler sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)
```

```
# Fitting the classifier classifier to the Training set from
sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state= 0) classifier.fit(X_train,
y_train)

# Predicting the Test set results y_pred
= classifier.predict(X_test)

# Making the Confusion Matrix from
sklearn.metrics import confusion_matrix cm
= confusion_matrix(y_test, y_pred)

# Visualising the Training set results from
matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1,
step = 0.01),                np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max()
+ 1, step =
0.01)) plt.contourf(X1, X2,
classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),          alpha = 0.5,
cmap = ListedColormap(('red', 'green'))) plt.xlim(X1.min(),
X1.max()) plt.ylim(X2.min(), X2.max()) for i, j in
enumerate(np.unique(y_set)):
```

```
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], alpha=0.5,
c = ListedColormap(('red', 'green'))(i), label = j) plt.title('SVM
(Training set)') plt.xlabel('Age') plt.ylabel('Estimated Salary')
plt.legend() plt.show()
# Visualising the Test set results from
matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1,
step = 0.01),                np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max()
+ 1, step =
0.01)) plt.contourf(X1, X2,
classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),          alpha = 0.5,
cmap = ListedColormap(('red', 'green'))) plt.xlim(X1.min(),
X1.max()) plt.ylim(X2.min(), X2.max()) for i, j in
enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], alpha=0.9,
c = ListedColormap(('red', 'green'))(i), label = j) plt.title('SVM
(Test set)') plt.xlabel('Age') plt.ylabel('Estimated Salary')
plt.legend() plt.show()
```

O/P :-


SVM (Training set)
Estimated Salary
Age
0
1
3
2
1
0
−1
−2
−2
−1
0
1
2
3


SVM (Test set)
Estimated Salary
Age
0
1
3
2
1
0
−1
−2
−2
−1
0
1
2
3

Practical No.7 AIM:

Implementation of Bagging Algorithm.

7.1 Decision Tree

Objectives: This chapter will enable students to:

- Make use of Data sets in implementing the machine learning algorithms
- Implement the machine learning concepts and algorithms in any suitable language of choice. Data sets can be taken from standard repositories or constructed by the students.

Introduction:

Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. Makes use of the Tree representation. Can be used for classification. Given a decision tree, how do we predict an outcome for a class label? We start from the root of the tree.

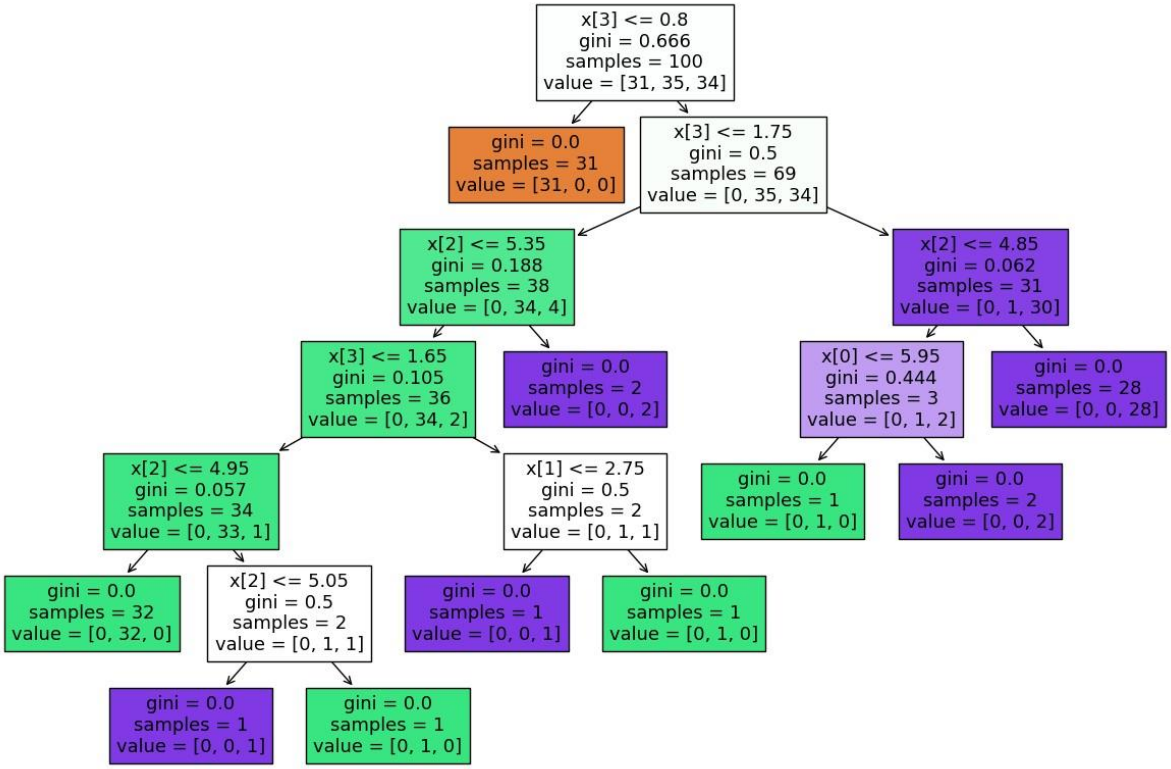
CART stands for Classification and Regression Trees.

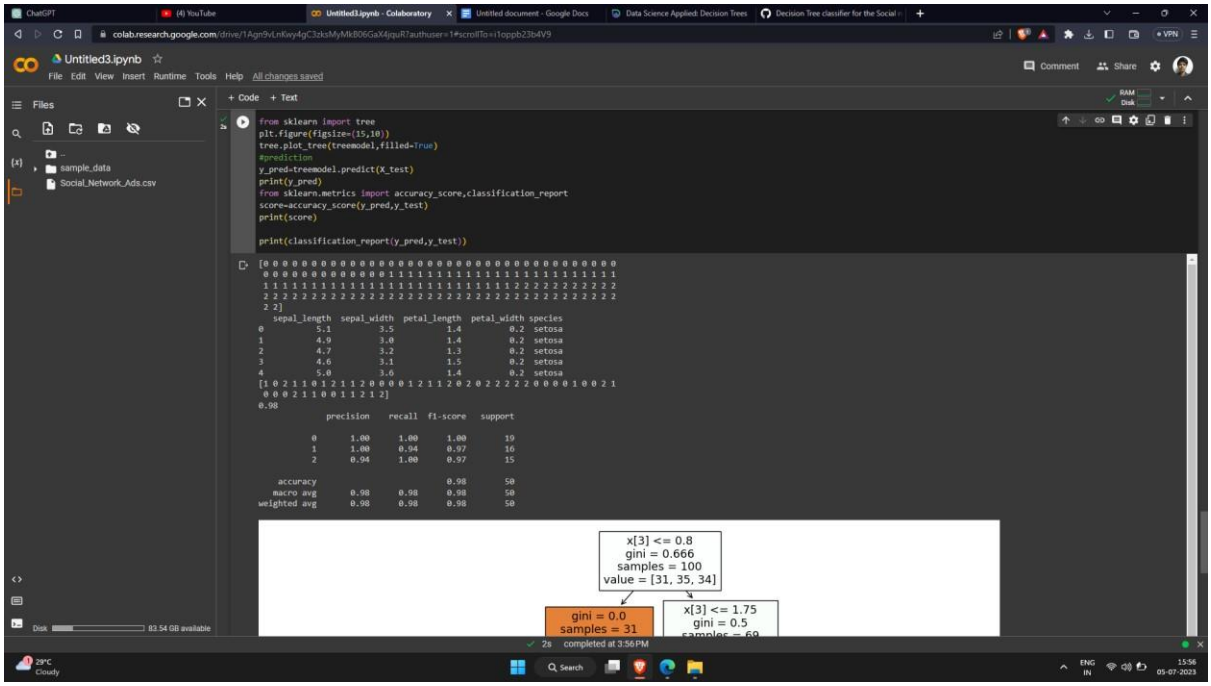
For example, consider a dataset of cats and dogs, with their features. The label here is accordingly "cat", or "dog", and the goal is to identify the animal based on its features, using a decision tree. Say, if at a particular node in the tree, the input to a node contains only a single type of label, say cats, we can infer that it is perfectly grouped, or "unmixed". On the other hand, if the input contains a mix of cats and dogs, we would have to ask another question about the features in the dataset that can help us narrow down, and divide the mix further to try and "unmix" them completely.

Code :- import pandas as pd import
matplotlib.pyplot as plt %matplotlib
inline from sklearn.datasets import
load_iris iris=load_iris()
print(iris.target) import seaborn as
sns df=sns.load_dataset('iris')
print(df.head()) X=df.iloc[:,:-1]
y=iris.target ### train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.33, random_state=42) from
sklearn.tree import DecisionTreeClassifier
Postpruning treemodel=DecisionTreeClassifier()

treemodel.fit(X_train,y_train)
DecisionTreeClassifier() from
sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(treemodel,filled=True)
#prediction y_pred=treemodel.predict(X_test)
print(y_pred)
from sklearn.metrics import accuracy_score,classification_report
score=accuracy_score(y_pred,y_test) print(score)
print(classification_report(y_pred,y_test))

O/P :-


x[3] <= 0.8
gini = 0.666
samples = 100
value = [31, 35, 34]
gini = 0.0
samples = 31
value = [31, 0, 0]
x[3] <= 1.75
gini = 0.5
samples = 69
value = [0, 35, 34]
x[2] <= 5.35
gini = 0.188
samples = 38
value = [0, 34, 4]
x[2] <= 4.85
gini = 0.062
samples = 31
value = [0, 1, 30]
x[3] <= 1.65
gini = 0.105
samples = 36
value = [0, 34, 2]
gini = 0.0
samples = 2
value = [0, 0, 2]
x[0] <= 5.95
gini = 0.444
samples = 3
value = [0, 1, 2]
gini = 0.0
samples = 28
value = [0, 0, 28]
x[2] <= 4.95
gini = 0.057
samples = 34
value = [0, 33, 1]
x[1] <= 2.75
gini = 0.5
samples = 2
value = [0, 1, 1]
gini = 0.0
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 2
value = [0, 0, 2]
gini = 0.0
samples = 32
value = [0, 32, 0]
x[2] <= 5.05
gini = 0.5
samples = 2
value = [0, 1, 1]
gini = 0.0
samples = 1
value = [0, 0, 1]
gini = 0.0
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 1
value = [0, 0, 1]
gini = 0.0
samples = 1
value = [0, 1, 0]


Untitled3.ipynb
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(treemodel,filled=True)
#prediction
y_pred=treemodel.predict(X_test)
print(y_pred)
from sklearn.metrics import accuracy_score,classification_report
score=accuracy_score(y_pred,y_test)
print(score)
print(classification_report(y_pred,y_test))
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
0.98
precision recall f1-score support
0 1.00 1.00 1.00 19
1 1.00 0.94 0.97 16
2 0.94 1.00 0.97 15
accuracy 0.98 50
macro avg 0.98 0.98 0.98 50
weighted avg 0.98 0.98 0.98 50
x[3] <= 0.8
gini = 0.666
samples = 100
value = [31, 35, 34]
gini = 0.0
samples = 31
x[3] <= 1.75
gini = 0.5

7.2 Random Forest

Random forest is like bootstrapping algorithm with Decision tree (CART) model. Suppose we have 1000 observations in the complete population with 10 variables. Random forest will try to build multiple CART along with different samples and different initial variables. It will take a random sample of 100 observations and then chose 5 initial variables randomly to build a CART model. It will go on repeating the process say about 10 times and then make a final prediction on each of the observations. Final prediction is a function of each prediction. This final prediction can simply be the mean of each prediction.

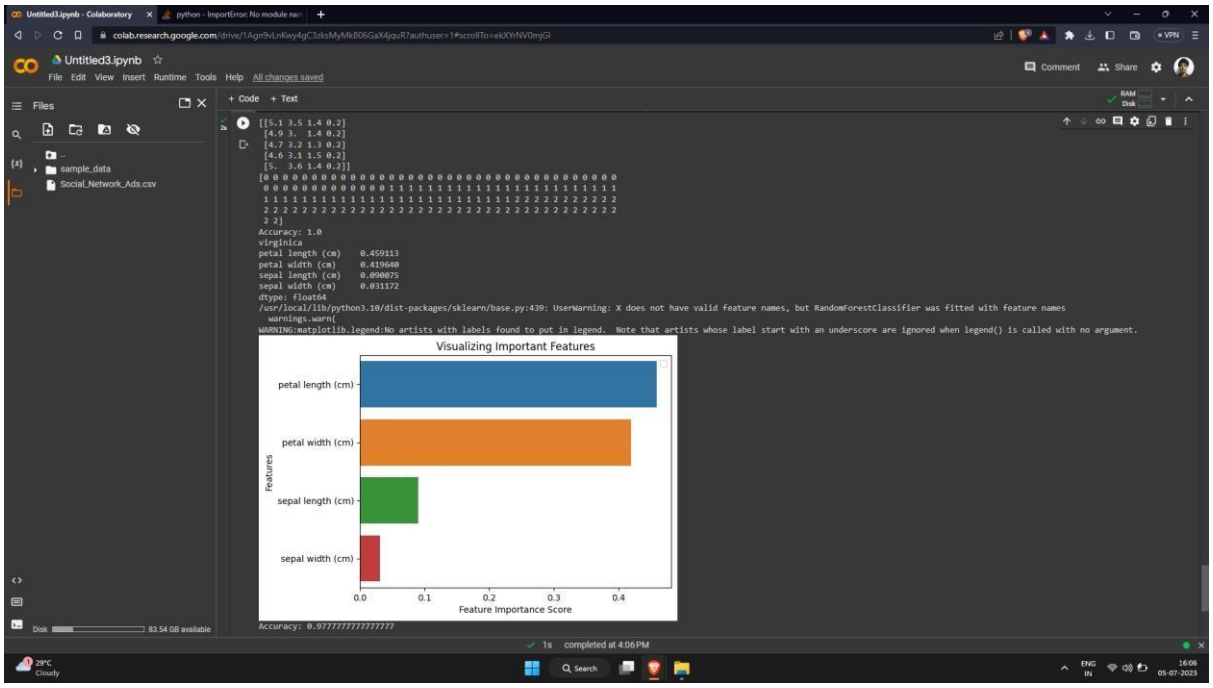
Code :-

```
#Import scikit-learn dataset library from
sklearn import datasets
#Load dataset iris =
datasets.load_iris()
# print the iris data (top 5 records) print(iris.data[0:5])
# print the iris labels (0:setosa, 1:versicolor, 2:virginica)
print(iris.target) import pandas as pd
data=pd.DataFrame({
   'sepal length':iris.data[:,0],
   'sepal width':iris.data[:,1],
   'petal length':iris.data[:,2],
   'petal width':iris.data[:,3],
   'species':iris.target
}) data.head()
# Import train_test_split function from
sklearn.model_selection import train_test_split

X=data[['sepal length', 'sepal width', 'petal length', 'petal width']]  # Features y=data['species']
# Labels

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30%
test
#Import Random Forest Model from sklearn.ensemble
import RandomForestClassifier

#Create a Gaussian Classifier clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
```

```
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
#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))
species_idx = clf.predict([[3, 5, 4, 2]])[0]
print(iris.target_names[species_idx]) import pandas as pd
feature_imp =
pd.Series(clf.feature_importances_,index=iris.feature_names).sort_values(ascending=False)
print(feature_imp) import matplotlib.pyplot as plt import seaborn as sns
%matplotlib inline

# Creating a bar plot sns.barplot(x=feature_imp,
y=feature_imp.index)

# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features') plt.title("Visualizing
Important Features") plt.legend()
plt.show()

# Import train_test_split function from
sklearn.model_selection import train_test_split

# Split dataset into features and labels
X=data[['petal length', 'petal width','sepal length']]  # Removed feature "sepal length"
y=data['species']                                 # Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30%
test
from sklearn.ensemble import RandomForestClassifier
```

```
#Create a Gaussian Classifier clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test) clf.fit(X_train,y_train)

# prediction on test set y_pred=clf.predict(X_test)

#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))
```

O/P :-


Untitled3.ipynb
Visualizing Important Features
petal length (cm)
petal width (cm)
sepal length (cm)
sepal width (cm)
Features
Feature Importance Score
0.0
0.1
0.2
0.3
0.4

Practical No.8 AIM:

Implementation of Boosting Algorithm.

Boosting algorithms are the exceptional algorithms that are utilized to enhance the existing result of the data model and assist to fix the errors. [1,4,7] They utilize the concept of the weak learner and strong learner discussion through the weighted average values and higher votes values for prediction. They use decision stamp, margin maximizing classification for processing purpose. Machine learning algorithms like AdaBoost or Adaptive boosting Algorithm, Gradient, XG Boosting algorithm and Voting Ensemble are used to follow the process of training for predicting and fine-tuning of the result. [1,4,7] To choose the right distributions follow the steps as specified:

Step 1: The base Learning algorithm combines each distribution and applies equal weight to each distribution.

Step 2: If any prediction occurs during the first base learning algorithm, then we pay high attention to that prediction error.

Step 3: Repeat step 2 until the limit of the Base Learning algorithm has been reached or high accuracy.

Step 4: Combines the entire weak learner to create one strong prediction rule.

An adaBoost calculation can be utilized to boost the execution of any machine learning calculation. Machine Learning has gotten to be a capable tool which can make predictions based on a huge sum of data. It has ended up so well known in later times that the application of machine learning can be found in our day-to-day exercises [1,4,7]. A common illustration of it is getting proposals for items whereas shopping online based on the past things bought by the client. Machine Learning, frequently alluded to as predictive analysis, can be characterized as the capability of computers to memorize without being programmed unequivocally. As a substitute, it utilizes the algorithms to

Boosting originated from the question of whether a set of weak classifiers could be converted to a strong classifier or not? A weak learner is a learner who is better than random guessing. AdaBoost transforms weak learners or predictors to strong predictors in order to solve problems of classification [1,4,7].

For classification, the final equation can be put as below:

$$F(x) = sign(\sum_{m=1}^{M} \theta_m f_m(x)),$$

Code :-

```
# Boosting Algorithm # 1- AdaBoost import pandas
as pd import numpy as np from
sklearn.model_selection import train_test_split from
sklearn.ensemble import AdaBoostClassifier


df=pd.read_csv("diabetes.csv")
#print(df)
"""
Separate the independet and dependent variables saving the features in x
and the target variable in y and later we need to divide data into train and
test set using train_test_split
""" x=df.drop(['Outcome'],axis=1) y=df['Outcome']
train_x,test_x,train_y,test_y=train_test_split(x,y,random_state=101,stratify=y)
# Creating object of AdaBoostClassifier
obj1=AdaBoostClassifier(random_state=96)
obj1.fit(train_x,train_y) # Printing accuracy
print(obj1.score(train_x,train_y))
print(obj1.score(test_x,test_y))
```

O/P:-

```
0.8263888888888888
0.765625
```

Practical No.9 AIM:

Deployment of Machine Learning Models.

Index.html

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'> <link
href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet'
type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
 <div class="login">
        <h1>Predict Salary Analysis</h1>

    <!-- Main Input For Receiving Query to our ML -->
   <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="experience" placeholder="Experience"
required="required" />
      <input type="text" name="test_score" placeholder="Test Score" required="required" />
<input type="text" name="interview_score" placeholder="Interview Score"
required="required" />

      <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
</form>

   <br>
   <br>
   {{ prediction_text }}

 </div>


</body>
</html>
```

Style.css

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
.btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px 4px; margin-bottom: 0; font-size: 13px; line-height: 18px; color: #333333; text-align: center;text-shadow: 0 1px 1px rgba(255, 255, 255, 0.75); vertical-align: middle; background-color: #f5f5f5; backgroundimage: -moz-linear-gradient(top, #ffffff, #e6e6e6); background-image: -ms-lineargradient(top, #ffffff, #e6e6e6); background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#ffffff), to(#e6e6e6)); background-image: -webkit-linear-gradient(top, #ffffff, #e6e6e6); background-image: -o-linear-gradient(top, #ffffff, #e6e6e6); background-image: lineargradient(top, #ffffff, #e6e6e6); background-repeat: repeat-x; filter: progid:dximagetransform.microsoft.gradient(startColorstr=#ffffff, endColorstr=#e6e6e6, GradientType=0); border-color: #e6e6e6 #e6e6e6 #e6e6e6; border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25); border: 1px solid #e6e6e6; -webkit-border-radius: 4px; moz-border-radius: 4px; border-radius: 4px; -webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05); -moz-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05); box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05); cursor: pointer; *margin-left: .3em; }
.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] { background-color: #e6e6e6; }
.btn-large { padding: 9px 14px; font-size: 15px; line-height: normal; -webkit-border-radius: 5px; -moz-border-radius: 5px; border-radius: 5px; }
.btn:hover { color: #333333; text-decoration: none; background-color: #e6e6e6; backgroundposition: 0 -15px; -webkit-transition: background-position 0.1s linear; -moz-transition: background-position 0.1s linear; -ms-transition: background-position 0.1s linear; -o-transition: background-position 0.1s linear; transition: background-position 0.1s linear; }
.btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #ffffff; }
.btn-primary.active { color: rgba(255, 255, 255, 0.75); }
.btn-primary { background-color: #4a77d4; background-image: -moz-linear-gradient(top, #6eb6de, #4a77d4); background-image: -ms-linear-gradient(top, #6eb6de, #4a77d4); background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#6eb6de), to(#4a77d4)); background-image: -webkit-linear-gradient(top, #6eb6de, #4a77d4); background-image: -olinear-gradient(top, #6eb6de, #4a77d4); background-image: linear-gradient(top, #6eb6de, #4a77d4); background-repeat: repeat-x; filter: progid:dximagetransform.microsoft.gradient(startColorstr=#6eb6de, endColorstr=#4a77d4, GradientType=0); border: 1px solid #3762bc; text-shadow: 1px 1px 1px rgba(0,0,0,0.4); boxshadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.5); }
.btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-primary.disabled, .btnprimary[disabled] { filter: none; background-color: #4a77d4; }
.btn-block { width: 100%; display:block; }
```

```
* { -webkit-box-sizing:border-box; -moz-box-sizing:border-box; -ms-box-sizing:border-box;
o-box-sizing:border-box; box-sizing:border-box; }

html { width: 100%; height:100%; overflow:hidden; }

body {
        width: 100%;
height:100%;
 font-family: 'Open Sans', sans-serif; background:
        #092756;
 color: #fff; font-size: 18px; text-align:center; letter-spacing:1.2px; background: -moz-radial-
gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%),-moz-
linear-gradient(top, rgba(57,173,219,.25) 0%, rgba(42,60,87,.4) 100%), -moz-linear-gradient(-
45deg, #670d10 0%, #092756 100%); background: -webkit-radial-gradient(0% 100%, ellipse
cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -webkit-linear-gradient(top,
rgba(57,173,219,.25) 0%,rgba(42,60,87,.4) 100%), -webkit-linear-gradient(-45deg, #670d10
0%,#092756 100%); background: -o-radial-gradient(0% 100%, ellipse cover,
rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -o-linear-gradient(top,
rgba(57,173,219,.25) 0%,rgba(42,60,87,.4) 100%), -o-linear-gradient(-45deg, #670d10
0%,#092756 100%); background: -ms-radial-gradient(0% 100%, ellipse cover,
rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -ms-linear-gradient(top,
rgba(57,173,219,.25) 0%,rgba(42,60,87,.4) 100%), -ms-linear-gradient(-45deg, #670d10
0%,#092756 100%); background: -webkit-radial-gradient(0% 100%, ellipse cover,
rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), linear-gradient(to bottom,
rgba(57,173,219,.25) 0%,rgba(42,60,87,.4) 100%), linear-gradient(135deg, #670d10
0%,#092756 100%); filter: progid:DXImageTransform.Microsoft.gradient(
startColorstr='#3E1D6D', endColorstr='#092756',GradientType=1 );

}
.login {
        position: absolute;     top:
40%;   left: 50%;       margin: -
150px 0 0 -150px;
width:400px;
        height:400px;
}

.login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing:1px;
textalign:center; }

input {
        width: 100%;            margin-
bottom: 10px;
        background: rgba(0,0,0,0.3);
        border: none;
outline: none;
```

```
padding: 10px;           font-
size: 13px;      color: #fff;
 text-shadow: 1px 1px 1px rgba(0,0,0,0.3);  border: 1px
solid rgba(0,0,0,0.3);
        border-radius: 4px;
 box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);
        -webkit-transition: box-shadow .5s ease;
        -moz-transition: box-shadow .5s ease;
        -o-transition: box-shadow .5s ease;    -ms-
transition: box-shadow .5s ease;
        transition: box-shadow .5s ease;
}
input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px
rgba(255,255,255,0.2); }
```

app.py import
numpy as np
from flask import Flask, request, jsonify, render_template import
pickle

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

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

@app.route('/predict',methods=['POST']) def
predict():
   '''
   For rendering results on HTML GUI
   '''
   int_features = [int(x) for x in request.form.values()]
final_features = [np.array(int_features)]
   prediction = model.predict(final_features)

   output = round(prediction[0], 2)

   return render_template('index.html', prediction_text='Employee Salary should be $
{}'.format(output))

@app.route('/predict_api',methods=['POST']) def
predict_api():
   '''
   For direct API calls trought request
```

```
    '''
    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

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

model.py

```
# Importing the libraries
import numpy as np import
matplotlib.pyplot as plt import
pandas as pd
import pickle

dataset = pd.read_csv('hiring.csv')

dataset['experience'].fillna(0, inplace=True)

dataset['test_score'].fillna(dataset['test_score'].mean(), inplace=True)

X = dataset.iloc[:, :3]

#Converting words to integer values def
convert_to_int(word):
    word_dict = {'one':1, 'two':2, 'three':3, 'four':4, 'five':5, 'six':6, 'seven':7, 'eight':8,
            'nine':9, 'ten':10, 'eleven':11, 'twelve':12, 'zero':0, 0: 0}
return word_dict[word]

X['experience'] = X['experience'].apply(lambda x : convert_to_int(x))

y = dataset.iloc[:, -1]

#Splitting Training and Test Set
#Since we have a very small dataset, we will train our model with all availabe data.

from sklearn.linear_model import LinearRegression regressor
= LinearRegression()

#Fitting model with trainig data regressor.fit(X,
y)

# Saving model to disk
```

```
pickle.dump(regressor, open('model.pkl','wb'))

# Loading model to compare the results model
= pickle.load(open('model.pkl','rb'))
print(model.predict([[2, 9, 6]]))
```

request.py import
requests

```
url = 'http://localhost:5000/predict_api'
r = requests.post(url,json={'experience':2, 'test_score':9, 'interview_score':6})

print(r.json())
```

O/P :-