Jahangirnagar University

ICT 4102– Artificial Intelligence Lab

LAB REPORT

SUBMITTED BY
Name: Sanjida Akter
Roll No: 1982
Exam Roll: 192299
Submission Date: 05-09-2023

SUBMITTED TO
Md. Mahmudur Rahman
Lecturer, IIT

Index

<table>
<tr><td></td><td>
<table>
<tr><td>xii)</td><td>Exercise 12</td><td>40</td></tr>
<tr><td>xiii)</td><td>Exercise 13</td><td>40</td></tr>
<tr><td>xiv)</td><td>Exercise 14</td><td>41</td></tr>
<tr><td>xv)</td><td>Exercise 15</td><td>41</td></tr>
</table>
</td></tr>
<tr><td>04</td><td>Lab Report: 04
<table>
<tr><td>i)</td><td>Apply linear regression techniques to predict housing prices based on the provided features in the attached Dataset.</td><td>44</td></tr>
</table>
</td></tr>
<tr><td>05</td><td>Lab Report: 05
<table>
<tr><td>i)</td><td>Analysis using Logistic Regression and predict the 10-year risk of coronary heart disease.</td><td>54</td></tr>
</table>
</td></tr>
<tr><td>06</td><td>Lab Report: 06
<table>
<tr><td>i)</td><td>Analysis using K-NN using any dataset.</td><td>66</td></tr>
</table>
</td></tr>
<tr><td>07</td><td>Lab Report: 07
<table>
<tr><td>i)</td><td>Use any dataset to classify using SVM.</td><td>76</td></tr>
</table>
</td></tr>
<tr><td>08</td><td>Lab Report: 08
<table>
<tr><td>i)</td><td>Use any dataset to use unsupervised algorithm: K means clustering</td><td>86</td></tr>
</table>
</td></tr>
</table>

Institute of Information Technology (IIT)

Jahangirnagar University

Lab Report: 01

Submitted by:

Name: Sanjida Akter

Roll No: 1982

Lab Date: 23-05-2023

Submission Date: 29-05-2023

Lab Report # Day 01

Example 1:

Problem Name: Knowledge base 1.

Clause:

```
woman(rupa).
woman(joba).
woman(jui).
playsAirGuitar(joba).
party.
```

Queries:

1. woman(rupa).
2. playsAirGuitar(joba).
3. playsAirGuitar(rupa).
4. tattoed(joba).
5. party.
6. rockConcert.

Result:

```
% f:/4th Year, 1st Semester/AI LAB/knowledge base 1.pl compiled 0.00 se
c, 5 clauses
?- woman(rupa).
true.

?- playsAirGuitar(joba).
true.

?- playsAirGuitar(rupa).
false.

?- tattoed(joba).
ERROR: Unknown procedure: tattoed/1 (DWIM could not correct goal)
?- party.
true.

?- rockConcert.
ERROR: Unknown procedure: rockConcert/0 (DWIM could not correct goal)
?-
```

Example 2:

Problem Name: Knowledge base 2.

Clause:

```
happy(jui).
```

```
listens2music(rupa).
listens2music(jui):- happy(jui).
playsAirGuitar(rupa):- listens2music(rupa).
playsAirGuitar(jui):- listens2music(jui).
```

Queries:

1. playsAirGuitar(rupa).
2. playsAirGuitar(jui).

Result:

```
?- playsAirGuitar(rupa).
true.

?- playsAirGuitar(jui).
true.
```

Example 3:

Problem Name: Knowledge base 3. (expressing conjunction)

Clause:

```
happy(viki).
listens2music(beni).
playsAirGuitar(viki):- listens2music(viki), happy(viki).
playsAirGuitar(beni):- happy(beni).
playsAirGuitar(beni):- listens2music(beni).
```

Queries:

1. playsAirGuitar(viki).
2. playsAirGuitar(beni).

Result:

```
?- playsAirGuitar(viki).
false.

?- playsAirGuitar(beni).
true.
```

Example 4:

Problem Name: Knowledge base 4. (prolog variables, variable instantiation)

Clause:

```
woman(rupa).
woman(joba).
woman(jui).
loves(viki, rupa).
loves(marin, rupa).
loves(potato, honey).
loves(honey, potato).
```

Queries:

1. woman(X).
2. loves(marin,X), woman(X).
3. loves(potato, X), woman(X).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/knowledge base 4.pl compiled 0.00 se
c, 7 clauses
?- woman(X).
X = rupa ;
X = joba ;
X = jui.

?- loves(marin,X),woman(X).
X = rupa.

?- loves(potato,X),woman(X).
false.
```

Example 5:

Problem Name: Knowledge base 5.

Clause:

```
loves(viki,rupa).
loves(marin,rupa).
loves(potato, honey).
loves(honey, potato).
jealous(X,Y):- loves(X,Z), loves(Y,Z).
```

Queries:

1. jealous(marin, Y).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/knowledge base 5.pl compiled 0.00 se
c, 5 clauses
?- jealous(marin,Y).
Y = viki ;
Y = marin.
```

Example 6:

Problem Name: Find out different relationships from the family tree.

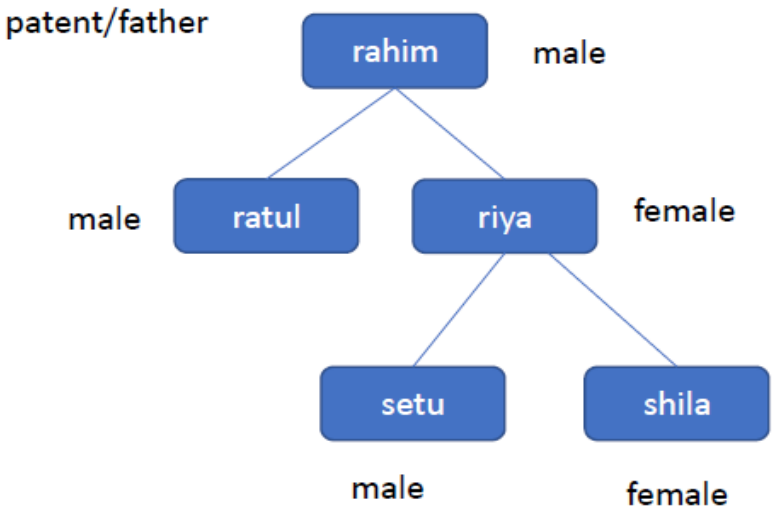


Fig-1: family tree[from given slide].

Clause:

```
male(rahim).
male(ratul).
male(setu).
female(riya).
female(shila).
parents(rahim,ratul).
parents(rahim,riya).
parents(riya,setu).
parents(riya,shila).
father(X,Y) :- parents(X,Y),male(X).
mother(X,Y) :- parents(X,Y),female(X).
sibling(X,Y):-parents(Z,X),parents(Z,Y),X\=Y.
grandfather(X,Z):-parents(X,Y),parents(Y,Z),male(X).
grandmother(X,Z):-parents(X,Y),parents(Y,Z),female(X).
```

Queries:

1. father(X,shila).
2. father(X,Y).

3. mother(X,shila).
4. mother(X,Y).
5. sibling(X,Y).
6. sibling(X,shila).
7. grandfather(X,shila).
8. grandfather(X,Y).
9. grandmother(X,shila).
10. grandmother(X,Y).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Family tree.pl compiled 0.00 sec, 14 clauses
?- father(X,shila).
false.

?- father(X,Y).
X = rahim,
Y = ratul ;
X = rahim,
Y = riya ;
false.

?- mother(X,shila).
X = riya.

?- mother(X,Y).
X = riya,
Y = setu ;
X = riya,
Y = shila.


?- sibling(X,shila).
X = setu ;
false.

?- sibling(X,Y).
X = ratul,
Y = riya ;
X = riya,
Y = ratul ;
X = setu,
Y = shila ;
X = shila,
Y = setu ;
false.

?- grandfather(X,shila).
X = rahim .

?- grandfather(X,Y).
X = rahim,
Y = setu ;
X = rahim,
Y = shila ;
false.

?- grandmother(X,shila).
false.

?- grandmother(X,Y).
false.
```

Example 7:

Problem Name : Addition/Subtraction/Multiplication/Division/Power

Clause:

%no need any clause

Queries:

1. X is 10+7+5.
2. X is 13-8.
3. X is 2*3.
4. X is 9/3.
5. X is 2^3.

Result:

```
?- X is 10+7+5.
X = 22.

?- X is 13-8.
X = 5.

?- X is 2*3.
X = 6.

?- X is 9/3.
X = 3.

?- X is 2^3.
X = 8.
```

Example 8:

Problem Name : Min/Max value.

Clause:

%no need any clause

Queries:

1. X is max(99,57).
2. X is min(68,19).

Result:

```
?- X is max(99,57).
X = 99.

?- X is min(68,19).
X = 19.
```

Example 9:

Problem Name: Use of underscore.

Clause:

```
%no need any clause
```

Queries:

1. division(A,B,C,D).
2. division(_,X,_,Y).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/underscore.pl compiled 0.00 sec, 1 claus
es
?- division(A,B,C,D).
A = dhaka,
B = rajshahi,
C = khulna,
D = sylhet.

?- division(_,X,_,Y).
X = rajshahi,
Y = sylhet.
```

Example 10:

Problem Name: Input two numbers & print them. (Exercise 1)

Clause:

```
func1:-
write('enter the first number'),nl,
read(X),nl,
write('enter the second number'),nl,
read(Y),nl,
write('The numbers are:'),nl,
write(X),nl,
write(Y).
```

Queries:

- func1.

  33.

  57.

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Excercise 1.pl compiled 0.00 sec, 1 clau
ses
?- func1.
enter the first number
|: 33.

enter the second number
|: 57.

The numbers are:
33
57
true.
```

Example 11:

Problem Name: Sum of two numbers. (Exercise 2)

Clause:

```
func2:-
write('enter the first number:'),nl,
read(A),nl,
write('enter the second number:'),nl,
read(B),nl,
sum(A,B).
sum(A,B):-S is A+B,
write('sum is: '),
write(S).
```

Queries:

- func2.

  7.

  9.

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Excercise 2.pl compiled 0.00 sec, 2
clauses
?- func2.
enter the first number:
|: 7.

enter the second number:
|: 9.

sum is: 16
true.
```

Example 12:

Problem Name: Input a string and print it. (Exercise 3)

Clause:

```
take_input_string(Input) :-
write('Enter a string: '),
read_line_to_codes(user_input, Codes),
string_codes(Input, Codes).

process_string(String) :-
string_upper(String, Output),
write('Output: '), write(Output).

main :-
take_input_string(Input),
process_string(Input).
```

Queries:

- take_input_string(X).

  sanjida.

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Exercise 3.pl compiled 0.00 sec, 0 c
lauses
?- take_input_string(X).
Enter a string: sanjida.
X = "sanjida.".
```

Example 13:

Problem Name: Find average from 3 numbers. (Task 1 from slide)

Clause:

```
take_input_string(Input) :-
write('Enter a string: '),
read_line_to_codes(user_input, Codes),
string_codes(Input, Codes).
process_string(String) :-
string_upper(String, Output),
write('Output: '), write(Output).

main :-
take_input_string(Input),
process_string(Input).
```

Queries:

- func3.

  3.

  7.

  5.

Result:

```
% f:/4th Year, 1st Semester/AI LAB/averag of 3 numbers.pl compiled 0.00
 sec, 2 clauses
?- func3.
enter the 1st number:
|: 3.

enter the 2nd number:
|: 7.

enter the 3rd number:
|: 5.

Average is : 5
true.
```

Example 14:

Problem Name: Find brother & sister rule from the family tree. (Task 2 from slide)

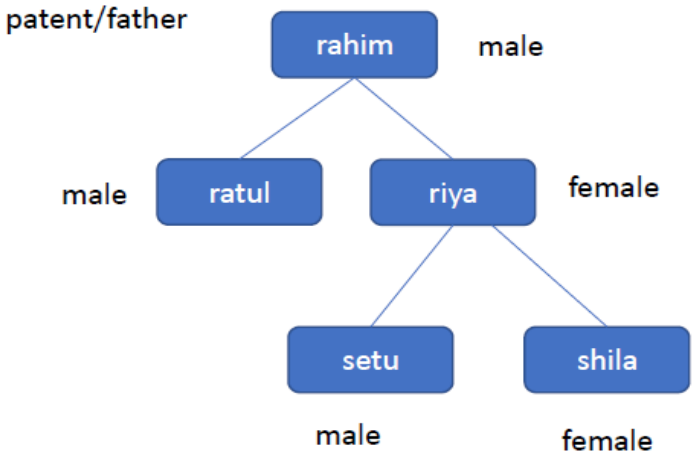


Fig-2: family tree [from given slide]

Clause:

```
male(rahim).
male(ratul).
male(setu).
female(riya).
female(shila).
parents(rahim,ratul).
parents(rahim,riya).
```

```
parents(riya,setu).
parents(riya,shila).
sibling(X,Y):-parents(Z,X),parents(Z,Y),X\=Y.
brother(X,Y):-sibling(X,Y),male(X).
sister(X,Y):-sibling(X,Y),female(X).
```

Queries:

1. brother(X,ria).
2. brother(X,setu).
3. brother(X,Y).
4. sister(X,setu).
5. sister(X,shila).
6. sister(X,Y).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Sibling rules.pl compiled 0.00 sec,
0 clauses
?- brother(X,riya).
X = ratul ;
false.

?- brother(X,setu).
false.

?- brother(X,Y).
X = ratul,
Y = riya ;
X = setu,
Y = shila ;
false.
```

```
?- sister(X,setu).
X = shila.

?- sister(X,shila).
false.

?- sister(X,Y).
X = riya,
Y = ratul ;
X = shila,
Y = setu ;
false.
```

Example 15:

Problem Name: How would you formulate the following queries from the following figure?

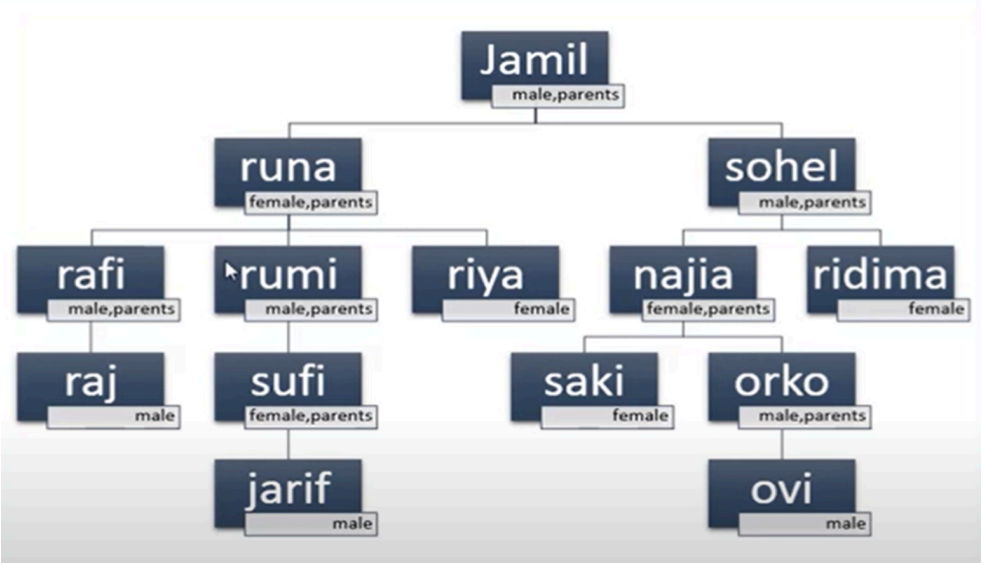


Fig-3: family tree [task-01]

•is runa is male?
•is sohel is male?
•is jarif is male?
•is sufi is female?
•is ridima is female?
•is jamil is parents?
•is sufi is parents?
•is saki is parents?
•is rumi is parents?
•find out runa's children name.
•find out jamil's children name.
•find out rafi's sibling's name.
•find out najia's sibling's name.
•find out who is riya's mother.
•find out who is orko's mother.

Clause:

```
male(jamil).
male(sohel).
male(rafi).
male(raj).
male(orko).
male(jarif).
male(ovi).
female(runa).
female(ria).
female(najia).
```

```
female(ridima).
female(sufi).
female(saki).
parents(jamil,runa).
parents(jamil,sohel).
parents(runa,rafi).
parents(sohel,najia).
parents(rafi,raj).
parents(runa,rumi).
parents(runa,riya).
parents(sohel,ridima).
parents(rumi,sufi).
parents(najia,saki).
parents(najia,orko).
parents(sufi,jarif).
parents(orko,ovi).
father(X,Y) :- parents(X,Y),male(X).
mother(X,Y) :- parents(X,Y),female(X).
siblings(X,Y):-parents(Z,X),parents(Z,Y),X\=Y.
childrens(X,Y):-parents(Y,X).
isparents(X):-childrens(Y,X).
```

Queries:

1. male(runa).
2. male(sohel).
3. male(jarif).
4. female(sufi).
5. female(ridima).
6. isparents(jamil).
7. isparents(sufi).
8. isparents(saki).
9. isparents(rumi).
10. childrens(X,runa).
11. childrens(X,jamil).
12. siblings(X,rafi).
13. siblings(X,najia).
14. mother(X,riya).
15. mother(X,orko).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/family tree(2).pl compiled 0.00 sec,
 -2 clauses
?- male(runa).
false.

?- male(sohel).
true.

?- male(jarif).
true.

?- female(sufi).
true.

?- female(ridima).
true.


?- isparents(jamil).
true .

?- isparents(sufi).
true.

?- isparents(saki).
false.

?- isparents(rumi).
true.


?- childrens(X,runa).
X = rafi ;
X = rumi ;
X = riya.

?- childrens(X,jamil).
X = runa ;
X = sohel.

?- siblings(X,rafi).
X = rumi ;
X = riya ;
false.

?- siblings(X,najia).
X = ridima ;
false.

?- mother(X,riya).
X = runa.

?- mother(X,orko).
X = najia.
```

Institute of Information Technology (IIT)

Jahangirnagar University

Lab Report: 02

Submitted by:

Name: Sanjida Akter

Roll No: 1982

Lab Date: 08-06-2023

Submission Date: 12-06-2023

Lab Report # Day 02

Exercise Task 1:

Problem Name: Implement a Prolog predicate equal_length /2' that takes two lists as input and succeeds if both lists have the same length. Give some example queries and their expected outputs.

Clauses:

```
list_input :-
   nl,
   write('Enter the first list: '),
   nl,
   read(X),
   nl,
    write('Enter the second list: '),
    nl,
   read(Y),
   nl,
  equal_length(X,Y).
  equal_length([],[]).
  equal_length([_|T1],[_|T2]) :- equal_length(T1,T2).
```

Queries:

1. list_input.

   [].

   [].

2. list_input.

   [b , [a, [b, c]]].

   [[b,c], [a,[b,c]]].

3. list_input.

   [rahim, karim, anna].

   [5,7].

4. list_input.

   [1,3,5].

   [].

Result:

```
?- list_input.

Enter the first list:
|: [].

Enter the second list:
|: [].

true.

?- list_input.

Enter the first list:
|: [b,[a,[b,c]]].

Enter the second list:
|: [[b,c],[a,[b,c]]].

true.

?- list_input.

Enter the first list:
|: [rahim,karim,anna].

Enter the second list:
|: [5,7].

false.

?- list_input.

Enter the first list:
|: [1,3,5].

Enter the second list:
|: [].

false.
```

Exercise Task 2:

Problem Name: Write a Prolog predicate ‘maximum/3 ’ that takes three integers as input and returns the maximum of the three.

Clauses:

```
start:-
   nl,
```

```
   write('Enter the 1st number: '),
   nl,
   read(X),
 nl,
   write('Enter the 2nd number: '),
   nl,
   read(Y),
 nl,
   write('Enter the 3rd number: '),
   nl,
   read(Z),
   maximum(X,Y,Z).

maximum(X, Y, Z) :-
   X >= Y,
   X >= Z,
   nl,
   write('Maximum number is: '),
   write(X),nl.
maximum(X, Y, Z) :-
   Y >= X,
   Y >= Z,
   nl,
   write('Maximum number is: '),
   write(Y),nl.
maximum(X, Y, Z) :-
  Z >= X,
   Z>=Y,
   nl,
   write('Maximum number is: '),
   write(Z),nl.
```

Queries:

1. start.

   13. 9. 17.

2. start.

   19. 7. 11.

3. start.

   15. 29. 13.

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Maximum of 3.pl compiled 0.00 sec, 0 clauses
?- start.

Enter the 1st number:
|: 13.

Enter the 2nd number:
|: 9.

Enter the 3rd number:
|: 17.

Maximum number is: 17
true.

?- start.

Enter the 1st number:
|: 19.

Enter the 2nd number:
|: 7.

Enter the 3rd number:
|: 11.

Maximum number is: 19
true .

?- start.

Enter the 1st number:
|: 15.

Enter the 2nd number:
|: 29.

Enter the 3rd number:
|: 13.

Maximum number is: 29
true .
```

Exercise Task 3:

Problem Name: Write a Prolog predicate to find the length of a list.

Clauses:

```
list_length([], 0).
list_length([_ | T], Len) :- list_length(T, TL), Len is TL + 1.
```

Queries:

1. list_length([anna,[ria,mila],rabeya],Length).
2. list_length([a,[b,[c,d]]],Length).
3. list_length([],Length).
4. list_length([w,7,arian,s],Length).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Length of a list.pl compiled 0.00 se
c, 0 clauses
?- list_length([anna,[ria,mila],rabeya],Length).
Length = 3.

?- list_length([a,[b,[c,d]]],Length).
Length = 2.

?- list_length([],Length).
Length = 0.

?- list_length([w,7,arian,s],Length).
Length = 4.
```

Example 1:

Problem Name: Comparison in prolog

Clauses:

```
%group-A
goal(morocco,2).
goal(germany,3).
goal(france,1).

%group-B
goal(spain,2).
goal(portugal,5).
goal(japan,1).

solve:-
write('enter a country name(from group-A): '),
nl,
read(X),
goal(X,Y),
nl,
write('Group-A country score is: '),
write(Y),
nl,nl,
write('enter a country name(from group-B): '),
nl,
read(A),
goal(A,B),
nl,
write('Group-B country score is: '),
write(B),
nl,
compare(X,Y,A,B).
compare(X,Y,A,B):-
Y>B,nl,
```

```
write(X),
```

Queries:

1. solve.

   -france.

   -portugal.

2. solve.

   -germany.

   -japan.

3. solve.

   -spain.

   -morocco.

Result:

```
% f:/4th Year, 1st Semester/AI LAB/comparison.pl compiled 0.00 sec, 0 clauses
?- solve.
enter a country name(from group-A):
|: france.

Group-A country score is: 1

enter a country name(from group-B):
|: portugal.

Group-B country score is: 5

portugal from group-B is the winner!
true .

?- solve.
enter a country name(from group-A):
|: germany.

Group-A country score is: 3

enter a country name(from group-B):
|: japan.

Group-B country score is: 1

germany from group-A is the winner!
true .

?- solve.
enter a country name(from group-A):
|: spain.

Group-A country score is: 2

enter a country name(from group-B):
|: morocco.

Group-B country score is: 2

Draw in both group!
true.
```

Example 2:

Problem Name: Backtracking (possible pairs).

Clauses:

```
boy(rahim).
boy(karim).
girl(anika).
girl(jui).
pair(X,Y):-boy(X),girl(Y).
```

Queries:

1. pair(X,Y).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/possible pair.pl compiled 0.00 sec, 5 clauses
?- pair(X,Y).
X = rahim,
Y = anika ;
X = rahim,
Y = jui ;
X = karim,
Y = anika ;
X = karim,
Y = jui.
```

Example 3:

Problem Name: Recursion (Eating).

Clauses:

```
justAte(mosquito,blood(karim)).
justAte(frog,mosquito).
justAte(stork,frog).
isDigesting(X,Y):- justAte(X,Y).
isDigesting(X,Y):- justAte(X,Z),isDigesting(Z,Y).
```

Queries:

1. isDigesting(stork,mosquito).
2. isDigesting(frog,blood(karim)).
3. isDigesting(mosquito,frog).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Eating(Recursion).pl compiled 0.00 s
ec, 5 clauses
?- isDigesting(stork,mosquito).
true .

?- isDigesting(frog,blood(karim)).
true .

?- isDigesting(mosquito,frog).
false.
```

Example 4:

Problem Name: Factorial (using recursion).

Clauses:

```
factorial(0, 1).
factorial(N, Result):-
N > 0,
N1 is N-1,
factorial(N1, SubResult),
Result is N * SubResult.
```

Queries:

1. factorial(5,Ans).
2. factorial(0,Ans).
3. factorial(20,Ans).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Factorial(Recursion).pl compiled 0.0
0 sec, 0 clauses
?- factorial(5,Ans).
Ans = 120 .

?- factorial(0,Ans).
Ans = 1 .

?- factorial(20,Ans).
Ans = 2432902008176640000 .
```

Example 5:

Problem Name: Check descendent (relative connection).

Clauses:

```
child(maria,jerin).
child(karim,keya).
child(keya,jhumur).
child(jhumur,emu).
descend(X,Y):-child(X,Y).
descend(X,Y):-child(X,Z), descend(Z,Y).
```

Queries:

1. descend(anika,dina).
2. descend(X,Y).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/descendant(relative connection).pl c
ompiled 0.00 sec, 0 clauses
?- descend(anika,dina).
false.

?- descend(X,Y).
X = anika,
Y = bithi ;
X = bithi,
Y = keya ;
X = keya,
Y = dina ;
X = dina,
Y = emu ;
X = anika,
Y = keya ;
X = bithi,
Y = dina ;
X = keya,
Y = emu ;
false.
```

Example 6:

Problem Name: Check descendent (in recursive way).

Clauses:

```
child(anika,bithi).
child(bithi,keya).
child(keya,dina).
child(dina,emu).
descend(X,Y):-child(X,Y).
descend(X,Y):-child(X,Z), descend(Z,Y).
```

Queries:

1. descend(anika,dina).
2. descend(X,Y).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Descendant(Recursion).pl compiled 0.
00 sec, 6 clauses
?- descend(anika,dina).
true .

?- descend(X,Y).
X = anika,
Y = bithi ;
X = bithi,
Y = keya ;
X = keya,
Y = dina ;
X = dina,
Y = emu ;
X = anika,
Y = keya ;
X = anika,
Y = dina ;
X = anika,
Y = emu ;
X = bithi,
Y = dina ;
X = bithi,
Y = emu ;
X = keya,
Y = emu ;
false.
```

[Note: We can find more relation in recursive way than manual way that's why though there have same queries for both example 5 & 6, but the outcome is different]

Example 7:

Problem Name: Head and Tail of a list.

Clauses:

```
% no clauses are needed in this regard
```

Queries:

1. [Head|Tail]=[ana,bina,rina,payel].
2. [A|B]=[dead(X)].
3. [X|Y]=[].
4. [Head|Tail]=[[],dead(Z),[2,[b,c]],[],Z,[2,[b,c]]].

Result:

```
?- [Head|Tail]=[ana,bina,rina,payel].
Head = ana,
Tail = [bina, rina, payel].

?- [A|B]=[dead(X)].
A = dead(X),
B = [].

?- [X|Y]=[].
false.

?- [Head|Tail]=[[],dead(Z),[2,[b,c]],[],Z,[2,[b,c]]].
Head = [],
Tail = [dead(Z), [2, [b, c]], [], Z, [2, [b, c]]].
```

Example 8:

Problem Name: Anonymous variable from list.

Clauses:

```
% no clauses are needed in this regard
```

Queries:

1. [X,Y|Tail]=[[],karim(blood),ayesha].
2. [A,B,C,D|T]=[ana,bithe,rabeya,akash,nila,zeba].

Result:

```
?- [X,Y|Tail]=[[],karim(blood),ayesha].
X = [],
Y = karim(blood),
Tail = [ayesha].

?- [A,B,C,D|T]=[ana,bithe,rabeya,akash,nila,zeba].
A = ana,
B = bithe,
C = rabeya,
D = akash,
T = [nila, zeba].
```

Example 9:

Problem Name: Some operation on lists (membership).

Clauses:

```
%no clauses are needed in this regard
```

Queries:

1. member(b,[a,b,c]).
2. member(b,[a,[b,c]]).
3. member([b,c],[a,[b,c]]).

Result:

```
?- member(b,[a,b,c]).
true .

?- member(b,[a,[b,c]]).
false.

?- member([b,c],[a,[b,c]]).
true.
```

Example 10:

Problem Name: Some operation on lists (concatenation).

Clauses:

```
concat_lists(L1, L2, Res):- append(L1, L2, Res).
```

Queries:

1. concat_lists([],[a,b,c],Res).
2. concat_lists([ana,bina],[a,[b,3]],Res).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Concatenation.pl compiled 0.00 sec,
1 clauses
?- concat_lists([],[a,b,c],Res).
Res = [a, b, c].

?- concat_lists([ana,bina],[a,[b,3]],Res).
Res = [ana, bina, a, [b, 3]].
```

Example 11:

Problem Name: Some operation on lists (deleting an item from list).

Clauses:

```
delete_item(X,[X|Tail],Tail).
delete_item(I,[Y|Tail],[Y|Tail1]):- delete_item(I,Tail,Tail1).
```

Queries:

1. delete_item(a,[d,b,a,c],NL).
2. delete_item(a,[d,b,[a,c]],NL).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Delete from list.pl compiled 0.00 se
c, 1 clauses
?- delete_item(a,[d,b,a,c],NL).
NL = [d, b, c] ;

?- delete_item(a,[d,b,[a,c]],NL).
false.
```

Example 12:

Problem Name: Some operation on lists (adding an item).

Clauses:

```
add_item(Item, List, NewList ):- append(List, [Item], NewList).
```

Queries:

1. add_item(feriha,[x,[1,2,3],[],jeny],Res).
2. add_item([],[x,jeny],Res).

Result:

```
% f:/4th Year, 1st Semester/AI LAB/Adding an item in the list.pl compil
ed 0.00 sec, 1 clauses
?- add_item(feriha,[x,[1,2,3],[],jeny],Res).
Res = [x, [1, 2, 3], [], jeny, feriha].

?- add_item([],[x,jeny],Res).
Res = [x, jeny, []].
```

Institute of Information Technology (IIT)

Jahangirnagar University

Lab Report: 03

Submitted by:

Name: Sanjida Akter

Roll No: 1982

Lab Date: 13-06-2023

Submission Date: 22-06-2023

Lab Report # Day 03

Exercise 01:

Write a Python program to find the sum of all the elements in a list.

Code:

```
sum=0
list=[7,3,-5,3,11]
for i in range(0,len(list)):
    sum=sum + list[i]
print("Sample List is: ",list)
print("Sum of the list elements is: ",sum)
```

Output:

```
Sample List is:  [7, 3, -5, 3, 11]
Sum of the list elements is:  19
```

Exercise 02:

Write a Python program to find the largest, smallest, second largest, and second smallest elements in a list.

Code:

```
X=[7,0,4,-1,23,11,5]
print("Sample List is: ",X)
X.sort()
print("Largest: ",X[-1])
print("Smallest: ",X[0])
print("Second Largest: ",X[-2])
print("Second Smallest: ",X[1])
```

Output:

```
Sample List is:  [7, 0, 4, -1, 23, 11, 5]
Largest:  23
Smallest:  -1
Second Largest:  11
Second Smallest:  0
```

Exercise 03:

Write a Python program to count the number of occurrences of each character in a string.

Code:

```
str=input('Sample String is: ')
s=set(str)
for i in s:
    cnt=0
    for j in str:
        if(i==j):
            cnt+=1
    print("Occurrence of '{}' is {}".format(i, cnt))
```

Output:

```
Sample String is: desserts
Occurrence of 'd' is 1
Occurrence of 't' is 1
Occurrence of 'r' is 1
Occurrence of 'e' is 2
Occurrence of 's' is 3
```

Exercise 04:

Write a Python program to create a tuple with elements from a list and print it.

Code:

```
X=[9,[],2,[3,'a'],'hello']
t=tuple(X)
print("Sample list is: ",X)
print("Tuple: ",t)
```

Output:

```
Sample list is:  [9, [], 2, [3, 'a'], 'hello']
Tuple:  (9, [], 2, [3, 'a'], 'hello')
```

Exercise 05:

Write a Python function that takes a list of numbers as input and returns the largest sum of non-adjacent numbers.

Code:

```
def solve(a):
    size = len(a)
    if size <= 2:
        return max(a)
    x = 0
    y = a[0]
    for i in range(1, size):
        y, x = x + a[i],max(x,y)
    return max(x,y)

L = []
n = int(input("Enter list size: "))
print("Enter list elements:")
for i in range(0, n):
    val = int(input())
    L.append(val)
print("List:",L)
print("Largest sum of non-adjacent numbers is:", solve(L))
```

Output:

```
Enter list size: 6
Enter list elements:
4
7
2
1
9
3
List: [4, 7, 2, 1, 9, 3]
Largest sum of non-adjacent numbers is: 16
```

Exercise 06:

Write a Python program to remove duplicates from a list and return the resultant list.

Code:

```
A=[1,2,2,1,3,1,5,3,2]
B=set(A)
Res=[]
for i in B:
    Res.append(i)
print("Sample List:",A)
print("Resultant List(After removing duplicates):",Res)
```

Output:

```
Sample List: [1, 2, 2, 1, 3, 1, 5, 3, 2]
Resultant List(After removing duplicates): [1, 2, 3, 5]
```

Exercise 07:

Write a Python program to find the common elements between two lists and return the resultant list.

Code:

```
def solve(a, b):
   res = [i for i in a if i in b]
   return res

a = [1, 2, 3, 4, 5]
b = [5, 6, 4, 8, 9]
print("First List:",a)
print("Second List:",b)
print("Common elements between two lists are: ",solve(a,b))
```

Output:

```
First List: [1, 2, 3, 4, 5]
Second List: [5, 6, 4, 8, 9]
Common elements between two lists are:  [4, 5]
```

Exercise 08:

Write a Python program to find the first n Fibonacci numbers using recursion.

Code:

```
def solve(n):
  if n <= 1:
     return n
  else:
     return(solve(n-1) + solve(n-2))

n= int(input("How many numbers? "))
print("Fibonacci sequence is:")
for i in range(n):
   print(solve(i))
```

Output:

```
How many numbers? 6
Fibonacci sequence is:
0
1
1
2
3
5
```

Exercise 09:

Write a Python function to replace all occurrences of a substring in a string.

Code:

```
s = "Some men"
s1 = "me"
s2 = "Nice"
print("Sample String: ",s)
print("Selected substring: ",s1)
print("Substring replaced by: ",s2)
s = s.replace(s1, s2)
print("Resultant String:",s)
```

Output:

```
Sample String:  Some men
Selected substring:  me
Substring replaced by:  Nice
Resultant String: SoNice Nicen
```

Exercise 10:

Write a function to add a key-value pair to a dictionary in Python

Code:

```
CountryGoal = {"Brazil": 4, "France" : 2 , "Portugal" : 3}
print('Sample dictionary:',CountryGoal,'\n')
CountryGoal["Spain"]= 34
print ("After adding new:",CountryGoal)
```

Output:

```
Sample dictionary: {'Brazil': 4, 'France': 2, 'Portugal': 3}

After adding new: {'Brazil': 4, 'France': 2, 'Portugal': 3, 'Spain': 34}
```

Exercise 11:

Write a function to remove a key from a dictionary in Python.

Code:

```
CountryGoal = {"Brazil": 4, "France" : 2 , "Portugal" : 3}
print('Sample dictionary:',CountryGoal,'\n')
del CountryGoal['France']
print("After Remove:",CountryGoal)
```

Output:

```
Sample dictionary: {'Brazil': 4, 'France': 2, 'Portugal': 3}

After Remove: {'Brazil': 4, 'Portugal': 3}
```

Exercise 12:

Write a function to reverse a list of numbers.

Code:

```
 L=[3,5,7,[],[1,0],9]
print("Sample List:",L)
L.reverse()
print("Reverse:",L)
```

Output:

```
Sample List: [3, 5, 7, [], [1, 0], 9]
Reverse: [9, [1, 0], [], 7, 5, 3]
```

Exercise 13:

Write a Python program to find and print the key with the maximum value in a dictionary.

Code:

```
CountryGoal = {"Brazil": 4, "France" : 2 , "Portugal" : 3}
print('Sample dictionary:',CountryGoal,'\n')
max_key=max(CountryGoal,key=lambda i:CountryGoal[i])
print("Key with maximum value is:",max_key)
```

Output:

```
Sample dictionary: {'Brazil': 4, 'France': 2, 'Portugal': 3}

Key with maximum value is: Brazil
```

Exercise 14:

Write a Python program to merge two dictionaries and create a new dictionary.

Code:

```
X = {"Brazil": 4, "France" : 2 , "Portugal" : 3}
print('First dictionary:',X)
Y = {"Spain": 2, "Italy": 1}
print("Second dictionary:",Y)
X.update(Y)
print("New Merged dictionary is:",X)
```

Output:

```
First dictionary: {'Brazil': 4, 'France': 2, 'Portugal': 3}
Second dictionary: {'Spain': 2, 'Italy': 1}
New Merged dictionary is: {'Brazil': 4, 'France': 2, 'Portugal': 3, 'Spain': 2, 'Italy': 1}
```

Exercise 15:

Given a list of dictionaries, you want to sort them based on a specific key 'age' in each dictionary. Write a lambda function as the key parameter in the sorted() function to achieve this.

Code:

```
D = [{"name": "Fatema", "age": 23},
     {"name": "Miraj", "age": 20},
     {"name": "Nil", "age": 19},
   {"name": "Riya", "age": 20}]
print("Original list:")
print(D,"\n")
```

```
print("The list printed sorting by age(ascending order): ")
print(sorted(D, key=lambda i: i['age']),)
print("\r")
print("The list printed sorting by age(descending order): ")
print(sorted(D, key=lambda i: i['age'], reverse=True))
```

Output:

```
Original list:
[{'name': 'Fatema', 'age': 23}, {'name': 'Miraj', 'age': 20}, {'name': 'Nil', 'age': 19}, {'name': 'Riya', 'age': 20}]

The list printed sorting by age(ascending order):
[{'name': 'Nil', 'age': 19}, {'name': 'Miraj', 'age': 20}, {'name': 'Riya', 'age': 20}, {'name': 'Fatema', 'age': 23}]

The list printed sorting by age(descending order):
[{'name': 'Fatema', 'age': 23}, {'name': 'Miraj', 'age': 20}, {'name': 'Riya', 'age': 20}, {'name': 'Nil', 'age': 19}]
```

Institute of Information Technology (IIT)

Jahangirnagar University

Lab Report: 04

Submitted by:

Name: Sanjida Akter

Roll No: 1982

Lab Date: 01-08-2023

Submission Date: 11-08-2023

Lab Report # Day 08

Given Exercise:

Apply linear regression techniques to predict housing prices based on the provided features in the attached Dataset.

The column name of the attached dataset is given below with a description:

CRIM: Per capita crime rate by town
ZN: Proportion of residential land zoned for lots over 25,000 sq. ft
INDUS: Proportion of non-retail business acres per town
CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX: Nitric oxide concentration (parts per 10 million)
RM: Average number of rooms per dwelling
AGE: Proportion of owner-occupied units built prior to 1940
DIS: Weighted distances to five Boston employment centers
RAD: Index of accessibility to radial highways
TAX: Full-value property tax rate per $10,000
PTRATIO: Pupil-teacher ratio by town
B: 1000(Bk — 0.63)², where Bk is the proportion of [*people of African American descent*] by town
LSTAT: Percentage of lower status of the population
MEDV: Median value of owner-occupied homes in $1000s. *[Target Variable]*

The prices of the house indicated by the variable MEDV is our and the remaining are the based on which we will predict the value of a house.

#Check out the data

1.Import Libraries

Code:

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

2.Check out the data

Code:

```
hs =pd.read_csv('housing.csv')
hs.head()
```

Output:

	0.00632 18.00 2.310 0 0.5380 6.5750 65.20 4.0900 1 296.0 15.30 396.90 4.98 24.00
0	0.02731 0.00 7.070 0 0.4690 6.4210 78...
1	0.02729 0.00 7.070 0 0.4690 7.1850 61...
2	0.03237 0.00 2.180 0 0.4580 6.9980 45...
3	0.06905 0.00 2.180 0 0.4580 7.1470 54...
4	0.02985 0.00 2.180 0 0.4580 6.4300 58...

3. Including column names & check out the data again

Code:

```
column = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
'PTRATIO', 'B', 'LSTAT', 'MEDV']
data = pd.read_csv('housing.csv', header=None, delimiter=r"\s+", names=column)
data.head()
```

Output:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

Code:

```
data.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

Code:

data.describe()

Output:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.0(
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.6
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.1
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.7
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.9
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.3
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.9
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.97

Code:

data.columns

Output:

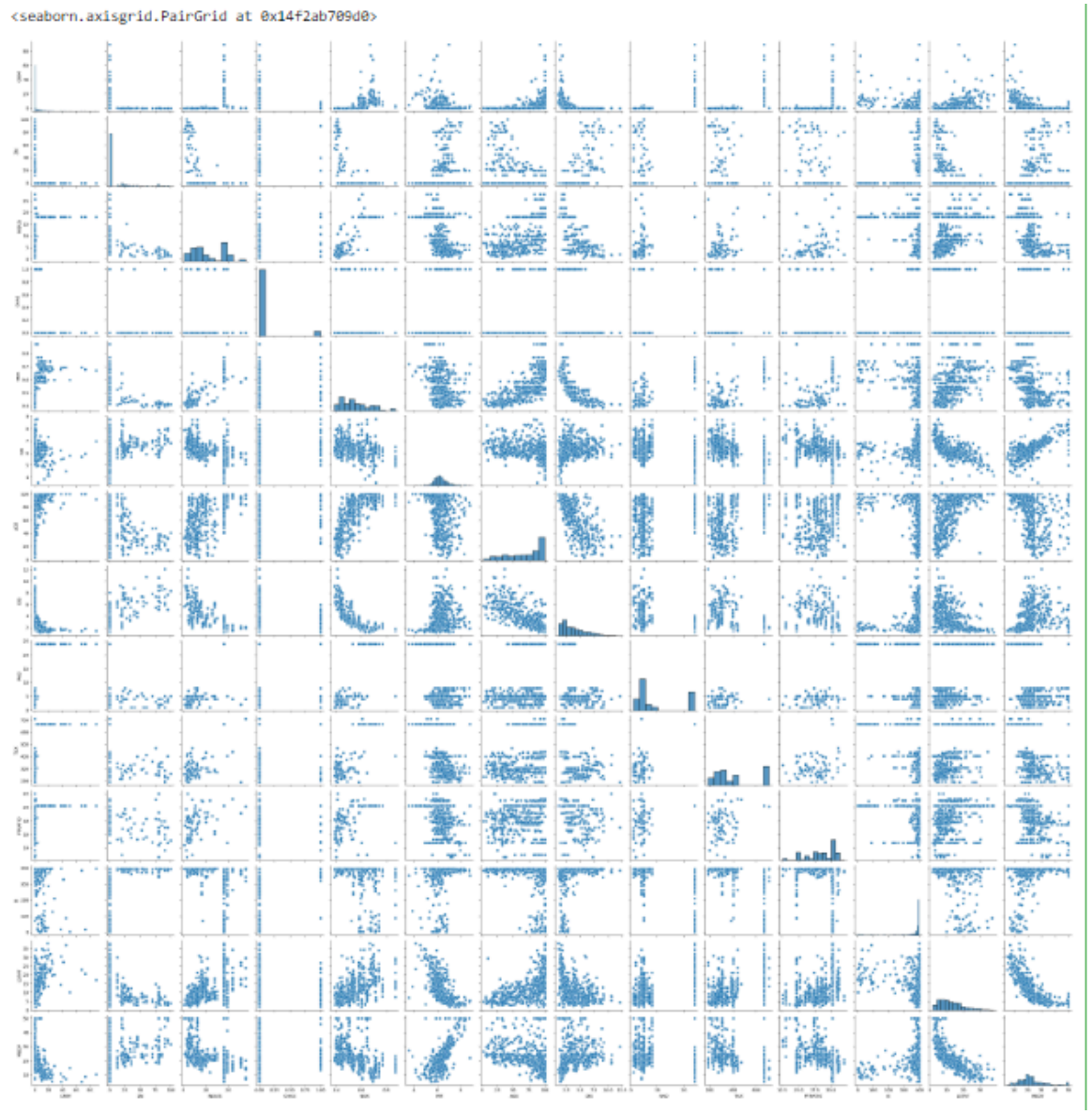
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

#Exploratory data analysis

Code:

```
sns.pairplot(data)
```

Output:

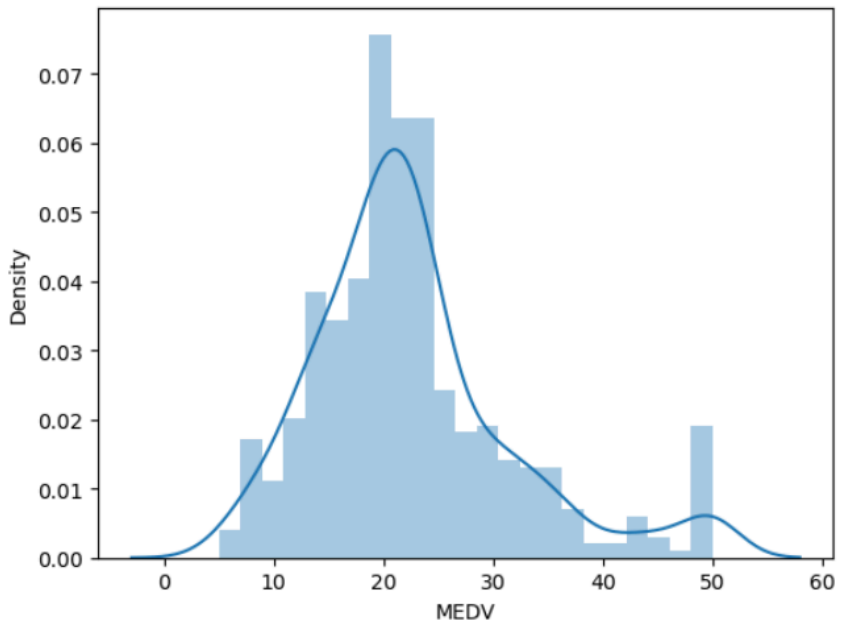


Code:

```
sns.distplot(data['MEDV'])
```

Output:

```
<Axes: xlabel='MEDV', ylabel='Density'>
```



Code:

```
house=data[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV']]
house
```

Output:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

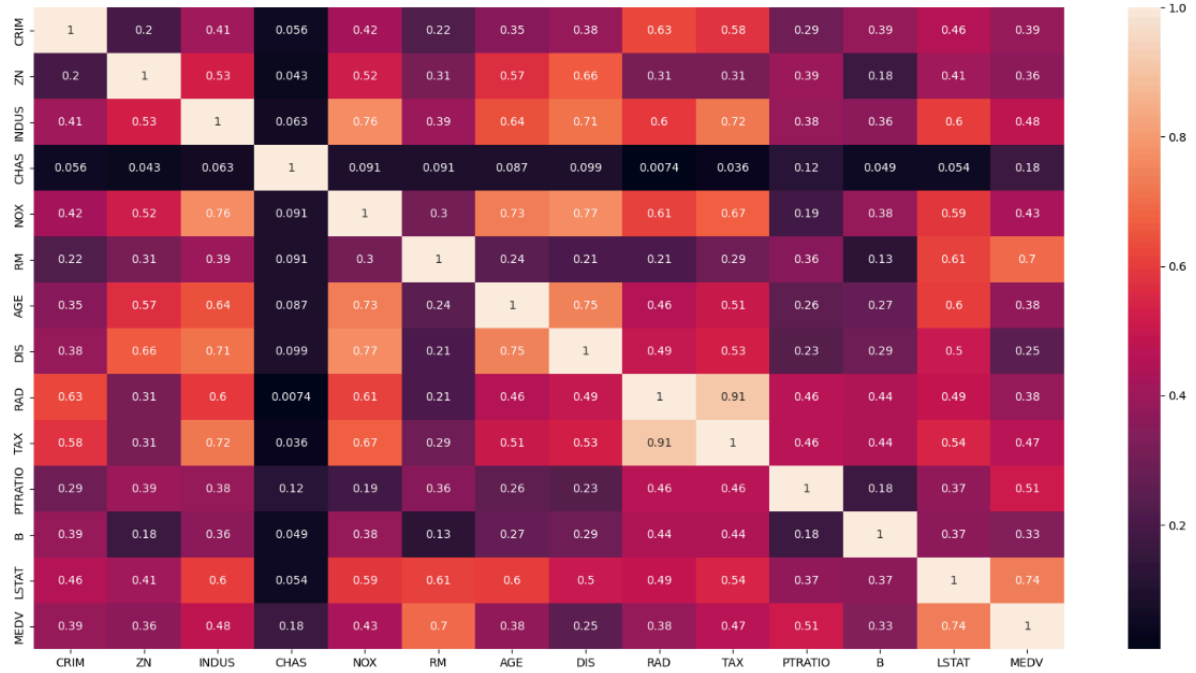
506 rows × 14 columns

Code:

```
plt.figure(figsize=(20,10))
sns.heatmap(data.corr().abs(),annot=True)
```

Output:

<Axes: >



#Training a linear regression model

1.X and Y arrays

Code:

```
x=data[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT']]
y=data['MEDV']
```

#Train Test Split

Code:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=42)
```

#Creating and Training the Model

Code:

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(x_train,y_train)
```

Output:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

#Model Evaluation

Code:

```
print(lm.intercept_)
```

Output:

```
31.63108403569186
```

Code:

```
coeff = pd.DataFrame(lm.coef_,x.columns,columns=["Coefficient"])
coeff
```

Output:

	Coefficient
CRIM	-0.133470
ZN	0.035809
INDUS	0.049523
CHAS	3.119835
NOX	-15.417061
RM	4.057199
AGE	-0.010821
DIS	-1.385998
RAD	0.242727
TAX	-0.008702
PTRATIO	-0.910685
B	0.011794
LSTAT	-0.547113

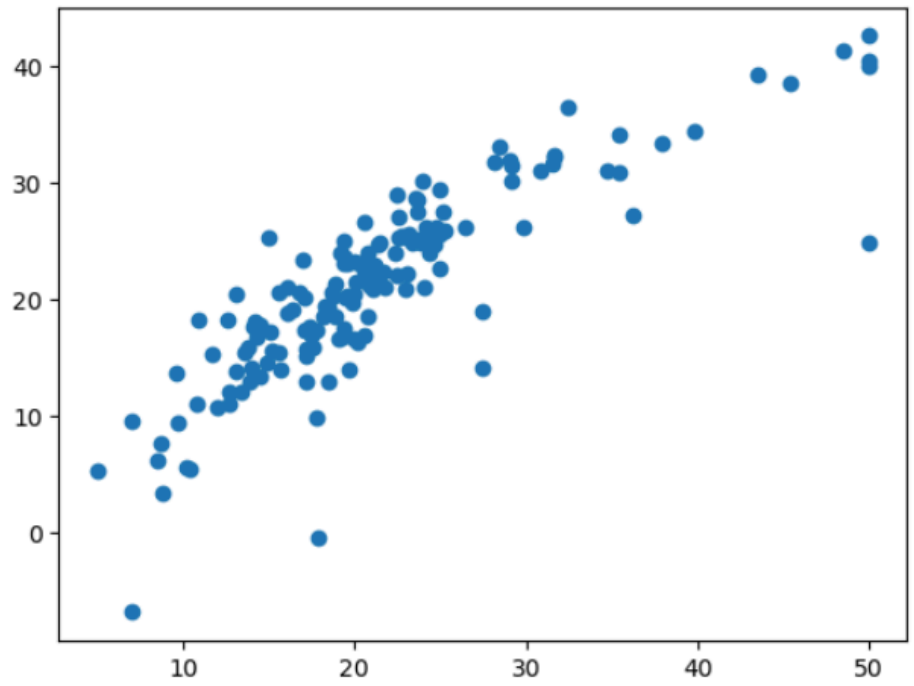
#Prediction from our Model

Code:

```
predictions=lm.predict(x_test)
plt.scatter(y_test,predictions)
```

Output:

```
<matplotlib.collections.PathCollection at 0x14f3eac0f40>
```



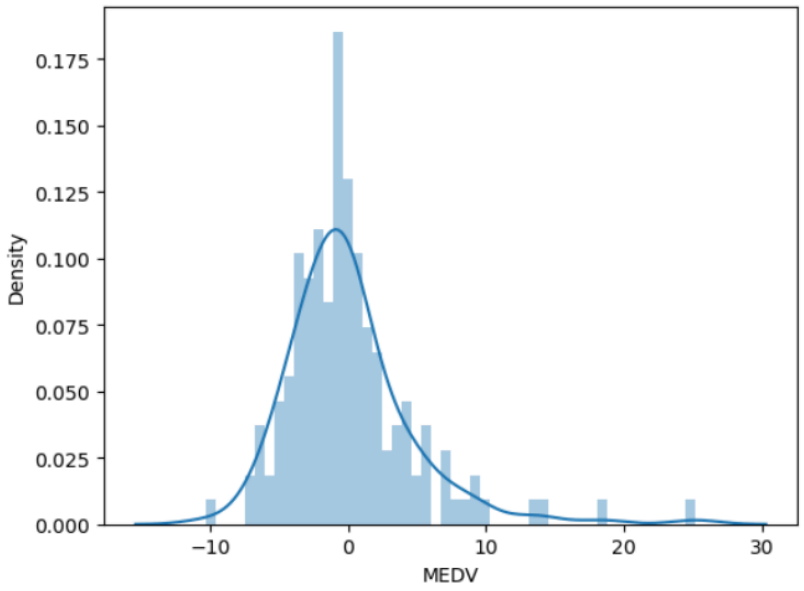
Residual Histogram

Code:

```
sns.distplot((y_test-predictions),bins=50)
```

Output:

```
<Axes: xlabel='MEDV', ylabel='Density'>
```

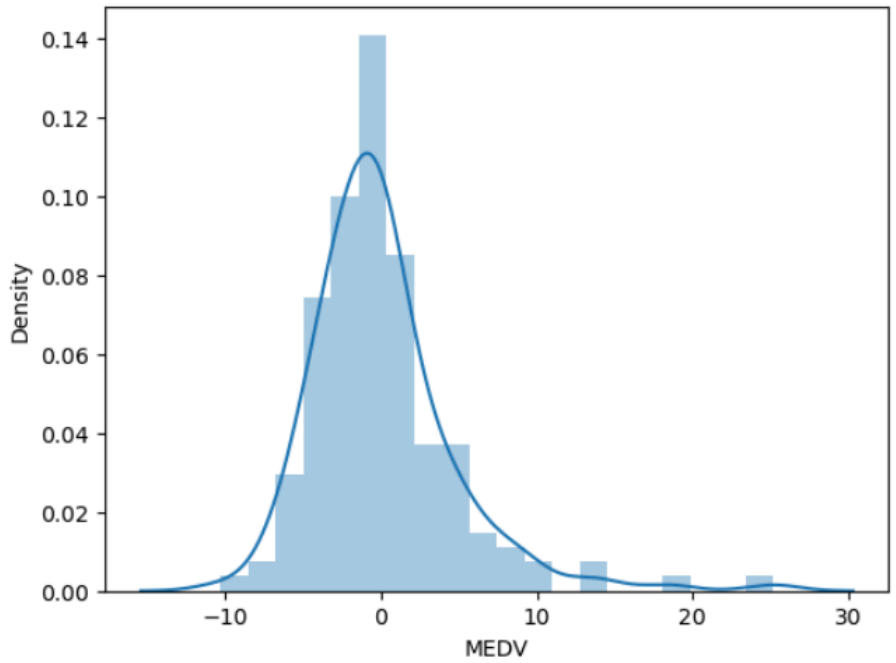


Code:

```
sns.distplot((y_test-predictions),kde=True,bins=20)
```

Output:

```
<Axes: xlabel='MEDV', ylabel='Density'>
```



#Regression Evaluation Metrics

Code:

```
from sklearn import metrics
print('MAE :',metrics.mean_absolute_error(y_test,predictions))
print('MSE :',metrics.mean_squared_error(y_test,predictions))
print('RMSE :',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

Output:

```
MAE : 3.1627098714574253
MSE : 21.517444231177432
RMSE : 4.6386899261728445
```

Institute of Information Technology (IIT)

Jahangirnagar University

Lab Report: 05

Submitted by:

Name: Sanjida Akter

Roll No: 1982

Lab Date: 08-08-2023

Submission Date: 18-08-2023

Lab Report # Day 09

Given Task:

Analysis using Logistic Regression and predict the 10-year risk of coronary heart disease.

Step 1: Importing Required Libraries

Code:

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

Step 2: Data Preparation and Description

Code:

```
df =pd.read_csv('framingham.csv')
df.head()
```

Output:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	Ten
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	

Code:

```
df.columns
```

Output:

```
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

Code:

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4238 non-null   int64
 1   age              4238 non-null   int64
 2   education        4133 non-null   float64
 3   currentSmoker    4238 non-null   int64
 4   cigsPerDay       4209 non-null   float64
 5   BPMeds           4185 non-null   float64
 6   prevalentStroke  4238 non-null   int64
 7   prevalentHyp     4238 non-null   int64
 8   diabetes         4238 non-null   int64
 9   totChol          4188 non-null   float64
 10  sysBP            4238 non-null   float64
 11  diaBP            4238 non-null   float64
 12  BMI              4219 non-null   float64
 13  heartRate        4237 non-null   float64
 14  glucose          3850 non-null   float64
 15  TenYearCHD       4238 non-null   int64
dtypes: float64(9), int64(7)
memory usage: 529.9 KB
```

Code:

df.describe()

Output:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP
count	3656.000000	3656.000000	3656.000000	3656.000000	3656.000000	3656.000000	3656.000000	3656.000000	3656.000000	3656.000000	3656.000000
mean	0.443654	49.557440	1.979759	0.489059	9.022155	0.030361	0.005744	0.311543	0.027079	236.873085	132.368025
std	0.496883	8.561133	1.022657	0.499949	11.918869	0.171602	0.075581	0.463187	0.162335	44.096223	22.092444
min	0.000000	32.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	113.000000	83.500000
25%	0.000000	42.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	206.000000	117.000000
50%	0.000000	49.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	234.000000	128.000000
75%	1.000000	56.000000	3.000000	1.000000	20.000000	0.000000	0.000000	1.000000	0.000000	263.250000	144.000000
max	1.000000	70.000000	4.000000	1.000000	70.000000	1.000000	1.000000	1.000000	1.000000	600.000000	295.000000

Step 3: Exploratory Data Analysis

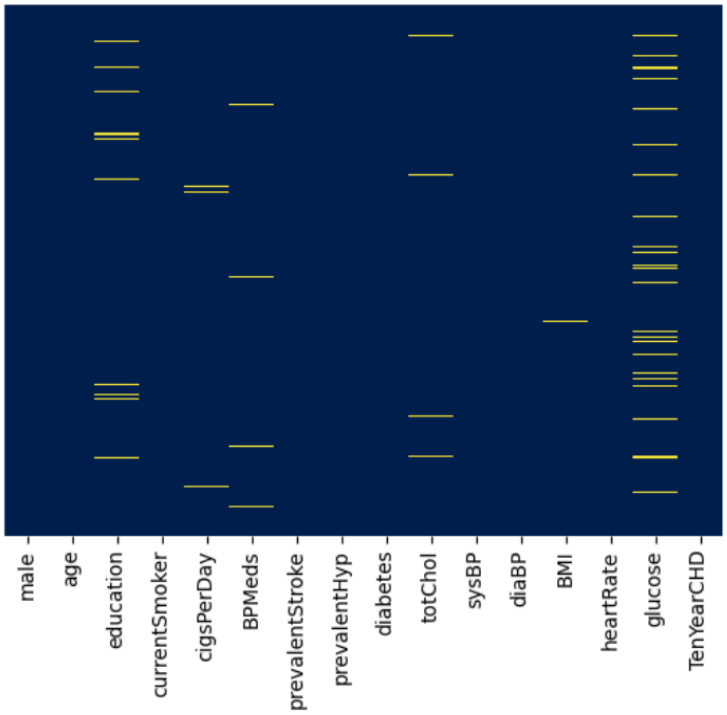
#Missing Data

Code:

sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='cividis')

Output:

```
<Axes: >
```



Code:

```
df.isna().sum()
```

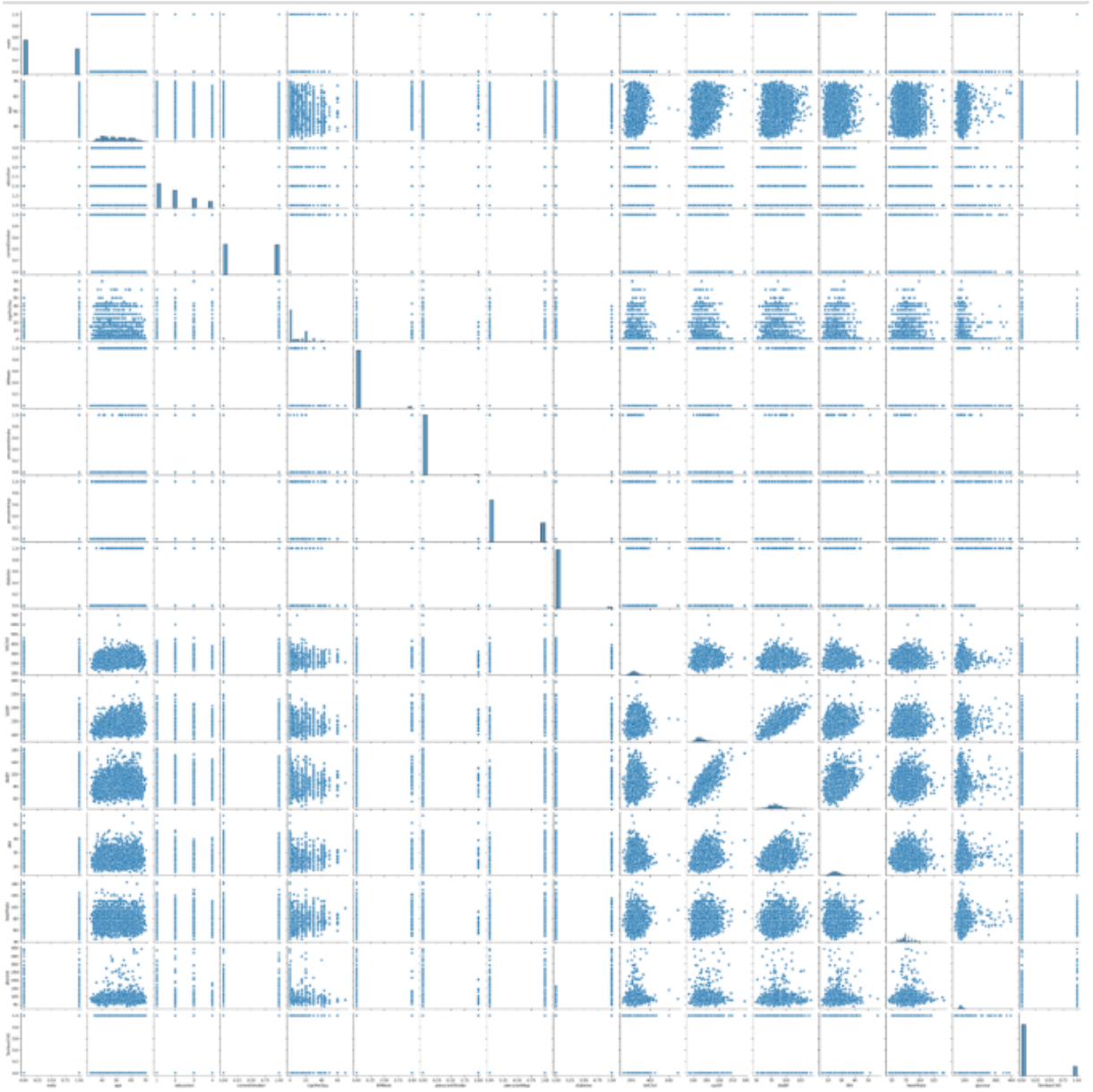
Output:

```
male               0
age                0
education        105
currentSmoker      0
cigsPerDay        29
BPMeds            53
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol           50
sysBP              0
diaBP              0
BMI               19
heartRate          1
glucose          388
TenYearCHD         0
dtype: int64
```

Code:

```
sns.pairplot(df)
```

Output:

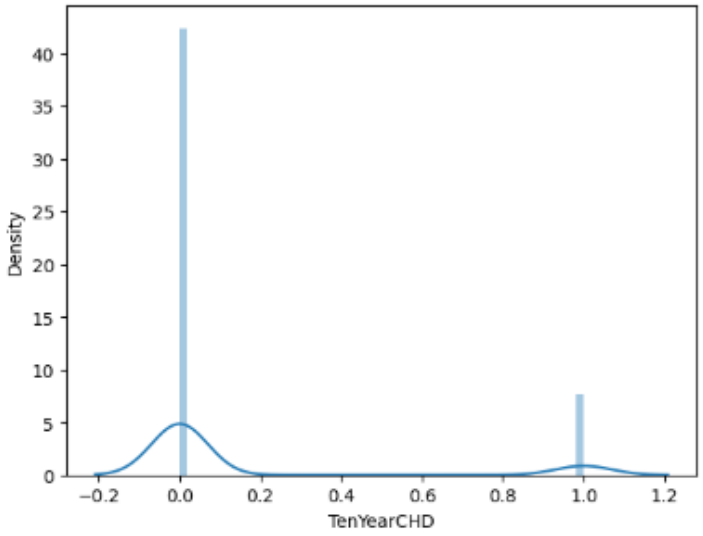


Code:

```
sns.distplot(df['TenYearCHD'])
```

Output:

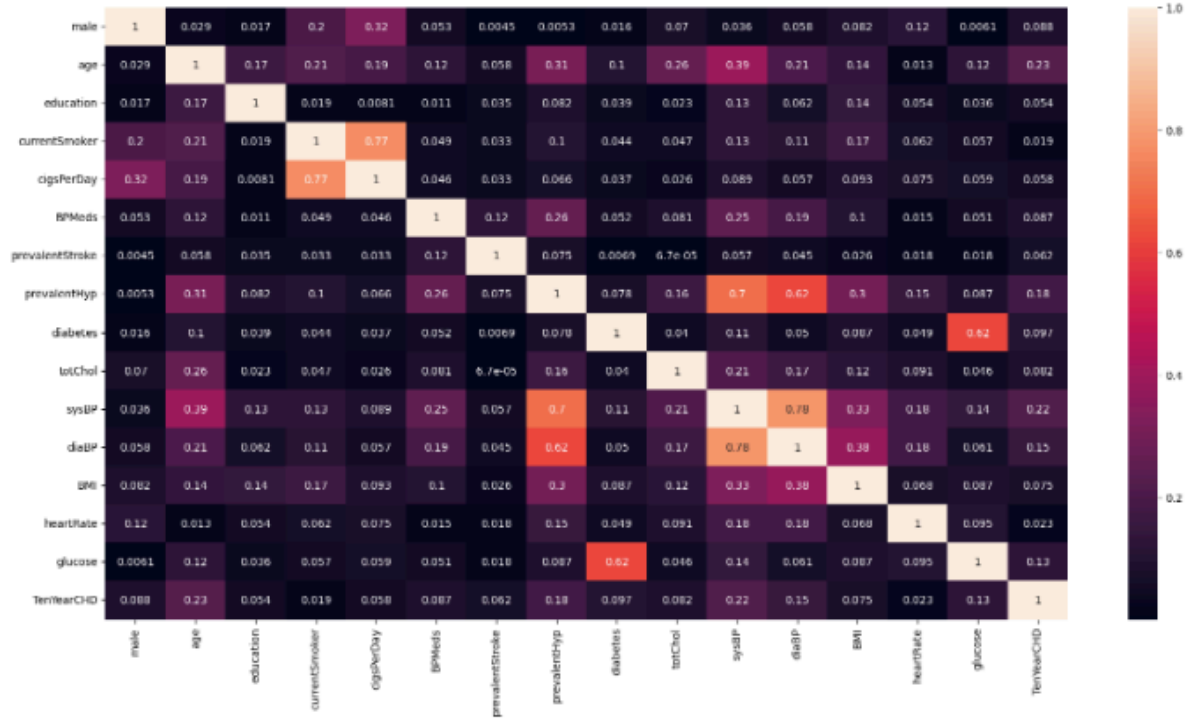
```
<Axes: xlabel='TenYearCHD', ylabel='Density'>
```



Code:

```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr().abs(),annot=True)
```

Output:

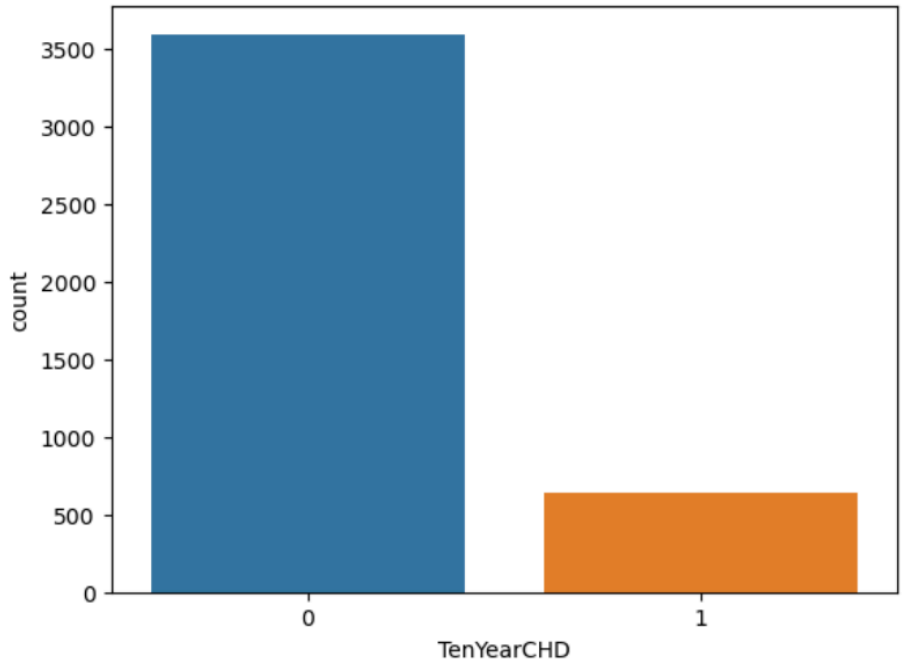


Step 4: Exploratory Visual Analysis

Code:

```
sns.countplot(x='TenYearCHD', data=df)
```

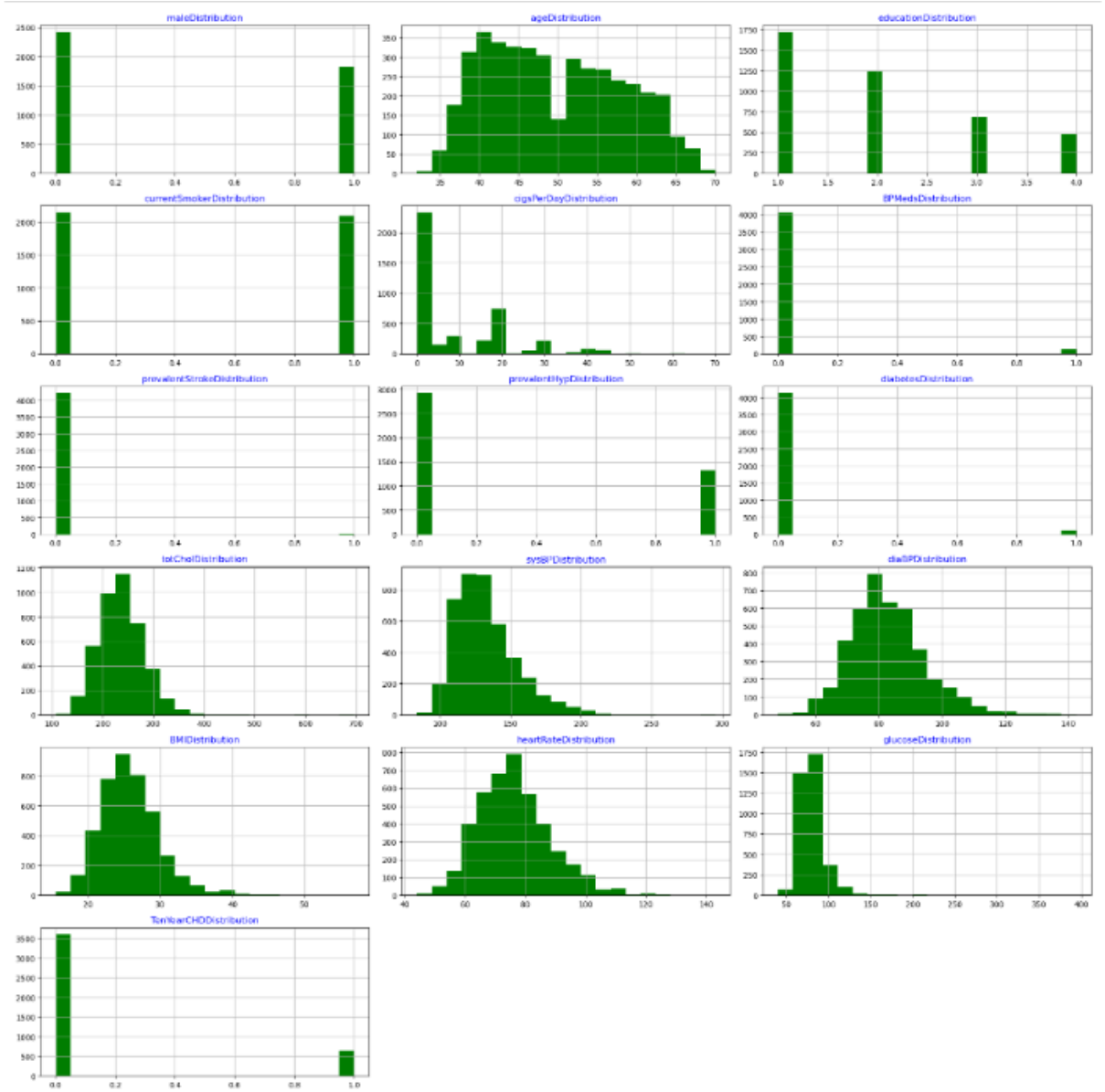
Output:



Code:

```
def draw_histograms(dataframe, features, rows, cols):
    fig=plt.figure(figsize=(20,20))
    for i, feature in enumerate(features):
        ax=fig.add_subplot(rows,cols,i+1)
        dataframe[feature].hist(bins=20,ax=ax,facecolor='green')
        ax.set_title(feature+"Distribution", color='blue')
    fig.tight_layout()
    plt.show()
draw_histograms(df, df.columns, 6, 3)
```

Output:



Step 5: Building a Logistic Regression Model

Code:

```
X = df.iloc[:,0:15]
y = df.iloc[:,15:16]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=21)
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

Output:

```
▾ LogisticRegression
LogisticRegression()
```

Step 6: Model Evaluation

Code:

```
print(logreg.intercept_)
```

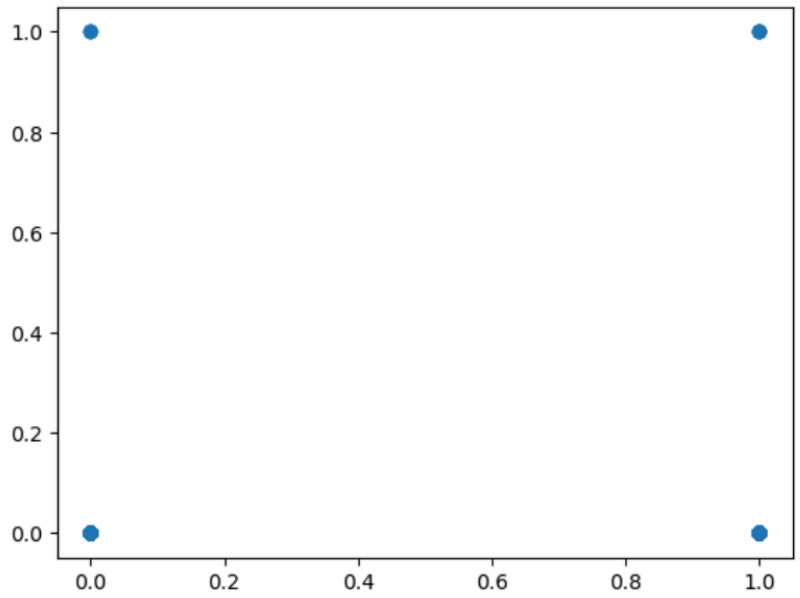
Output:

```
[-0.14796842]
```

Code:

```
predictions=logreg.predict(X_test)
plt.scatter(y_test,predictions)
```

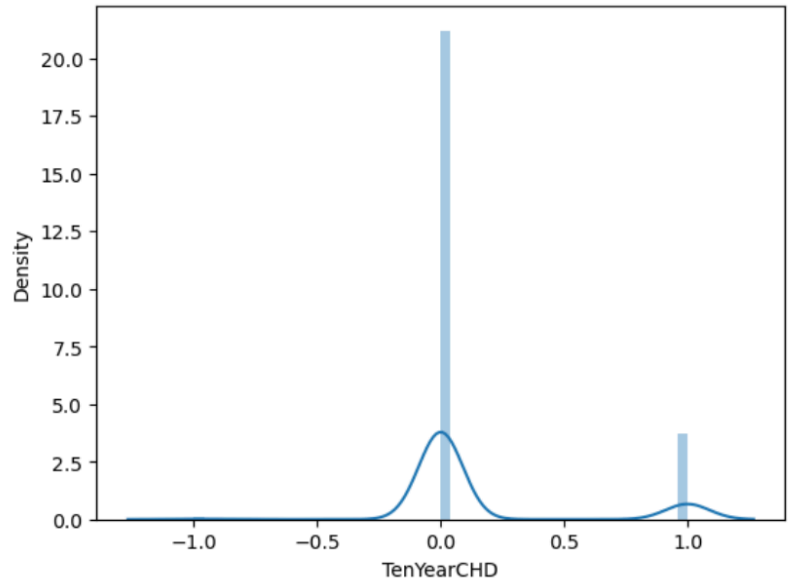
Output:



Code:

```
sns.distplot((y_test-predictions),bins=50)
```

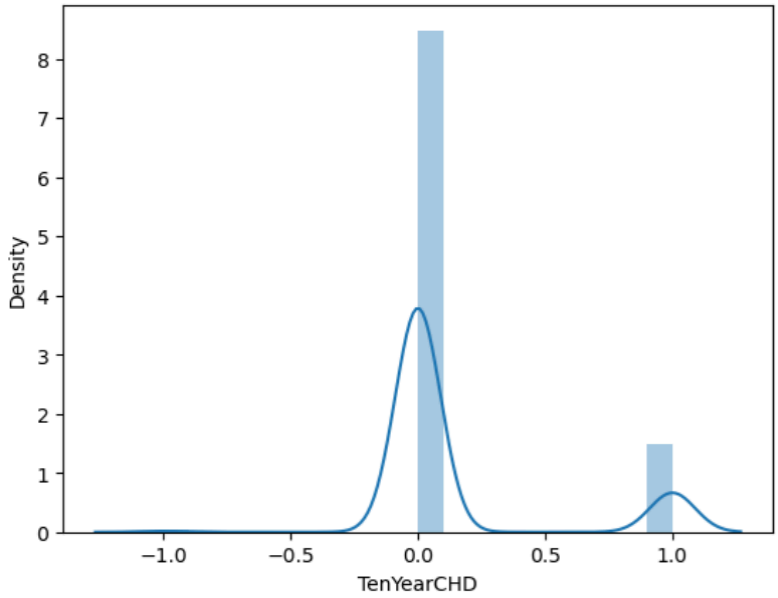
Output:



Code:

```
sns.distplot((y_test-predictions),kde=True,bins=20)
```

Output:



Code:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
accuracy = accuracy_score(y_test,y_pred)
precision = precision_score(y_test,y_pred)
recall = recall_score(y_test,y_pred)
f1 = f1_score(y_test, y_pred)
print("Accuracy is:", accuracy)
```

```
print("Precision is:", precision)
print("Recall is:", recall)
print("F1 Score is:", f1)
```

Output:

```
Accuracy is: 0.8477666362807658
Precision is: 0.6
Recall is: 0.035502958579881656
F1 Score is: 0.06703910614525141
```

Code:

```
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix is:\n",cm)
```

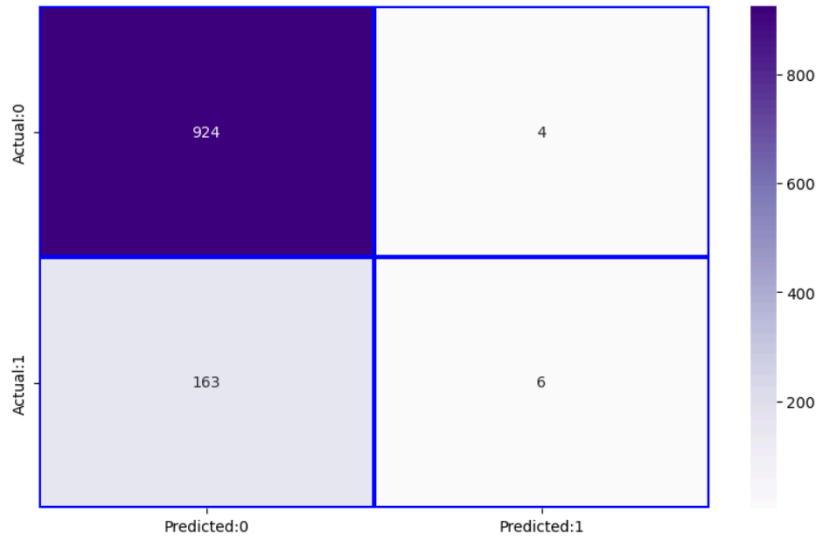
Output:

```
Confusion Matrix is:
 [[924   4]
 [163   6]]
```

Code:

```
conf_matrix = pd.DataFrame(data = cm,
columns = ['Predicted:0', 'Predicted:1'],
index =['Actual:0', 'Actual:1'])
plt.figure(figsize = (10, 6))
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Purples", linecolor="Blue",
linewidths=1.5)
plt.show()
```

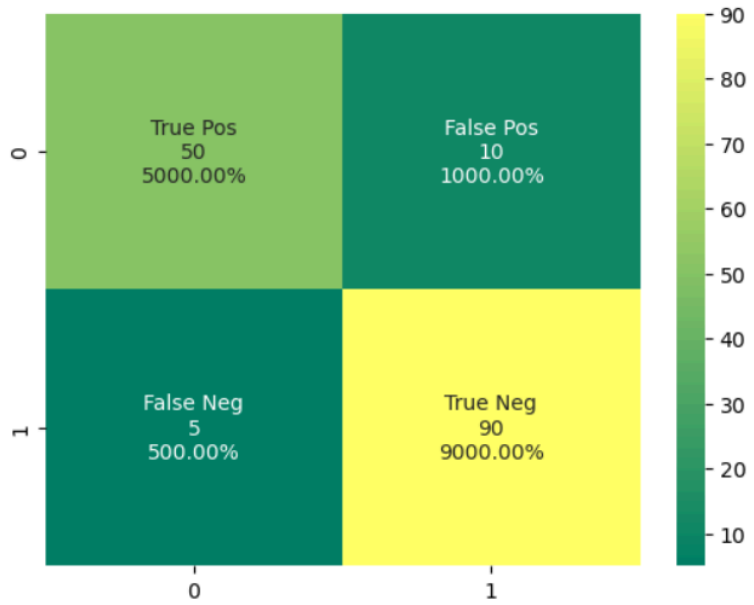
Output:



Code:

```
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_pred)
conf_matrix = np.array([[50, 10], [5, 90]])
group_names = ['True Pos','False Pos','False Neg','True Neg']
group_counts = ["{0:0.0f}".format(value) for value in conf_matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in conf_matrix.flatten()]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
 zip(group_names, group_counts, group_percentages)]
labels = np.asarray(labels).reshape(2, 2)
sns.heatmap(conf_matrix, annot=labels, fmt='', cmap='summer')
plt.show()
```

Output:



Institute of Information Technology (IIT)

Jahangirnagar University

Lab Report: 06

Submitted by:

Name: Sanjida Akter

Roll No: 1982

Lab Date: 22-08-2023

Submission Date: 29-08-2023

Lab Report # Day 10

Given Task:

Analysis using K-NN using any dataset.

Dataset Information:

This dataset is a collection of basic health biological signal data. The goal is to determine the presence or absence of smoking through bio-signals.

Columns:

data shape : (55692, 27)

- ID : index
- gender
- age : 5-years gap
- height(cm)
- weight(kg)
- waist(cm) : Waist circumference length
- eyesight(left)
- eyesight(right)
- hearing(left)
- hearing(right)
- systolic : Blood pressure
- relaxation : Blood pressure
- fasting blood sugar
- Cholesterol : total
- triglyceride
- HDL : cholesterol type
- LDL : cholesterol type
- hemoglobin
- Urine protein
- serum creatinine
- AST : glutamic oxaloacetic transaminase type
- ALT : glutamic oxaloacetic transaminase type
- Gtp : γ-GTP
- oral : Oral Examination status
- dental caries
- tartar : tartar status
- smoking

Step 1: Importing Required Libraries

Code:

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Step 2: Data Preparation and Description

Code:

```
df=pd.read_csv("smoking.csv",index_col=0)
df
```

Output:

ID	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	hemoglobin	Urine protein	serum creatinine	A
0	F	40	155	60	81.3	1.2	1.0	1.0	1.0	114.0	...	12.9	1.0	0.7	1
1	F	40	160	60	81.0	0.8	0.6	1.0	1.0	119.0	...	12.7	1.0	0.6	2
2	M	55	170	60	80.0	0.8	0.8	1.0	1.0	138.0	...	15.8	1.0	1.0	2
3	M	40	165	70	88.0	1.5	1.5	1.0	1.0	100.0	...	14.7	1.0	1.0	1
4	F	40	155	60	86.0	1.0	1.0	1.0	1.0	120.0	...	12.5	1.0	0.6	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
55676	F	40	170	65	75.0	0.9	0.9	1.0	1.0	110.0	...	12.3	1.0	0.6	1
55681	F	45	160	50	70.0	1.2	1.2	1.0	1.0	101.0	...	14.0	1.0	0.9	2
55683	F	55	160	50	68.5	1.0	1.2	1.0	1.0	117.0	...	12.4	1.0	0.5	1
55684	M	60	165	60	78.0	0.8	1.0	1.0	1.0	133.0	...	14.4	1.0	0.7	2
55691	M	55	160	65	85.0	0.9	0.7	1.0	1.0	124.0	...	15.0	1.0	0.8	2

55692 rows × 26 columns

waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	hemoglobin	Urine protein	serum creatinine	AST	ALT	Gtp	oral	dental caries	tartar	smoking
81.3	1.2	1.0	1.0	1.0	114.0	...	12.9	1.0	0.7	18.0	19.0	27.0	Y	0	Y	0
81.0	0.8	0.6	1.0	1.0	119.0	...	12.7	1.0	0.6	22.0	19.0	18.0	Y	0	Y	0
80.0	0.8	0.8	1.0	1.0	138.0	...	15.8	1.0	1.0	21.0	16.0	22.0	Y	0	N	1
88.0	1.5	1.5	1.0	1.0	100.0	...	14.7	1.0	1.0	19.0	26.0	18.0	Y	0	Y	0
86.0	1.0	1.0	1.0	1.0	120.0	...	12.5	1.0	0.6	16.0	14.0	22.0	Y	0	N	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
75.0	0.9	0.9	1.0	1.0	110.0	...	12.3	1.0	0.6	14.0	7.0	10.0	Y	1	Y	0
70.0	1.2	1.2	1.0	1.0	101.0	...	14.0	1.0	0.9	20.0	12.0	14.0	Y	0	Y	0
68.5	1.0	1.2	1.0	1.0	117.0	...	12.4	1.0	0.5	17.0	11.0	12.0	Y	0	N	0
78.0	0.8	1.0	1.0	1.0	133.0	...	14.4	1.0	0.7	20.0	19.0	18.0	Y	0	N	0
85.0	0.9	0.7	1.0	1.0	124.0	...	15.0	1.0	0.8	26.0	29.0	41.0	Y	0	Y	1

Code:

```
df.shape
```

Output:

```
(55692, 26)
```

Code:

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 55692 entries, 0 to 55691
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   gender               55692 non-null  object
 1   age                  55692 non-null  int64
 2   height(cm)           55692 non-null  int64
 3   weight(kg)           55692 non-null  int64
 4   waist(cm)            55692 non-null  float64
 5   eyesight(left)       55692 non-null  float64
 6   eyesight(right)      55692 non-null  float64
 7   hearing(left)        55692 non-null  float64
 8   hearing(right)       55692 non-null  float64
 9   systolic             55692 non-null  float64
 10  relaxation           55692 non-null  float64
 11  fasting blood sugar  55692 non-null  float64
 12  Cholesterol          55692 non-null  float64
 13  triglyceride         55692 non-null  float64
 14  HDL                  55692 non-null  float64
 15  LDL                  55692 non-null  float64
 16  hemoglobin           55692 non-null  float64
 17  Urine protein        55692 non-null  float64
 18  serum creatinine     55692 non-null  float64
 19  AST                  55692 non-null  float64
 20  ALT                  55692 non-null  float64
 21  Gtp                  55692 non-null  float64
 22  oral                 55692 non-null  object
 23  dental caries        55692 non-null  int64
 24  tartar               55692 non-null  object
 25  smoking              55692 non-null  int64
dtypes: float64(18), int64(5), object(3)
memory usage: 11.5+ MB
```

Code:

```
df.columns
```

Output:

```
Index(['gender', 'age', 'height(cm)', 'weight(kg)', 'waist(cm)',
       'eyesight(left)', 'eyesight(right)', 'hearing(left)', 'hearing(right)',
       'systolic', 'relaxation', 'fasting blood sugar', 'Cholesterol',
       'triglyceride', 'HDL', 'LDL', 'hemoglobin', 'Urine protein',
       'serum creatinine', 'AST', 'ALT', 'Gtp', 'oral', 'dental caries',
       'tartar', 'smoking'],
      dtype='object')
```

Step 3: Converting String column to Integer

Code:

```
df['gender'].replace({'F':0, 'M':1}, inplace=True)
#df['gender'] = pd.factorize(df['gender'])[0].astype(float)
df['oral'].replace({'N':0, 'Y':1}, inplace=True)
df['tartar'].replace({'N':0, 'Y':1}, inplace=True)
df
```

Output:

ID	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	hemoglobin	Urine protein	serum creatinine
0	0	40	155	60	81.3	1.2	1.0	1.0	1.0	114.0	...	12.9	1.0	0.7
1	0	40	160	60	81.0	0.8	0.6	1.0	1.0	119.0	...	12.7	1.0	0.6
2	1	55	170	60	80.0	0.8	0.8	1.0	1.0	138.0	...	15.8	1.0	1.0
3	1	40	165	70	88.0	1.5	1.5	1.0	1.0	100.0	...	14.7	1.0	1.0
4	0	40	155	60	86.0	1.0	1.0	1.0	1.0	120.0	...	12.5	1.0	0.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
55676	0	40	170	65	75.0	0.9	0.9	1.0	1.0	110.0	...	12.3	1.0	0.6
55681	0	45	160	50	70.0	1.2	1.2	1.0	1.0	101.0	...	14.0	1.0	0.9
55683	0	55	160	50	68.5	1.0	1.2	1.0	1.0	117.0	...	12.4	1.0	0.5
55684	1	60	165	60	78.0	0.8	1.0	1.0	1.0	133.0	...	14.4	1.0	0.7
55691	1	55	160	65	85.0	0.9	0.7	1.0	1.0	124.0	...	15.0	1.0	0.8

55692 rows × 26 columns

waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	hemoglobin	Urine protein	serum creatinine	AST	ALT	Gtp	oral	dental caries	tartar	smoking
81.3	1.2	1.0	1.0	1.0	114.0	...	12.9	1.0	0.7	18.0	19.0	27.0	1	0	1	0
81.0	0.8	0.6	1.0	1.0	119.0	...	12.7	1.0	0.6	22.0	19.0	18.0	1	0	1	0
80.0	0.8	0.8	1.0	1.0	138.0	...	15.8	1.0	1.0	21.0	16.0	22.0	1	0	0	1
88.0	1.5	1.5	1.0	1.0	100.0	...	14.7	1.0	1.0	19.0	26.0	18.0	1	0	1	0
86.0	1.0	1.0	1.0	1.0	120.0	...	12.5	1.0	0.6	16.0	14.0	22.0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
75.0	0.9	0.9	1.0	1.0	110.0	...	12.3	1.0	0.6	14.0	7.0	10.0	1	1	1	0
70.0	1.2	1.2	1.0	1.0	101.0	...	14.0	1.0	0.9	20.0	12.0	14.0	1	0	1	0
68.5	1.0	1.2	1.0	1.0	117.0	...	12.4	1.0	0.5	17.0	11.0	12.0	1	0	0	0
78.0	0.8	1.0	1.0	1.0	133.0	...	14.4	1.0	0.7	20.0	19.0	18.0	1	0	0	0
85.0	0.9	0.7	1.0	1.0	124.0	...	15.0	1.0	0.8	26.0	29.0	41.0	1	0	1	1

Code:

```
df.info()
```

Output:

```
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   gender               55692 non-null  int64  
 1   age                  55692 non-null  int64  
 2   height(cm)           55692 non-null  int64  
 3   weight(kg)           55692 non-null  int64  
 4   waist(cm)            55692 non-null  float64
 5   eyesight(left)       55692 non-null  float64
 6   eyesight(right)      55692 non-null  float64
 7   hearing(left)        55692 non-null  float64
 8   hearing(right)       55692 non-null  float64
 9   systolic             55692 non-null  float64
 10  relaxation           55692 non-null  float64
 11  fasting blood sugar  55692 non-null  float64
 12  Cholesterol          55692 non-null  float64
 13  triglyceride         55692 non-null  float64
 14  HDL                  55692 non-null  float64
 15  LDL                  55692 non-null  float64
 16  hemoglobin           55692 non-null  float64
 17  Urine protein        55692 non-null  float64
 18  serum creatinine     55692 non-null  float64
 19  AST                  55692 non-null  float64
 20  ALT                  55692 non-null  float64
 21  Gtp                  55692 non-null  float64
 22  oral                 55692 non-null  int64  
 23  dental caries        55692 non-null  int64  
 24  tartar               55692 non-null  int64  
 25  smoking              55692 non-null  int64  
```

Step 4: Standardise the variables

Code:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(df.drop('smoking',axis=1))
scaler_features=scaler.transform(df.drop('smoking', axis=1))
df_feat=pd.DataFrame(scaler_features,columns=df.columns[:-1])
df_feat.head()
```

Output:

	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	LDL	hemoglobin	U prc
0	-1.320858	-0.346517	-1.049465	-0.457476	-0.080484	0.384861	-0.015315	-0.162046	-0.163846	-0.547989	...	0.269644	-1.101061	-0.21
1	-1.320858	-0.346517	-0.505663	-0.457476	-0.112832	-0.436715	-0.838428	-0.162046	-0.163846	-0.182381	...	0.294079	-1.228898	-0.21
2	0.757084	0.896099	0.581943	-0.457476	-0.220659	-0.436715	-0.426872	-0.162046	-0.163846	1.206928	...	0.880501	0.752585	-0.21
3	0.757084	-0.346517	0.038140	0.322543	0.641955	1.001044	1.013576	-0.162046	-0.163846	-1.571690	...	2.713072	0.049478	-0.21
4	-1.320858	-0.346517	-1.049465	-0.457476	0.426302	-0.025927	-0.015315	-0.162046	-0.163846	-0.109259	...	-0.194607	-1.356736	-0.21

5 rows × 25 columns

Step 5: Train, test, split

Code:

```
X=df[['gender', 'age', 'height(cm)', 'weight(kg)', 'waist(cm)',
      'eyesight(left)', 'eyesight(right)', 'hearing(left)', 'hearing(right)',
      'systolic', 'relaxation', 'fasting blood sugar', 'Cholesterol',
      'triglyceride', 'HDL', 'LDL', 'hemoglobin', 'Urine protein',
      'serum creatinine', 'AST', 'ALT', 'Gtp', 'oral', 'dental caries',
      'tartar']]
y=df['smoking']
from sklearn.model_selection import train_test_split
X_train, X_test,y_train,y_test=train_test_split(scaler_features,df['smoking'],test_size=0.3)
```

Step 6: By using KNN

Code:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
```

Output:

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

Code:

```
pred=knn.predict(X_test)
```

Step 7: Predictions and Evaluations

Code:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,pred))
```

Output:

```
[[8659 1958]
 [1926 4165]]
```

Code:

```
print(classification_report(y_test,pred))
```

Output:

	precision	recall	f1-score	support
0	0.82	0.82	0.82	10617
1	0.68	0.68	0.68	6091
accuracy			0.77	16708
macro avg	0.75	0.75	0.75	16708
weighted avg	0.77	0.77	0.77	16708

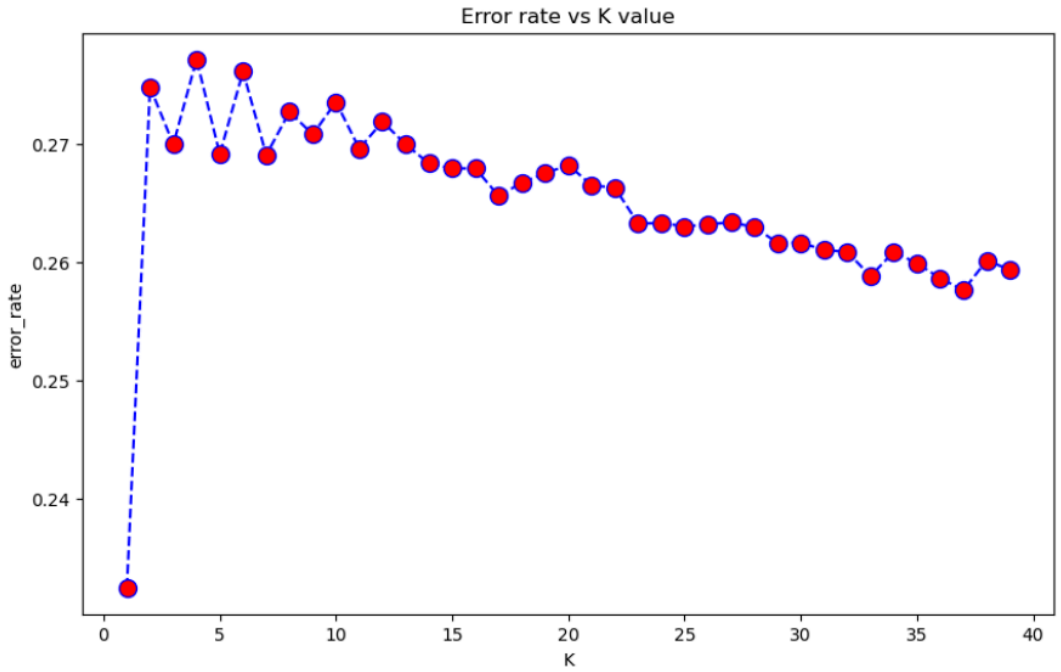
Step 8: Choosing a K value

Code:

```
error_rate=[]
for i in range (1,40):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i=knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue',linestyle='dashed',marker='o',markerfacecolor=
'red',markersize=10)
plt.title('Error rate vs K value')
plt.xlabel('K')
plt.ylabel('error_rate')
```

Output:

```
Text(0, 0.5, 'error_rate')
```



Code:

```
knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train,y_train)
pred=knn.predict(X_test)
print('With k=1')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

Output:

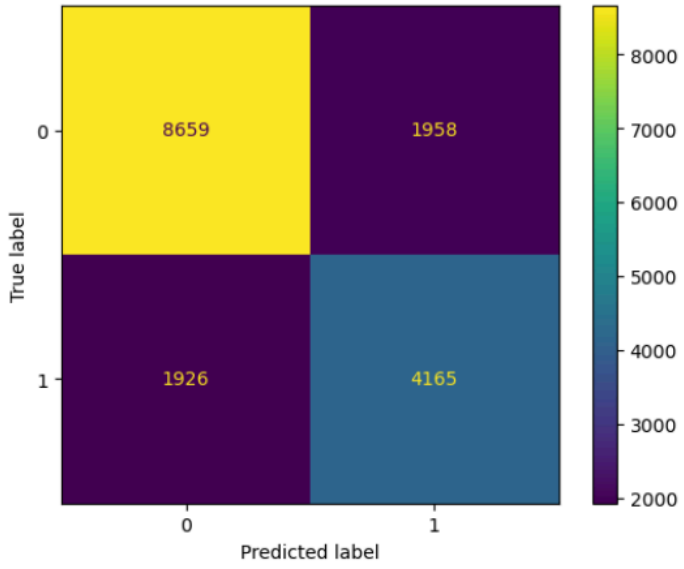
```
With k=1


[[8659 1958]
 [1926 4165]]

              precision    recall  f1-score   support

           0       0.82      0.82      0.82     10617
           1       0.68      0.68      0.68      6091

    accuracy                           0.77     16708
   macro avg       0.75      0.75      0.75     16708
weighted avg       0.77      0.77      0.77     16708
```

Code:

```
from sklearn.metrics import ConfusionMatrixDisplay
conf_matrix=confusion_matrix(y_test,pred)
vis=ConfusionMatrixDisplay(confusion_matrix=conf_matrix)
#sns.heatmap(conf_matrix,cmap='summer')
vis.plot()
```

Output:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x16126e1a140>
```



Step 9: Grid Search(for finding the best accuracy with the value of k)

Code:

```
from sklearn.model_selection import GridSearchCV
k_range = list(range(1, 31))
param_grid = dict(n_neighbors=k_range)
print(param_grid)
```

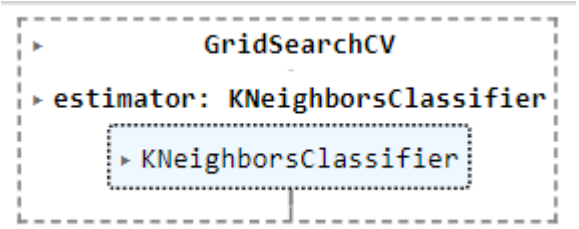
Output:

```
{'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 3
0]}
```

Code:

```
grid = GridSearchCV(knn, param_grid, cv=10, scoring='accuracy',
return_train_score=False)
grid.fit(X, y)
```

Output:



Code:

```
print("Best-Accuracy:",grid.best_score_)
print("when: ", grid.best_params_)
```

Output:

```
Best-Accuracy: 0.791228469473712
when  {'n_neighbors': 1}
```

Institute of Information Technology (IIT)

Jahangirnagar University

Lab Report: 07

Submitted by:

Name: Sanjida Akter

Roll No: 1982

Lab Date: 29-08-2023

Submission Date: 05-09-2023

Lab Report # Day 11

Given Task:

Use any dataset to classify using SVM.

Dataset Information:

This dataset is a collection of basic health biological signal data. The goal is to determine the presence or absence of smoking through bio-signals.

Columns:

data shape : (55692, 27)

- ID : index
- gender
- age : 5-years gap
- height(cm)
- weight(kg)
- waist(cm) : Waist circumference length
- eyesight(left)
- eyesight(right)
- hearing(left)
- hearing(right)
- systolic : Blood pressure
- relaxation : Blood pressure
- fasting blood sugar
- Cholesterol : total
- triglyceride
- HDL : cholesterol type
- LDL : cholesterol type
- hemoglobin
- Urine protein
- serum creatinine
- AST : glutamic oxaloacetic transaminase type
- ALT : glutamic oxaloacetic transaminase type
- Gtp : γ-GTP
- oral : Oral Examination status
- dental caries
- tartar : tartar status
- smoking

Step 1: Importing Required Libraries

Code:

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

Step 2: Data Preparation and Description

Code:

```
df=pd.read_csv("smoking.csv",index_col=0)
df
```

Output:

ID	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	hemoglobin	Urine protein	serum creatinine	A
0	F	40	155	60	81.3	1.2	1.0	1.0	1.0	114.0	...	12.9	1.0	0.7	1
1	F	40	160	60	81.0	0.8	0.6	1.0	1.0	119.0	...	12.7	1.0	0.6	2
2	M	55	170	60	80.0	0.8	0.8	1.0	1.0	138.0	...	15.8	1.0	1.0	2
3	M	40	165	70	88.0	1.5	1.5	1.0	1.0	100.0	...	14.7	1.0	1.0	1
4	F	40	155	60	86.0	1.0	1.0	1.0	1.0	120.0	...	12.5	1.0	0.6	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
55676	F	40	170	65	75.0	0.9	0.9	1.0	1.0	110.0	...	12.3	1.0	0.6	1
55681	F	45	160	50	70.0	1.2	1.2	1.0	1.0	101.0	...	14.0	1.0	0.9	2
55683	F	55	160	50	68.5	1.0	1.2	1.0	1.0	117.0	...	12.4	1.0	0.5	1
55684	M	60	165	60	78.0	0.8	1.0	1.0	1.0	133.0	...	14.4	1.0	0.7	2
55691	M	55	160	65	85.0	0.9	0.7	1.0	1.0	124.0	...	15.0	1.0	0.8	2

55692 rows × 26 columns

waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	hemoglobin	Urine protein	serum creatinine	AST	ALT	Gtp	oral	dental caries	tartar	smoking
81.3	1.2	1.0	1.0	1.0	114.0	...	12.9	1.0	0.7	18.0	19.0	27.0	Y	0	Y	0
81.0	0.8	0.6	1.0	1.0	119.0	...	12.7	1.0	0.6	22.0	19.0	18.0	Y	0	Y	0
80.0	0.8	0.8	1.0	1.0	138.0	...	15.8	1.0	1.0	21.0	16.0	22.0	Y	0	N	1
88.0	1.5	1.5	1.0	1.0	100.0	...	14.7	1.0	1.0	19.0	26.0	18.0	Y	0	Y	0
86.0	1.0	1.0	1.0	1.0	120.0	...	12.5	1.0	0.6	16.0	14.0	22.0	Y	0	N	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
75.0	0.9	0.9	1.0	1.0	110.0	...	12.3	1.0	0.6	14.0	7.0	10.0	Y	1	Y	0
70.0	1.2	1.2	1.0	1.0	101.0	...	14.0	1.0	0.9	20.0	12.0	14.0	Y	0	Y	0
68.5	1.0	1.2	1.0	1.0	117.0	...	12.4	1.0	0.5	17.0	11.0	12.0	Y	0	N	0
78.0	0.8	1.0	1.0	1.0	133.0	...	14.4	1.0	0.7	20.0	19.0	18.0	Y	0	N	0
85.0	0.9	0.7	1.0	1.0	124.0	...	15.0	1.0	0.8	26.0	29.0	41.0	Y	0	Y	1

Code:

```
df.shape
```

Output:

```
(55692, 26)
```

Code:

df.info()

Output:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 55692 entries, 0 to 55691
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   gender               55692 non-null  object 
 1   age                  55692 non-null  int64  
 2   height(cm)           55692 non-null  int64  
 3   weight(kg)           55692 non-null  int64  
 4   waist(cm)            55692 non-null  float64
 5   eyesight(left)       55692 non-null  float64
 6   eyesight(right)      55692 non-null  float64
 7   hearing(left)        55692 non-null  float64
 8   hearing(right)       55692 non-null  float64
 9   systolic             55692 non-null  float64
 10  relaxation           55692 non-null  float64
 11  fasting blood sugar  55692 non-null  float64
 12  Cholesterol          55692 non-null  float64
 13  triglyceride         55692 non-null  float64
 14  HDL                  55692 non-null  float64
 15  LDL                  55692 non-null  float64
 16  hemoglobin           55692 non-null  float64
 17  Urine protein        55692 non-null  float64
 18  serum creatinine     55692 non-null  float64
 19  AST                  55692 non-null  float64
 20  ALT                  55692 non-null  float64
 21  Gtp                  55692 non-null  float64
 22  oral                 55692 non-null  object 
 23  dental caries        55692 non-null  int64  
 24  tartar               55692 non-null  object 
 25  smoking              55692 non-null  int64  
dtypes: float64(18), int64(5), object(3)
memory usage: 11.5+ MB
```

Code:

df.columns

Output:

```
Index(['gender', 'age', 'height(cm)', 'weight(kg)', 'waist(cm)',
       'eyesight(left)', 'eyesight(right)', 'hearing(left)', 'hearing(right)',
       'systolic', 'relaxation', 'fasting blood sugar', 'Cholesterol',
       'triglyceride', 'HDL', 'LDL', 'hemoglobin', 'Urine protein',
       'serum creatinine', 'AST', 'ALT', 'Gtp', 'oral', 'dental caries',
       'tartar', 'smoking'],
      dtype='object')
```

Step 3: Converting String column to Integer

Code:

```
df['gender'].replace({'F':0, 'M':1}, inplace=True)
#df['gender'] = pd.factorize(df['gender'])[0].astype(float)
df['oral'].replace({'N':0, 'Y':1}, inplace=True)
df['tartar'].replace({'N':0, 'Y':1}, inplace=True)
df
```

Output:

ID	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	hemoglobin	Urine protein	serum creatinine
0	0	40	155	60	81.3	1.2	1.0	1.0	1.0	114.0	...	12.9	1.0	0.7
1	0	40	160	60	81.0	0.8	0.6	1.0	1.0	119.0	...	12.7	1.0	0.6
2	1	55	170	60	80.0	0.8	0.8	1.0	1.0	138.0	...	15.8	1.0	1.0
3	1	40	165	70	88.0	1.5	1.5	1.0	1.0	100.0	...	14.7	1.0	1.0
4	0	40	155	60	86.0	1.0	1.0	1.0	1.0	120.0	...	12.5	1.0	0.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
55676	0	40	170	65	75.0	0.9	0.9	1.0	1.0	110.0	...	12.3	1.0	0.6
55681	0	45	160	50	70.0	1.2	1.2	1.0	1.0	101.0	...	14.0	1.0	0.9
55683	0	55	160	50	68.5	1.0	1.2	1.0	1.0	117.0	...	12.4	1.0	0.5
55684	1	60	165	60	78.0	0.8	1.0	1.0	1.0	133.0	...	14.4	1.0	0.7
55691	1	55	160	65	85.0	0.9	0.7	1.0	1.0	124.0	...	15.0	1.0	0.8

55692 rows × 26 columns

waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	hemoglobin	Urine protein	serum creatinine	AST	ALT	Gtp	oral	dental caries	tartar	smoking
81.3	1.2	1.0	1.0	1.0	114.0	...	12.9	1.0	0.7	18.0	19.0	27.0	1	0	1	0
81.0	0.8	0.6	1.0	1.0	119.0	...	12.7	1.0	0.6	22.0	19.0	18.0	1	0	1	0
80.0	0.8	0.8	1.0	1.0	138.0	...	15.8	1.0	1.0	21.0	16.0	22.0	1	0	0	1
88.0	1.5	1.5	1.0	1.0	100.0	...	14.7	1.0	1.0	19.0	26.0	18.0	1	0	1	0
86.0	1.0	1.0	1.0	1.0	120.0	...	12.5	1.0	0.6	16.0	14.0	22.0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
75.0	0.9	0.9	1.0	1.0	110.0	...	12.3	1.0	0.6	14.0	7.0	10.0	1	1	1	0
70.0	1.2	1.2	1.0	1.0	101.0	...	14.0	1.0	0.9	20.0	12.0	14.0	1	0	1	0
68.5	1.0	1.2	1.0	1.0	117.0	...	12.4	1.0	0.5	17.0	11.0	12.0	1	0	0	0
78.0	0.8	1.0	1.0	1.0	133.0	...	14.4	1.0	0.7	20.0	19.0	18.0	1	0	0	0
85.0	0.9	0.7	1.0	1.0	124.0	...	15.0	1.0	0.8	26.0	29.0	41.0	1	0	1	1

Code:

```
df.info()
```

Output:

```
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   gender               55692 non-null  int64
 1   age                  55692 non-null  int64
 2   height(cm)           55692 non-null  int64
 3   weight(kg)           55692 non-null  int64
 4   waist(cm)            55692 non-null  float64
 5   eyesight(left)       55692 non-null  float64
 6   eyesight(right)      55692 non-null  float64
 7   hearing(left)        55692 non-null  float64
 8   hearing(right)       55692 non-null  float64
 9   systolic             55692 non-null  float64
 10  relaxation           55692 non-null  float64
 11  fasting blood sugar  55692 non-null  float64
 12  Cholesterol          55692 non-null  float64
 13  triglyceride         55692 non-null  float64
 14  HDL                  55692 non-null  float64
 15  LDL                  55692 non-null  float64
 16  hemoglobin           55692 non-null  float64
 17  Urine protein        55692 non-null  float64
 18  serum creatinine     55692 non-null  float64
 19  AST                  55692 non-null  float64
 20  ALT                  55692 non-null  float64
 21  Gtp                  55692 non-null  float64
 22  oral                 55692 non-null  int64
 23  dental caries        55692 non-null  int64
 24  tartar               55692 non-null  int64
 25  smoking              55692 non-null  int64
```

Code:

```
df.isnull().sum()
```

Output:

```
df.isnull().sum()

gender                 0
age                    0
height(cm)             0
weight(kg)             0
waist(cm)              0
eyesight(left)         0
eyesight(right)        0
hearing(left)          0
hearing(right)         0
systolic               0
relaxation             0
fasting blood sugar    0
Cholesterol            0
```

Code:

```
df.corr()
```

Output:

	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	hemoglobin	p
gender	1.000000	-0.290095	0.741556	0.574956	0.419568	0.127424	0.125680	-0.009407	-0.011579	0.167289	...	0.702214	0.0
age	-0.290095	1.000000	-0.479528	-0.324706	-0.026297	-0.195472	-0.192723	0.203993	0.208722	0.134023	...	-0.263078	0.0
height(cm)	0.741556	-0.479528	1.000000	0.675656	0.378902	0.151133	0.155665	-0.078663	-0.078323	0.080585	...	0.539367	0.0
weight(kg)	0.574956	-0.324706	0.675656	1.000000	0.822842	0.108433	0.113155	-0.050094	-0.052836	0.266131	...	0.492970	0.0
waist(cm)	0.419568	-0.026297	0.378902	0.822842	1.000000	0.027458	0.037996	0.023790	0.019286	0.316922	...	0.387066	0.0
eyesight(left)	0.127424	-0.195472	0.151133	0.108433	0.027458	1.000000	0.354574	-0.046571	-0.048788	-0.019330	...	0.095234	-0.0
eyesight(right)	0.125680	-0.192723	0.155665	0.113155	0.037996	0.354574	1.000000	-0.043877	-0.046623	-0.013720	...	0.096119	-0.0
hearing(left)	-0.009407	0.203993	-0.078663	-0.050094	0.023790	-0.046571	-0.043877	1.000000	0.510095	0.055219	...	-0.026116	0.0
hearing(right)	-0.011579	0.208722	-0.078323	-0.052836	0.019286	-0.048788	-0.046623	0.510095	1.000000	0.049095	...	-0.030943	0.0
systolic	0.167289	0.134023	0.080585	0.266131	0.316922	-0.019330	-0.013720	0.055219	0.049095	1.000000	...	0.186514	0.0
relaxation	0.177891	0.050745	0.113193	0.271634	0.292627	0.005199	0.011357	0.008754	0.001651	0.761051	...	0.232899	0.0
fasting blood sugar	0.098117	0.182351	0.019619	0.136237	0.211132	-0.041851	-0.044006	0.042314	0.041339	0.172700	...	0.099921	0.0
Cholesterol	-0.085270	0.055557	-0.082161	0.026403	0.065467	-0.004985	-0.007851	-0.023276	-0.023058	0.059572	...	0.061503	-0.0
triglyceride	0.241520	0.015102	0.156693	0.324429	0.361922	0.019717	0.019881	0.004750	0.000267	0.198826	...	0.273353	0.0
HDL	-0.306728	0.007047	-0.213284	-0.358868	-0.376203	-0.015296	-0.022220	-0.020159	-0.017986	-0.088487	...	-0.240095	-0.0
LDL	-0.042525	0.043007	-0.048419	0.040560	0.072817	-0.007257	-0.006172	-0.016706	-0.015426	0.016569	...	0.052903	-0.0
hemoglobin	0.702214	-0.263078	0.539367	0.492970	0.387066	0.095234	0.096119	-0.026116	-0.030943	0.186514	...	1.000000	0.0
Urine protein	0.015907	0.029625	0.005128	0.032566	0.045492	-0.002752	-0.013511	0.014527	0.019461	0.046170	...	0.021753	1.0
serum creatinine	0.507249	-0.106118	0.383883	0.324808	0.235024	0.071410	0.047608	0.003349	0.008060	0.072288	...	0.371382	0.0
AST	0.095718	0.032576	0.041737	0.120130	0.142690	-0.007966	-0.006921	0.018074	0.009393	0.083204	...	0.120575	0.0
ALT	0.167903	-0.063937	0.126511	0.250634	0.252478	0.019326	0.024182	0.004820	-0.005081	0.094893	...	0.202025	0.0
Gtp	0.237270	0.013031	0.139720	0.209625	0.243141	0.003850	0.012195	0.012864	0.009245	0.165724	...	0.223844	0.0
oral	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
dental caries	0.084408	-0.114984	0.079331	0.073536	0.044203	0.003684	0.016359	-0.016100	-0.016376	0.029894	...	0.067984	0.0
tartar	0.055473	-0.081796	0.055513	0.059921	0.046197	0.012532	0.006568	-0.035396	-0.026698	0.006542	...	0.053990	-0.0
smoking	0.510340	-0.162557	0.396675	0.302780	0.226259	0.061204	0.063017	-0.023209	-0.018855	0.073109	...	0.400678	0.0

Step 4: Standardise the variables

Code:

```
X=df.iloc[:, :-1]
X.head()
```

Output:

ID	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	...	LDL	hemoglobin	Urine protein	serum creatinine
0	0	40	155	60	81.3	1.2	1.0	1.0	1.0	114.0	...	126.0	12.9	1.0	0.7
1	0	40	160	60	81.0	0.8	0.6	1.0	1.0	119.0	...	127.0	12.7	1.0	0.6
2	1	55	170	60	80.0	0.8	0.8	1.0	1.0	138.0	...	151.0	15.8	1.0	1.0
3	1	40	165	70	88.0	1.5	1.5	1.0	1.0	100.0	...	226.0	14.7	1.0	1.0
4	0	40	155	60	86.0	1.0	1.0	1.0	1.0	120.0	...	107.0	12.5	1.0	0.6

5 rows × 25 columns

Code:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
scaler.fit(X)
X = scaler.transform(X)
y=df.iloc[:,-1]
```

Step 5: Train, test, split

Code:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

Step 6: By using SVM

Code: (default parameter)

```
from sklearn.svm import SVC
from sklearn import metrics
svc=SVC() #Default hyperparameters
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Accuracy Score:')
print(metrics.accuracy_score(y_test,y_pred))
```

Output:

```
Accuracy Score:
0.7666985875029926
```

Code: (default linear kernel)

```
svc=SVC(kernel='linear')
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Accuracy Score:')
print(metrics.accuracy_score(y_test,y_pred))
```

Output:

```
Accuracy Score:
0.7461096480727795
```

Code: (default RBF kernel)

```
svc=SVC(kernel='rbf')
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
```

```
print('Accuracy Score:')
print(metrics.accuracy_score(y_test,y_pred))
```

Output:

```
Accuracy Score:
0.7666985875029926
```

Code: (default polynomial kernel)

```
svc=SVC(kernel='poly')
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Accuracy Score:')
print(metrics.accuracy_score(y_test,y_pred))
```

Output:

```
Accuracy Score:
0.753531242518554
```

Step 7: Predictions and Evaluations

Code:

```
predictions = svc.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,predictions))
```

Output:

```
[[8265 2266]
 [1852 4325]]
```

Code:

```
print(classification_report(y_test,predictions))
```

Output:

```
              precision    recall  f1-score   support

           0       0.82      0.78      0.80     10531
           1       0.66      0.70      0.68      6177

    accuracy                           0.75     16708
   macro avg       0.74      0.74      0.74     16708
weighted avg       0.76      0.75      0.76     16708
```

Step 8: Gridsearch

Code:

```
param_grid = {'C': [0.1,1,5, 10, 50, 100, 1000], 'gamma': [10,1,0.1,0.01,0.001,0.0001],
'kernel': ['rbf']}
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=5)
grid.fit(X_train,y_train)
```

Output:

```
Fitting 5 folds for each of 42 candidates, totalling 210 fits
[CV 1/5] END .......C=0.1, gamma=10, kernel=rbf;, score=0.634 total time= 4.3min
[CV 2/5] END .......C=0.1, gamma=10, kernel=rbf;, score=0.634 total time= 4.1min
[CV 3/5] END .......C=0.1, gamma=10, kernel=rbf;, score=0.634 total time= 4.5min
[CV 4/5] END .......C=0.1, gamma=10, kernel=rbf;, score=0.634 total time= 4.4min
[CV 5/5] END .......C=0.1, gamma=10, kernel=rbf;, score=0.634 total time= 4.2min
```

Institute of Information Technology (IIT)

Jahangirnagar University

Lab Report: 08

Submitted by:

Name: Sanjida Akter

Roll No: 1982

Lab Date: 29-08-2023

Submission Date: 05-09-2023

Lab Report # Day 11

Given Task:

Use any dataset to use unsupervised algorithm : K means Clustering.

- Briefly explain the k-means clustering algorithm works.
- Describe the dataset you used for clustering. What are the features? What is the source of the data?
- How did you prepare the data for the k-means algorithm? Did you normalize or standardize the features?
- Explain how you determined the optimal number of clusters k. What values did you test?
- Provide the relevant code and parameters used for implementing k-means clustering on the dataset.
- Present the cluster results visually using charts, graphs, or other methods.
- Summarize the characteristics of each cluster. What common traits define each cluster?

K-means clustering algorithm:

The k-means clustering algorithm is an unsupervised learning method that partitions n observations into k clusters, where each observation belongs to the cluster with the nearest mean. The algorithm works by iteratively assigning each data point to its closest centroid and then recalculating the centroid of each cluster until the centroids no longer change or a maximum number of iterations is reached. The main objective is to minimize the sum of distances between the data points and their corresponding clusters.
The working of the K-Means algorithm is explained in the below steps:

- **Step-1:** Select the number K to decide the number of clusters.
- **Step-2:** Select random K points or centroids. (It can be different from the input dataset).
- **Step-3:** Assign each datapoint to their closest centroid, which will form the predefined K clusters.
- **Step-4:** Calculate the variance and place a new centroid of each cluster.
- **Step-5:** Repeat the third steps, which means re-assign each datapoint to the new closest centroid of each cluster.
- **Step-6:** If any reassignment occurs, then go to the step-4 else go to the FINISH.
- **Step-7:** The model is ready

Dataset:

The dataset is about credit card customers. It has information about customers that can be used to group them into different categories. The data has different features like the average credit limit, total credit cards, and total visits to the bank. The data comes from Kaggle and was updated 2 years ago.

Data preparation:

I will normalize the features before applying the k-means algorithm. This is because the features have different scales and ranges, which can affect the performance of the algorithm.

Determining the optimal number of clusters k:

There is no one definitive way to determine the optimal number of clusters k. One common approach is to use the elbow method. The elbow method plots the sum of squared errors (SSE) for different values of k. The optimal number of clusters is the point where the SSE curve starts to bend.

Code and parameters:

Code: (Importing required libraries)

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization
import seaborn as sns # for statistical data visualization
%matplotlib inline
```

Code: (Importing dataset)

```
dataset = pd.read_csv('Credit_Card_Customer_Data.csv')
dataset
```

Output:

:

	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
0	1	87073	100000	2	1	1	0
1	2	38414	50000	3	0	10	9
2	3	17341	50000	7	1	3	4
3	4	40496	30000	5	1	1	4
4	5	47437	100000	6	0	12	3
...	...	...	...	...	...	...	...
655	656	51108	99000	10	1	10	0
656	657	60732	84000	10	1	13	2
657	658	53834	145000	8	1	9	1
658	659	80655	172000	10	1	15	0
659	660	80150	167000	9	0	12	2

660 rows × 7 columns

Code: (summary of dataset)

```
dataset.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 660 entries, 0 to 659
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Sl_No                660 non-null    int64
 1   Customer Key         660 non-null    int64
 2   Avg_Credit_Limit     660 non-null    int64
 3   Total_Credit_Cards   660 non-null    int64
 4   Total_visits_bank    660 non-null    int64
 5   Total_visits_online  660 non-null    int64
 6   Total_calls_made     660 non-null    int64
dtypes: int64(7)
memory usage: 36.2 KB
```

Code: (summary of dataset)

```
dataset.info()
```

Output:

```
Index(['Sl_No', 'Customer Key', 'Avg_Credit_Limit', 'Total_Credit_Cards',
       'Total_visits_bank', 'Total_visits_online', 'Total_calls_made'],
      dtype='object')
```

Code: (data preparation)

```
X= dataset.iloc[:,2:].values
X
```

Output:

```
array([[100000,      2,      1,      1,      0],
       [ 50000,      3,      0,     10,      9],
       [ 50000,      7,      1,      3,      4],
       ...,
       [145000,      8,      1,      9,      1],
       [172000,     10,      1,     15,      0],
       [167000,      9,      0,     12,      2]], dtype=int64)
```

Code: (using standard scaler)

```
from sklearn.preprocessing import MinMaxScaler
sc_X = MinMaxScaler()
X  = sc_X.fit_transform(X)
X.shape
```

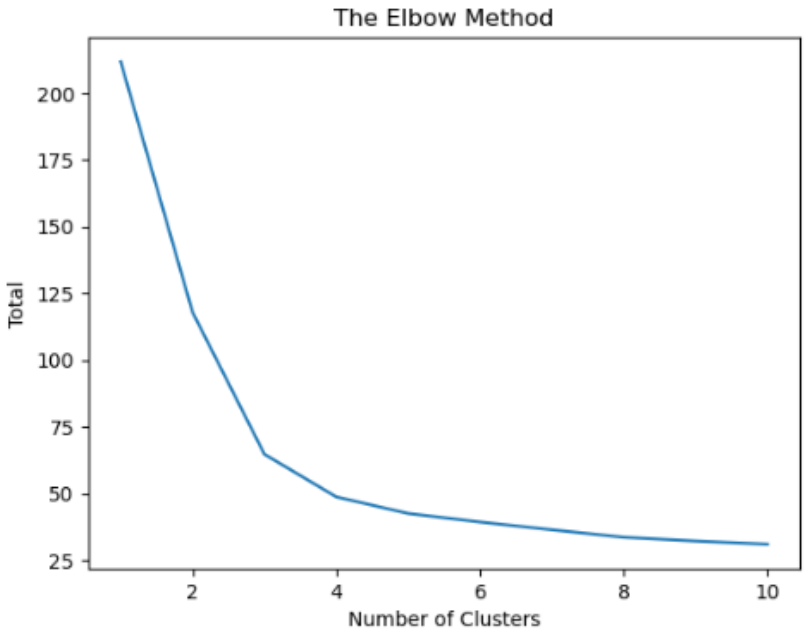
Output:

```
(660, 5)
```

Code: (Using the elbow method to find out optimal number of clusters)

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
    kmeans =
KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11),wcss)
plt.title("The Elbow Method")
plt.xlabel('Number of Clusters')
plt.ylabel('Total')
plt.show()
```

Output:



Code: (Applying k-means to the dataset)

```
kmeans =
KMeans(n_clusters=3,init='k-means++',max_iter=300,n_init=10,random_state=0)
y_kmeans = kmeans.fit_predict(X)
```

Output:

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=3.
  warnings.warn(
```

Code: (Visualising the clusters)

```
colors = ['red', 'blue', 'green']

for i in range(5):
    plt.scatter(X[y_kmeans == i, 0], X[y_kmeans == i, 1])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
        c='yellow', label='Centroids')
plt.title('Clusters of clients')

plt.show()
```

Output: