SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

VIDHYA NAGAR, KARUKUTTY, ERNAKULAM – 683582
(AN ISO 9001:2000 CERTIFIED INSTITUTION)

……………………………………………………
LABORATORY RECORD

Name:……………………………………………… Roll No:………………………………

Semester & Branch:…………………………………........... Year:………………………..

University Examination Reg No:………………….. of year…………Month………………

Certified that this is a bonafide record of work done by

Mr/Ms. ……….

In the. .... laboratory of SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY.

Place: Karukutty   Faculty in charge
Date: ……………

External Examiner   Internal Examiner

INDEX

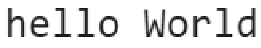
Experiment 1.A

Aim: To perform a review of python programming.

SOURCE CODE & OUTPUT:

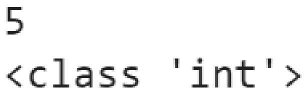
Basic Data Types

```
print("hello World")
```



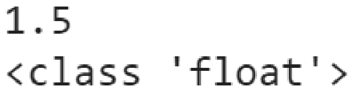
```
hello World
```

```
a=5
print(a)
print(type(a))
```



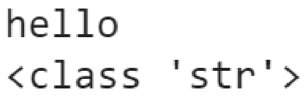
```
5
<class 'int'>
```

```
f=1.5
print(f)
print(type(f))
```



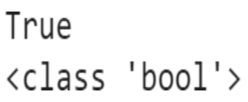
```
1.5
<class 'float'>
```

```
s="hello"
print(s)
print(type(s))
```



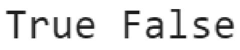
```
hello
<class 'str'>
```

```
b=True
print(b)
print(type(b))
```



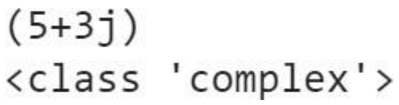
```
True
<class 'bool'>
```

```
t=True
f=False
print(t,f)
```



```
True False
```

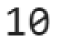
```
t=5+3j
print(t)
print(type(t))
```



```
(5+3j)
<class 'complex'>
```

Arithmetic Operators

```
a=7
b=3
sum=a+b
print(sum)
```

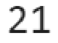


```
10
```

```
diff=a-b
print(diff)
```

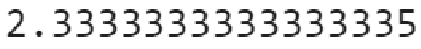
```
4
```

```
pro=a*b
print(pro)
```



```
21
```

```
quo=a/b
print(quo)
```



```
2.3333333333333335
```

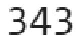
```
iquo=a//b
print(iquo)
```

```
2
```

```
rem=a % b
print(rem)
```

```
1
```

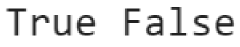
```
pow= a**b
print(pow)
```



```
343
```

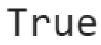
Boolean Operations

```
t=True
f=False
print(t,f)
```



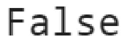
```
True False
```

```
p=5>3
print(p)
```



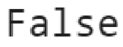
```
True
```

```
q=-1<-12.5
print(q)
```



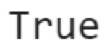
```
False
```

```
print(p and q)
```



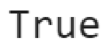
```
False
```

```
print(p or p)
```



```
True
```

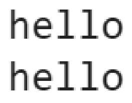
```
print(not q)
```



```
True
```

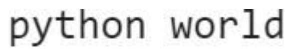
STRING OPERATIONS

```
s='hello'
u="hello"
print(s)
print(u)
```



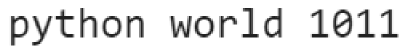
```
hello
hello
```

```
s1="python"
s2='world'
s3=s1+' '+s2
print(s3)
```



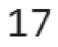
```
python world
```

```
s3='%s %s %d' %(s1,s2,1011)
print(s3)
```



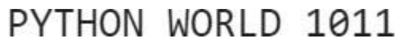
```
python world 1011
```

```
print(len(s3))
```



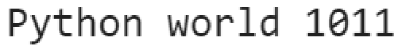
```
17
```

```
print(s3.upper())
```



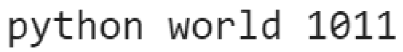
```
PYTHON WORLD 1011
```

```
print(s3.capitalize())
```



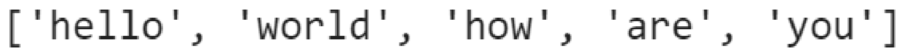
```
Python world 1011
```

```
print(s3.lower())
```



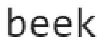
```
python world 1011
```

```
print('hello world how are you'.split(' '))
```



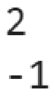
```
['hello', 'world', 'how', 'are', 'you']
```

```
print('book'.replace('o','e'))
```



```
beek
```

```
word='jewellery'
print(word.find('well'))
print(word.find('is'))
```

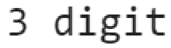


```
2
-1
```

Control Structures

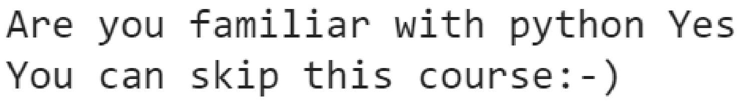
IF-ELSE

```
number= 123
if number>99 and number<1000 :
    print('3 digit')
else:
    print('Not 3 digit')
```



```
3 digit
```

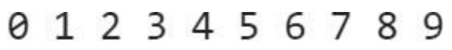
```
response=input('Are you familiar with python ')
if response.upper()=="YES":
  print("You can skip this course:-)")
elif response.upper() == "NO":
print("You are at the right place:-)")
else:
  print('Sorry wrong input :-(')
```



```
Are you familiar with python Yes
You can skip this course:-)
```

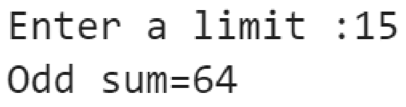
FOR LOOP

```
for x in range(10):
  print(x,end=' ')
```



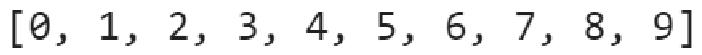
```
0 1 2 3 4 5 6 7 8 9
```

```
limit=int(input('Enter a limit :'))
sum=0
for i in range(1,limit+1):
  if i%2!=0:
    sum+=i
print("Odd sum="+str(sum))
```



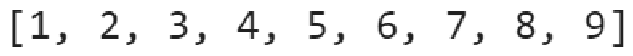
```
Enter a limit :15
Odd sum=64
```

```
print(list(range(10)))
```



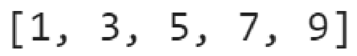
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
print(list(range(1,10)))
```



```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

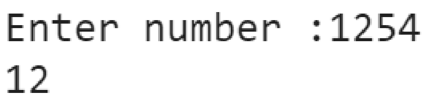
```
print(list(range(1,10,2)))
```



```
[1, 3, 5, 7, 9]
```

WHILE LOOP

```
number=int(input('Enter number :'))
s=0
while number>0:
s+=number%10
number=number//10
print(s)
```

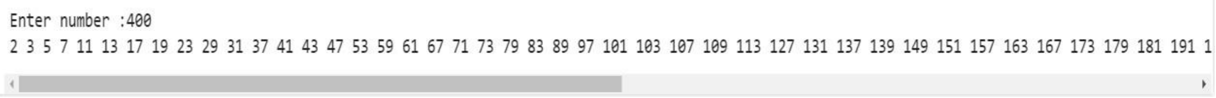


```
Enter number :1254
12
```

NESTED LOOP

```
limit=int(input('Enter number :'))
for num in range (2,limit+1):
is_divisible=False
```

```
 k=2
 while  k<=num//2:
   if num % k==0:
   is_divisible=True
   break;
   k+=1
 if not is_divisible:
   print(num,end=' ')
```

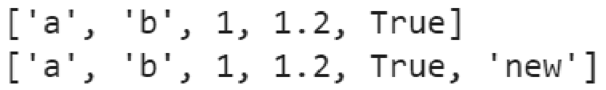


```
Enter number :400
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 1
```

Containers

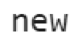
List

```
mylist=['a','b',1,1.2,True]
print(mylist)
mylist.append("new")
print(mylist)
```



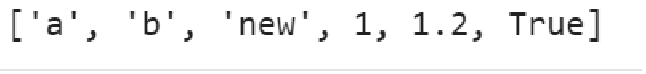
```
['a', 'b', 1, 1.2, True]
['a', 'b', 1, 1.2, True, 'new']
```

```
print(mylist.pop())
```



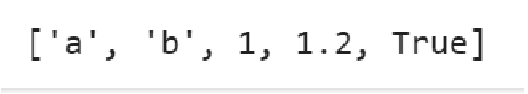
```
new
```

```
mylist.insert(2,'new')
print(mylist)
```



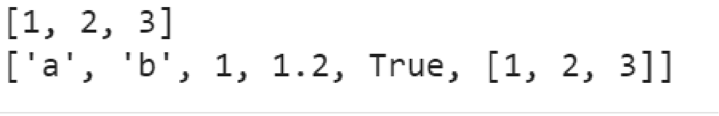
```
['a', 'b', 'new', 1, 1.2, True]
```

```
mylist.remove('new')
print(mylist)
```



```
['a', 'b', 1, 1.2, True]
```

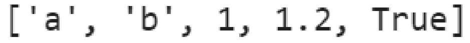
```
b=[1,2,3]
print(b)
mylist.append(b)
print(mylist)
```



```
[1, 2, 3]
['a', 'b', 1, 1.2, True, [1, 2, 3]]
```

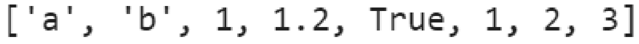
```
mylist.remove(b)
```

```
print(mylist)
```



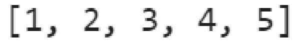
```
['a', 'b', 1, 1.2, True]
```

```
mylist.extend(b)
print(mylist)
```



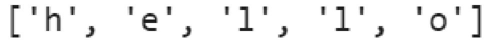
```
['a', 'b', 1, 1.2, True, 1, 2, 3]
```

```
a=[2,3,1,4,5]
a.sort()
print(a)
```



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

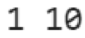
```
print(list('hello'))
```



```
['h', 'e', 'l', 'l', 'o']
```

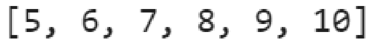
List Slicing

```
numbers=[0,1,2,3,4,5,6,7,8,9,10]
print(numbers[1],numbers[-1])
```



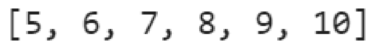
```
1 10
```

```
sliced=numbers[5:11]
print(sliced)
```



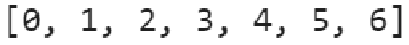
```
[5, 6, 7, 8, 9, 10]
```

```
slice1=numbers[5:]
print(slice1)
```



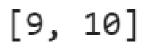
```
[5, 6, 7, 8, 9, 10]
```

```
Sliced=numbers[:7]
print(Sliced)
```



```
[0, 1, 2, 3, 4, 5, 6]
```

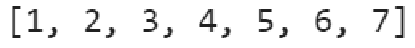
```
slice2=numbers[-2:]
print(slice2)
```



```
[9, 10]
```

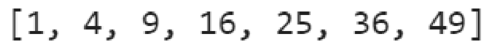
List Comprehension

```
numbers=list(range(1,8))
print(numbers)
```



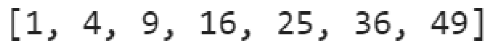
```
[1, 2, 3, 4, 5, 6, 7]
```

```
square=[]
for i in numbers:
   square.append(pow(i,2))
print(square)
```



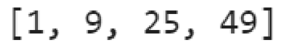
```
[1, 4, 9, 16, 25, 36, 49]
```

```
square=[x**2 for x in numbers]
print(square)
```



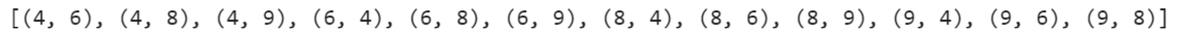
```
[1, 4, 9, 16, 25, 36, 49]
```

```
odd_square=[x**2 for x in numbers if x%2!=0]
print(odd_square)
```



```
[1, 9, 25, 49]
```

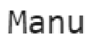
```
A=[4,6,8,9]
AxA=[(a,b) for a in A for b in A if a!=b ]
print(AxA)
```



```
[(4, 6), (4, 8), (4, 9), (6, 4), (6, 8), (6, 9), (8, 4), (8, 6), (8, 9), (9, 4), (9, 6), (9, 8)]
```

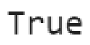
Dictionary

```
person={'name':'Manu','age':28}
print(person['name'])
```



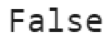
```
Manu
```

```
print('name' in person)
```



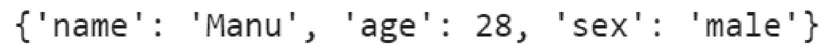
```
True
```

```
print('sex' in person)
```



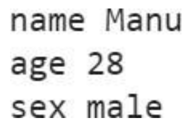
```
False
```

```
person['sex']='male'
print(person)
```



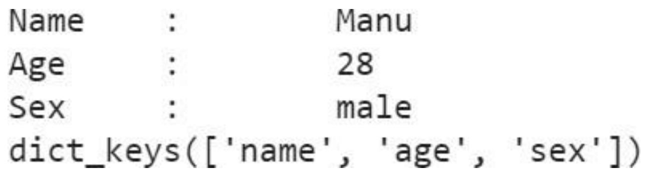
```
{'name': 'Manu', 'age': 28, 'sex': 'male'}
```

```
for item in person:
   print(item,person[item])
```



```
name Manu
age 28
sex male
```

```
for(key,value)in person.items():
   print(key.capitalize(),'\t:\t',value)
print(person.keys())
```



```
Name    :       Manu
Age     :       28
Sex     :       male
dict_keys(['name', 'age', 'sex'])
```

FUNCTIONS

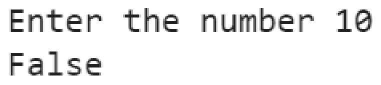
Finding the Square of the number

```
def square(number):
  return pow(number,2)
s=square(5)
print(s)
```

```
25
```

To check if a given number is prime

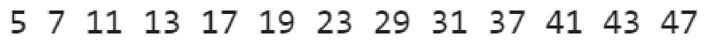
```
def isPrime(number):

    for factor in range(2, (number//2)+1):
        if number%factor == 0:

              return False

    return True

number = int(input('Enter the number '))
print(isPrime(number))
```



```
Enter the number 10
False
```

Prime in given range

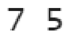
```
def printPrimes(llimit, ulimit):
  for num in range(llimit,ulimit+1):
    if isPrime(num)==True:
    print(num,end=' ')
printPrimes(5,50)
```



```
5 7 11 13 17 19 23 29 31 37 41 43 47
```

Swap 2 numbers

```
def swap(x,y):
  t=x
  x=y
  y=t
  return x,y
a=5
b=7
a,b=swap(a,b)
print(a,b)
```



```
7 5
```

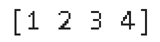
EXPERIMENT-1B

NUMPY

AIM: To perform a review of python and matrix operations using Numpy programming.

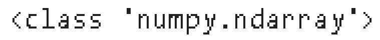
SOURCE CODE & OUTPUT:

```
import numpy as np
x = np.array([1,2,3,4])
print(x)
```



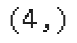
```
[1 2 3 4]
```

```
print(type(x))
```



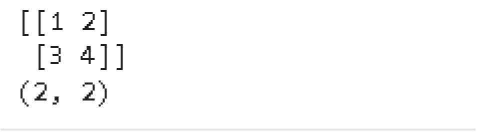
```
<class 'numpy.ndarray'>
```

```
print(x.shape)
```



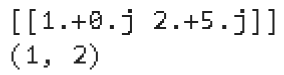
```
(4,)
```

```
y = np.array([[1,2],[3,4]])
print(y)
print(y.shape)
```



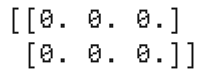
```
[[1 2]
 [3 4]]
(2, 2)
```

```
z = np.array([[1+0.j,2+5.j]])
print(z)
print(z.shape)
```



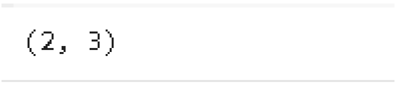
```
[[1.+0.j 2.+5.j]]
(1, 2)
```

```
a = np.zeros((2,3))
print(a)
```



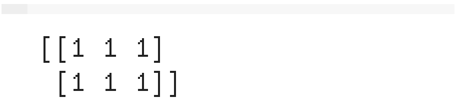
```
[[0. 0. 0.]
 [0. 0. 0.]]
```

```
print(a.shape)
```



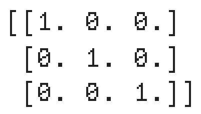
```
(2, 3)
```

```
b = np.ones((2,3), dtype=int)
print(b)
```



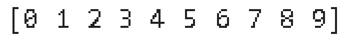
```
[[1 1 1]
 [1 1 1]]
```

```
d = np.eye(3)
print(d)
```



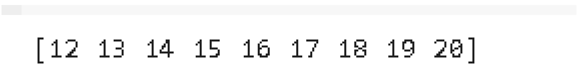
```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

```
e = np.arange(10)
print(e)
```



```
[0 1 2 3 4 5 6 7 8 9]
```

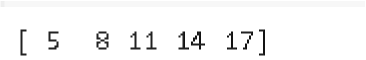
```
e = np.arange(12, 21)
print(e)
```



```
[12 13 14 15 16 17 18 19 20]
```

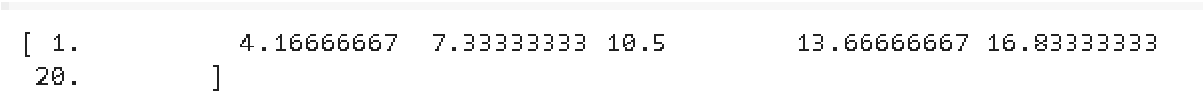
```
e = np.arange(5,20,3)
```

```
print(e)
```



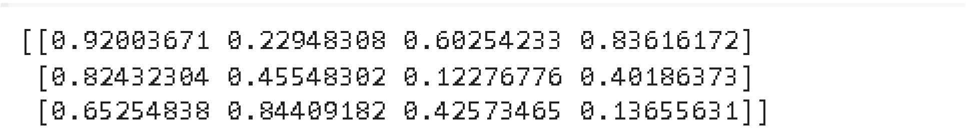
```
[ 5  8 11 14 17]
```

```
f = np.linspace(1,20,7)
print(f)
```



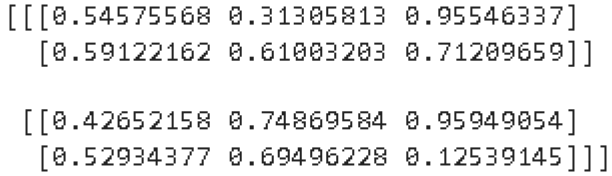
```
[ 1.          4.16666667  7.33333333 10.5        13.66666667 16.83333333
 20.        ]
```

```
g = np.random.random((3,4))
print(g)
```



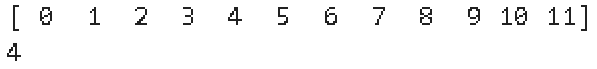
```
[[0.92003671 0.22948308 0.60254233 0.83616172]
 [0.82432304 0.45548302 0.12276776 0.40186373]
 [0.65254838 0.84409182 0.42573465 0.13655631]]
```

```
h = np.random.random((3,4))
print(h.reshape(2,2,3))
```



```
[[[0.54575568 0.31305813 0.95546337]
  [0.59122162 0.61003203 0.71209659]]

 [[0.42652158 0.74869584 0.95949054]
  [0.52934377 0.69496228 0.12539145]]]
```

```
x = np.arange(12)
print(x)
print(x[4])
```

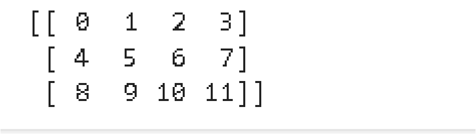


```
[ 0  1  2  3  4  5  6  7  8  9 10 11]
4
```

```
print(x[-1])
```

```
11
```

```
x.resize(3,4)
print(x)
```



```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

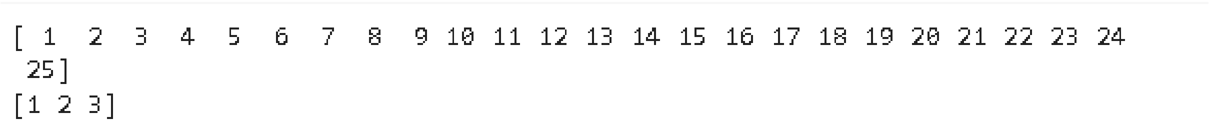
```
print(x[-1,-1])
```

```
11
```

```
print(x[2][3])
```

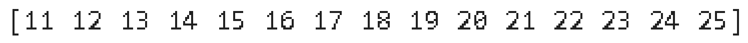
```
11
```

```
y = np.arange(1,26)
print(y)
print(y[:3])
```



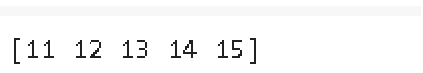
```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25]
[1 2 3]
```

```
print(y[10:])
```



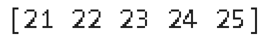
```
[11 12 13 14 15 16 17 18 19 20 21 22 23 24 25]
```

```
print(y[10:15])
```



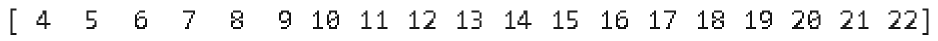
```
[11 12 13 14 15]
```

```
print(y[-5:])
```



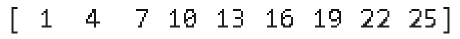
```
[21 22 23 24 25]
```

```
print(y[3:-3])
```



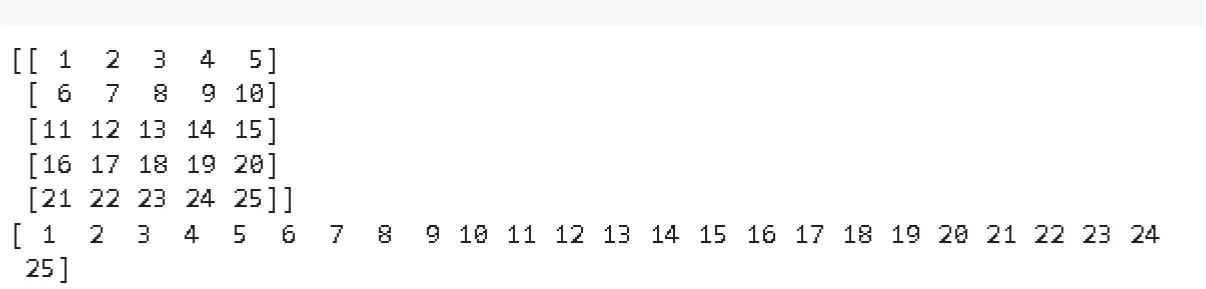
```
[ 4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22]
```

```
print(y[::3])
```



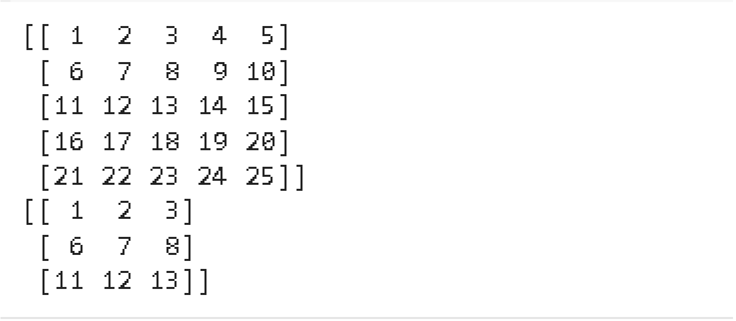
```
[ 1  4  7 10 13 16 19 22 25]
```

```
print(y.reshape((5,5)))
print(y)
```



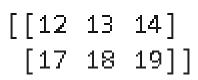
```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25]
```

```
y = y.reshape((5,5))
print(y)
print(y[:3,:3])
```



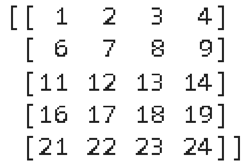
```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]
[[ 1  2  3]
 [ 6  7  8]
 [11 12 13]]
```

```
print(y[2:-1,1:-1])
```



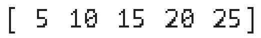
```
[[12 13 14]
 [17 18 19]]
```

print(y[:,:-1])



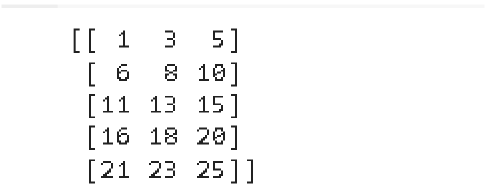
```
[[ 1  2  3  4]
 [ 6  7  8  9]
 [11 12 13 14]
 [16 17 18 19]
 [21 22 23 24]]
```

print(y[:,-1])



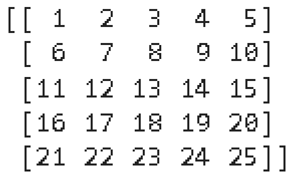
```
[ 5 10 15 20 25]
```

print(y[::,::2])



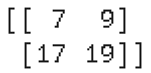
```
[[ 1  3  5]
 [ 6  8 10]
 [11 13 15]
 [16 18 20]
 [21 23 25]]
```

print(y)



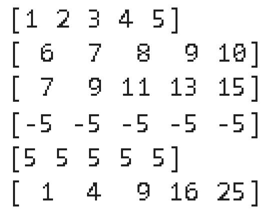
```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]
```

print(y[1::2,1::2])



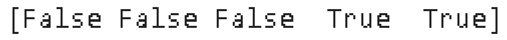
```
[[ 7  9]
 [17 19]]
```

```
a = np.arange(1,6)
b = np.arange(6,11)
print(a)
print(b)
print(a+b)
print(a-b)
print(b-a)
print(a**2)
```



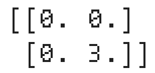
```
[1 2 3 4 5]
[ 6  7  8  9 10]
[ 7  9 11 13 15]
[-5 -5 -5 -5 -5]
[5 5 5 5 5]
[ 1  4  9 16 25]
```

```
print(a>3)
```



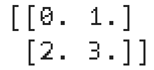
```
[False False False  True  True]
```

```
a = np.arange(0,4).reshape((2,2))
b = np.eye(2)
print(a*b)
```



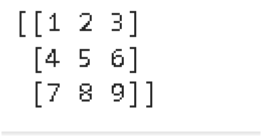
```
[[0. 0.]
 [0. 3.]]
```

```
print(np.dot(a,b))
```



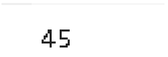
```
[[0. 1.]
 [2. 3.]]
```

```
x = np.arange(1,10).reshape(3,3)
print(x)
```



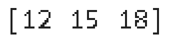
```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
print(x.sum())
```



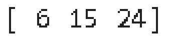
```
45
```

```
print(x.sum(axis=0))
```



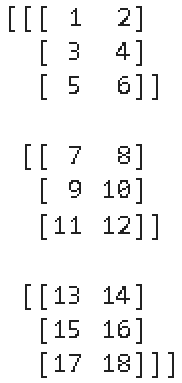
```
[12 15 18]
```

```
print(x.sum(axis=1))
```



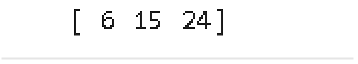
```
[ 6 15 24]
```

```
x = np.arange(1,19).reshape(3,3,2)
print(x)
```



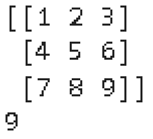
```
[[[ 1  2]
  [ 3  4]
  [ 5  6]]

 [[ 7  8]
  [ 9 10]
  [11 12]]

 [[13 14]
  [15 16]
  [17 18]]]
```

```
print(x.sum(axis=1))
```



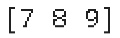
```
[ 6 15 24]
```

```
x = np.arange(1,10).reshape(3,3)
print(x)
print(x.max())
```



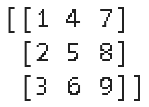
```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
9
```

```
print(x.max(axis=0))
```



```
[7 8 9]
```

```
print(x.transpose())
```



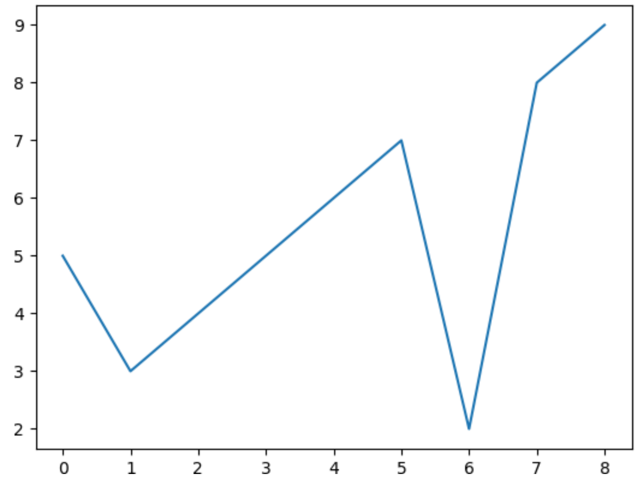
```
[[1 4 7]
 [2 5 8]
 [3 6 9]]
```

EXPERIMENT-1.C

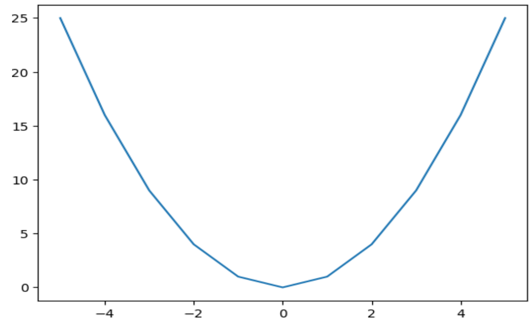
AIM: To perform Data visualisation using Matplotlib

SOURCE CODE & OUTPUT:

```
from matplotlib import pyplot as plt
y = [5,3,4,5,6,7,2,8,9]
plt.plot(y)
plt.show()
```



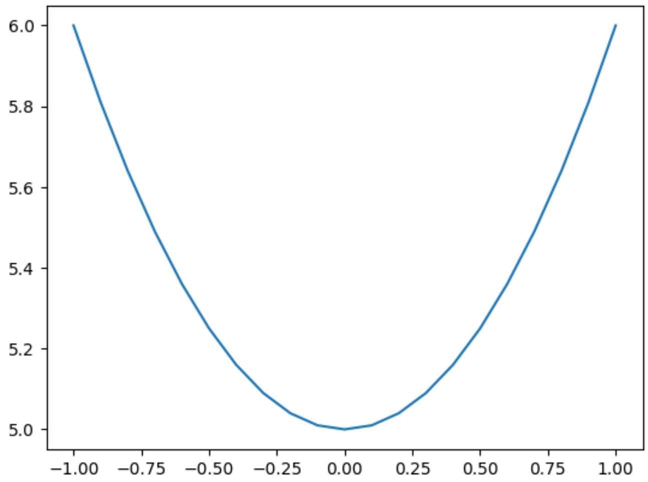
```
x = [-5,-4,-3,-2,-1,0,1,2,3,4,5]
y = [i**2 for i in x]
plt.plot(x,y)
plt.show()
```



```
import numpy as np
import math
x = np.arange(-1,1.1,0.1).tolist()
y = [i**2 + 5 for i in x]
```

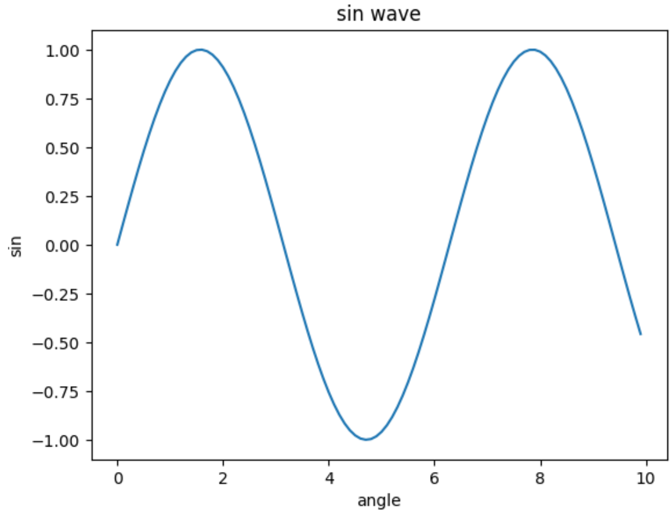
```
print(x)
print(y)
plt.plot(x,y)
plt.show()
```

```
[-1.0, -0.9, -0.8, -0.7000000000000001, -0.6000000000000001, -
0.5000000000000001, -0.40000000000000013, -0.30000000000000016, -
0.20000000000000018, -0.1000000000000002, -2.220446049250313e-16,
0.09999999999999964, 0.19999999999999973, 0.2999999999999998,
0.3999999999999997, 0.49999999999999956, 0.5999999999999996,
0.6999999999999997, 0.7999999999999996, 0.8999999999999995,
0.9999999999999996]
[6.0, 5.8100000000000005, 5.640000000000001, 5.49, 5.36, 5.25, 5.16,
5.09, 5.04, 5.01, 5.0, 5.01, 5.04, 5.09, 5.16, 5.25, 5.359999999999999,
5.489999999999999, 5.64, 5.809999999999999, 5.999999999999999]
```

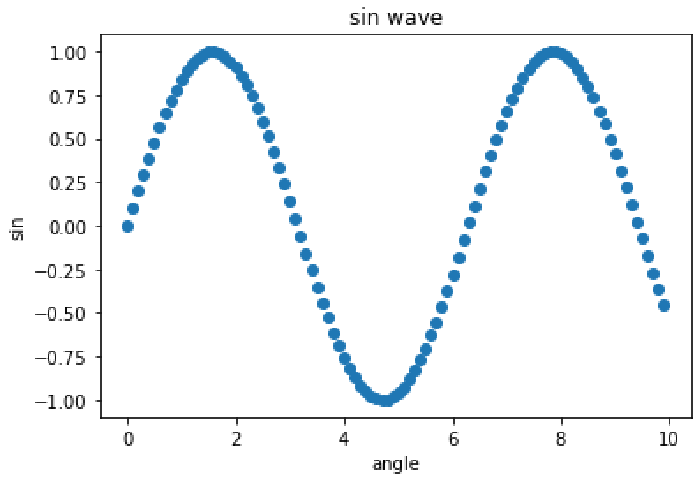


```
import numpy as np
x = np.arange(0,10,0.1)
y = np.sin(x)
print(x)
print(y)
plt.plot(x,y)
plt.xlabel('angle')
plt.ylabel('sin')
plt.title('sin wave')
plt.show()
```

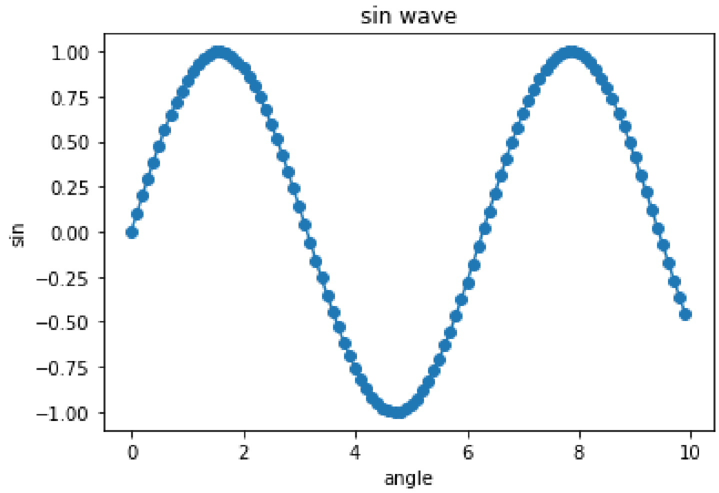
```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4 3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3
 5.4 5.5 5.6 5.7 5.8 5.9 6.  6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.  7.1
 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 8.  8.1 8.2 8.3 8.4 8.5 8.6 8.7 8.8 8.9
 9.  9.1 9.2 9.3 9.4 9.5 9.6 9.7 9.8 9.9]
[ 0.          0.09983342  0.19866933  0.29552021  0.38941834  0.47942554
  0.56464247  0.64421769  0.71735609  0.78332691  0.84147098  0.89120736
  0.93203909  0.96355819  0.98544973  0.99749499  0.9995736   0.99166481
  0.97384763  0.94630009  0.90929743  0.86320937  0.8084964   0.74570521
  0.67546318  0.59847214  0.51550137  0.42737988  0.33498815  0.23924933
  0.14112001  0.04158066 -0.05837414 -0.15774569 -0.2555411  -0.35078323
 -0.44252044 -0.52983614 -0.61185789 -0.68776616 -0.7568025  -0.81827711
 -0.87157577 -0.91616594 -0.95160207 -0.97753012 -0.993691   -0.99992326
 -0.99616461 -0.98245261 -0.95892427 -0.92581468 -0.88345466 -0.83226744
 -0.77276449 -0.70554033 -0.63126664 -0.55068554 -0.46460218 -0.37387666
 -0.2794155  -0.1821625  -0.0830894   0.0168139   0.1165492   0.21511999
  0.31154136  0.40484992  0.49411335  0.57843976  0.6569866   0.72896904
  0.79366786  0.85043662  0.8987081   0.93799998  0.96791967  0.98816823
  0.99854335  0.99894134  0.98935825  0.96988981  0.94073056  0.90217183
  0.85459891  0.79848711  0.7343971   0.66296923  0.58491719  0.50102086
  0.41211849  0.31909836  0.22288991  0.12445442  0.02477543 -0.07515112
 -0.17432678 -0.27176063 -0.36647913 -0.45753589]
```



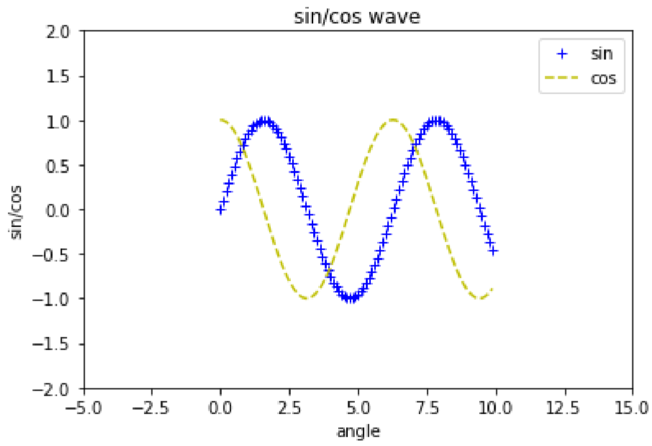
```
plt.scatter(x,y)
plt.xlabel('angle')
plt.ylabel('sin')
plt.title('sin wave')
plt.show()
```



```
plt.plot(x,y)
plt.scatter(x,y)
plt.xlabel('angle')
plt.ylabel('sin')
plt.title('sin wave')
plt.show()
```

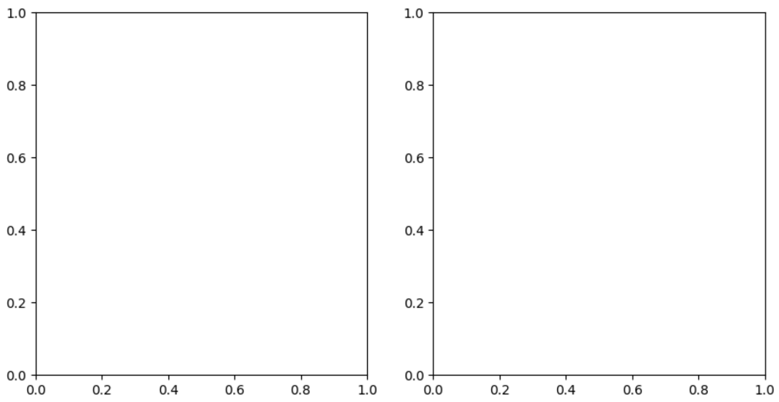


```
plt.plot(x,np.sin(x), 'b+', label='sin')
plt.plot(x,np.cos(x) ,'y--', label='cos')
plt.xlabel('angle')
plt.ylabel('sin/cos')
plt.title('sin/cos wave')
plt.ylim(-2,2)
plt.xlim(-5,15)
plt.legend()
plt.show()
```

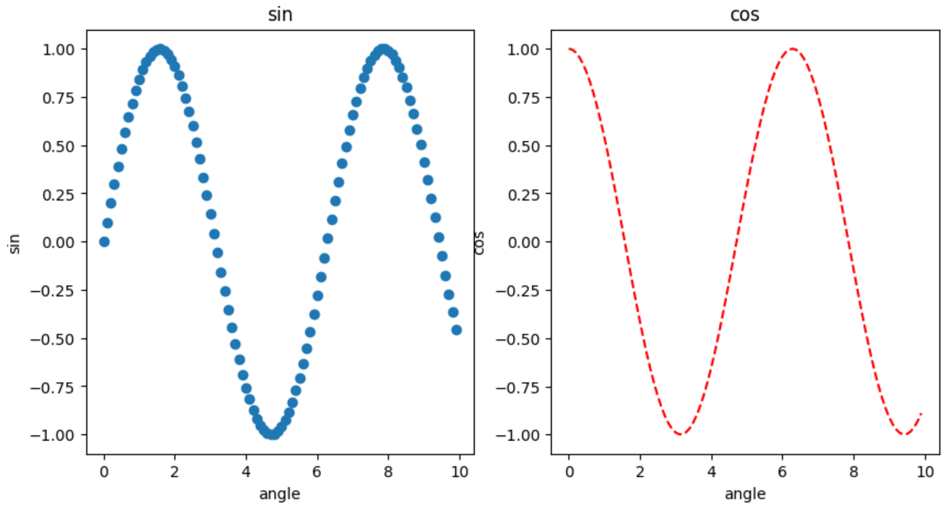


Subplot

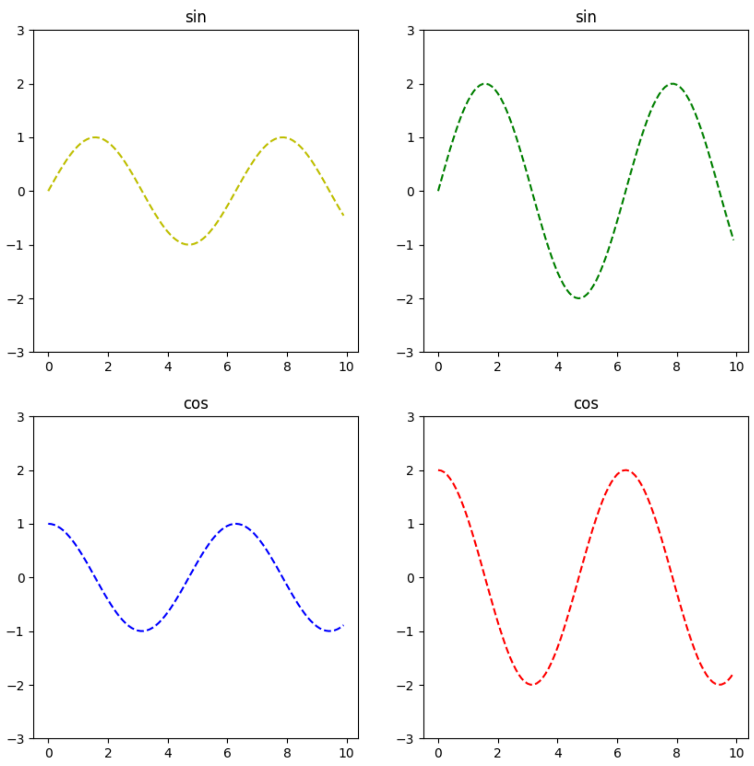
```
fig, axis = plt.subplots(1,2, figsize=(10,5))
print(axis.shape)
```



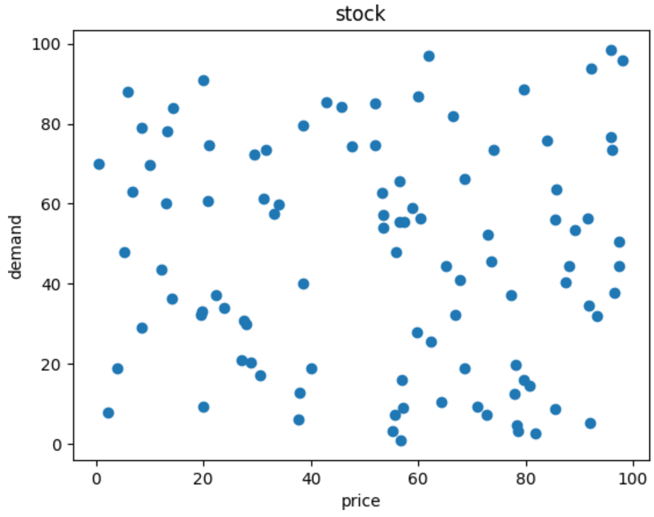
```
fig, axis = plt.subplots(1,2, figsize=(10,5))
x = np.arange(0,10,0.1)
axis[0].plot(x,np.sin(x), 'g--')
axis[0].set_title('sin')
axis[0].set_xlabel('angle')
axis[0].set_ylabel('sin')
axis[1].plot(x,np.cos(x), 'r--')
axis[1].set_title('cos')
axis[1].set_xlabel('angle')
axis[1].set_ylabel('cos')
plt.show()
```



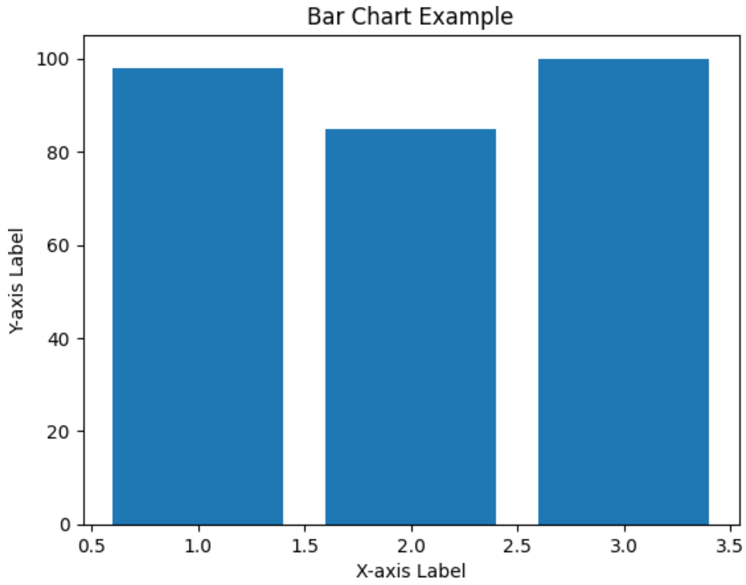
```
fig, axis = plt.subplots(2,2, figsize=(10,10))
x = np.arange(0,10,0.1)
axis[0][0].plot(x,np.sin(x), 'y--')
axis[0][0].set_title('sin')
axis[0][0].set_ylim(-3,3)
axis[0][1].plot(x,2*np.sin(x), 'g--')
axis[0][1].set_title('sin')
axis[0][1].set_ylim(-3,3)
axis[1][0].plot(x,np.cos(x), 'b--')
axis[1][0].set_title('cos')
axis[1][0].set_ylim(-3,3)
axis[1][1].plot(x,2*np.cos(x), 'r--')
axis[1][1].set_title('cos')
axis[1][1].set_ylim(-3,3)
plt.show()
```



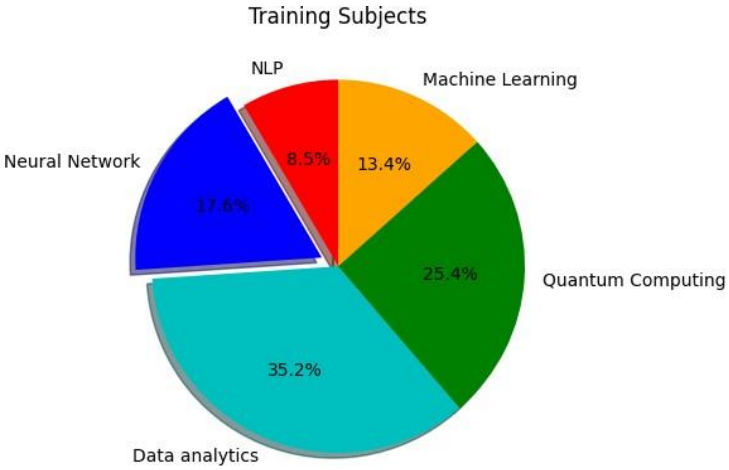
```
x = np.random.random(100)*100
y = np.random.random(100)*100
plt.scatter(x,y)
plt.xlabel('price')
plt.ylabel('demand')
plt.title('stock')
plt.show()
```



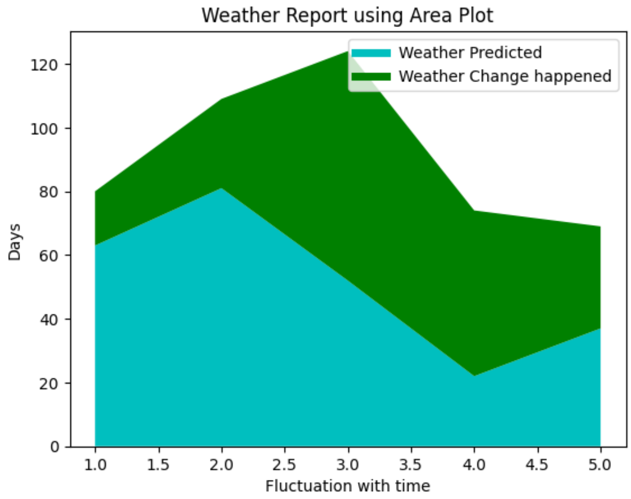
```
x = np.array([1,2,3])
y = [98,85,100]
plt.bar(x,y)
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.title('Bar Chart Example')
plt.show()
```



```
slice = [12, 25, 50, 36, 19]
activities = ['NLP', 'Neural Network', 'Data analytics', 'Quantum
Computing', 'Machine Learning']
cols = ['r', 'b', 'c', 'g', 'orange']
plt.pie(slice,
        labels=activities,
        colors=cols,
        startangle=90,
        shadow=True,
        explode=(0, 0.1, 0, 0, 0),
        autopct='%1.1f%%
        )

plt.title('Training Subjects')
plt.show()
```

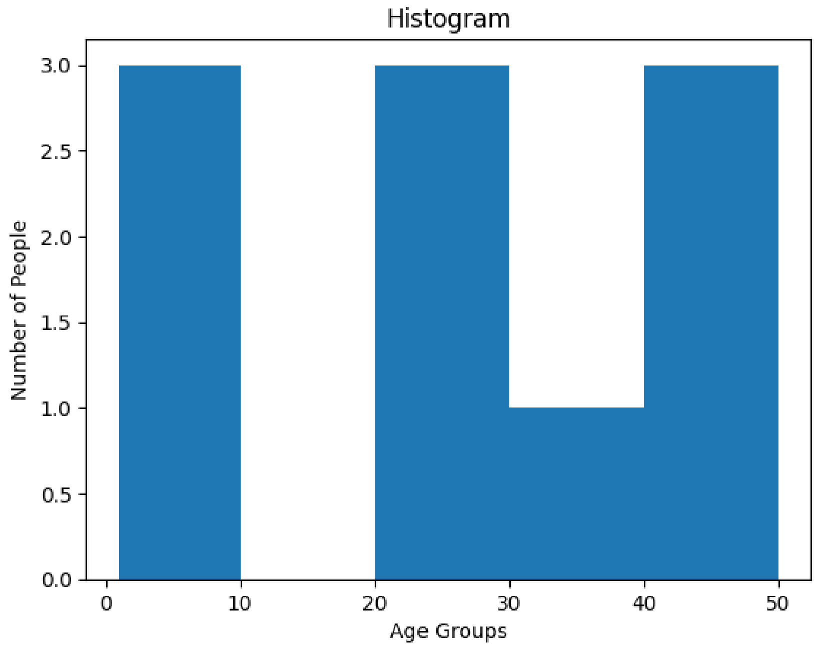


```
days = [1, 2, 3, 4, 5]
age = [63, 81, 52, 22, 37]
weight = [17, 28, 72, 52, 32]
plt.plot([], [], color='c', label='Weather Predicted', linewidth=5)
plt.plot([], [], color='g', label='Weather Change happened', linewidth=5)
plt.stackplot(days, age, weight, colors=['c', 'g'])
plt.xlabel('Fluctuation with time')
plt.ylabel('Days')
plt.title('Weather Report using Area Plot')
plt.legend()
plt.show()
```



```
pop = [22, 55, 62, 45, 21, 22, 34, 42, 42, 4, 2, 8]
```

```
bins = [1, 10, 20, 30, 40, 50]
plt.hist(pop, bins,
rwidth=1) plt.xlabel('Age
Groups')
plt.ylabel('Number of
People')
```



```
plt.title('Histogram')
plt.show()
```

RESULT

Program executed successfully.

EXPERIMENT-1.D

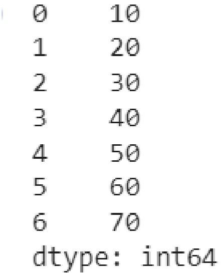
AIM :Familarization of Pandas Pandas is a popular open-source data manipulation and analysis library for Python. It provides easy-to-use data structures and data analysis tools for working with structured data, such as tabular data, time series, and more. Pandas is built on top of the NumPy library and is widely used in data science, data analysis, and machine learning tasks.

SOURCE CODE & OUTPUT:

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

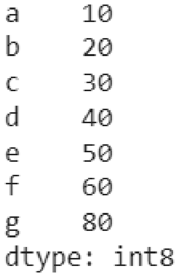
Pandas Series

```
data = pd.Series([10, 20, 30, 40, 50, 60, 70])
data
```



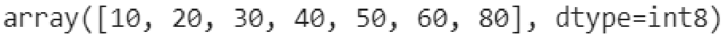
```
0    10
1    20
2    30
3    40
4    50
5    60
6    70
dtype: int64
```

```
data = pd.Series([10, 20, 30, 40, 50, 60, 80], index = ['a','b','c','d','e','f','g'], dtype
= 'int8')
data
```



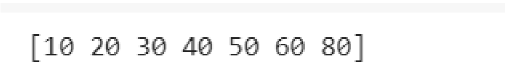
```
a    10
b    20
c    30
d    40
e    50
f    60
g    80
dtype: int8
```

```
data.values
```



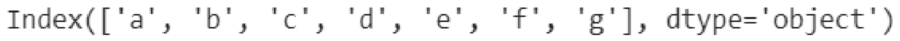
```
array([10, 20, 30, 40, 50, 60, 80], dtype=int8)
```

```
array_data = data.values

print(array_data)
```



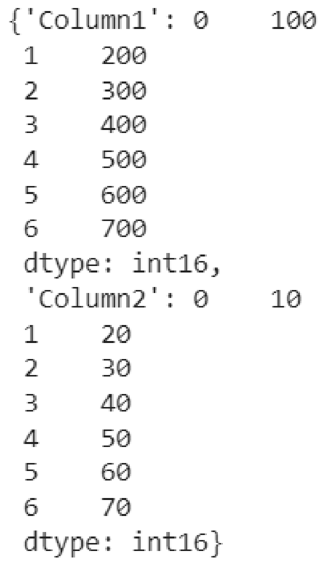
```
[10 20 30 40 50 60 80]
```

data.index



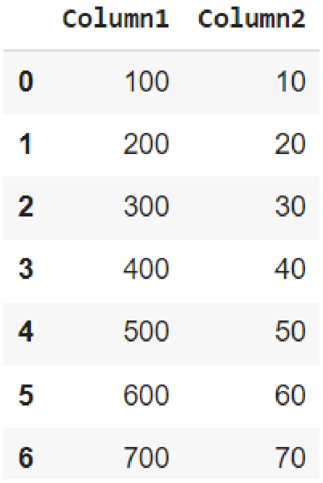
```
Index(['a', 'b', 'c', 'd', 'e', 'f', 'g'], dtype='object')
```

```
data_series = {
            'Column1' :  pd.Series([100, 200, 300, 400, 500, 600, 700], dtype = 'int16'),
            'Column2' :  pd.Series([10, 20, 30, 40, 50, 60, 70], dtype = 'int16')
           }
data_series
```



```
{'Column1': 0    100
 1    200
 2    300
 3    400
 4    500
 5    600
 6    700
 dtype: int16,
 'Column2': 0    10
 1    20
 2    30
 3    40
 4    50
 5    60
 6    70
 dtype: int16}
```

pd.DataFrame(data_series)

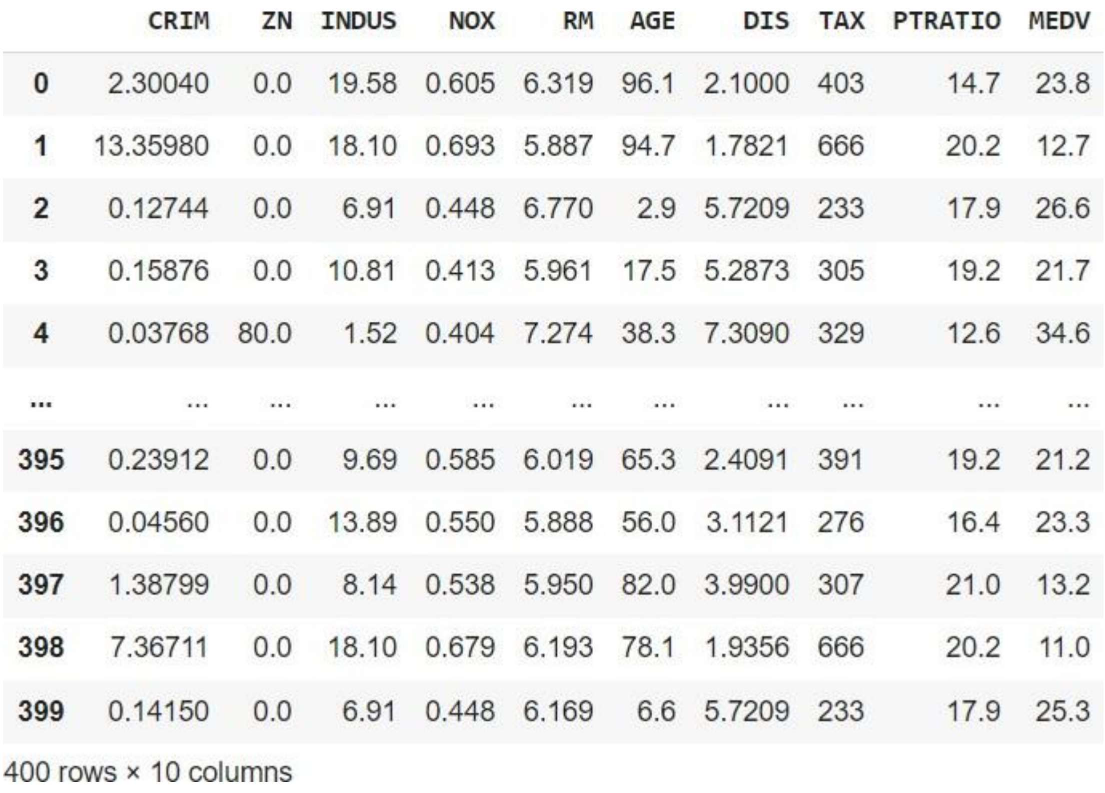


	Column1	Column2
0	100	10
1	200	20
2	300	30
3	400	40
4	500	50
5	600	60
6	700	70

DataFrame

```
movies_df = pd.read_csv('https://raw.githubusercontent.com/ammishra08/MachineLearning/master/Datasets/boston_train.csv', sep = ',')
```

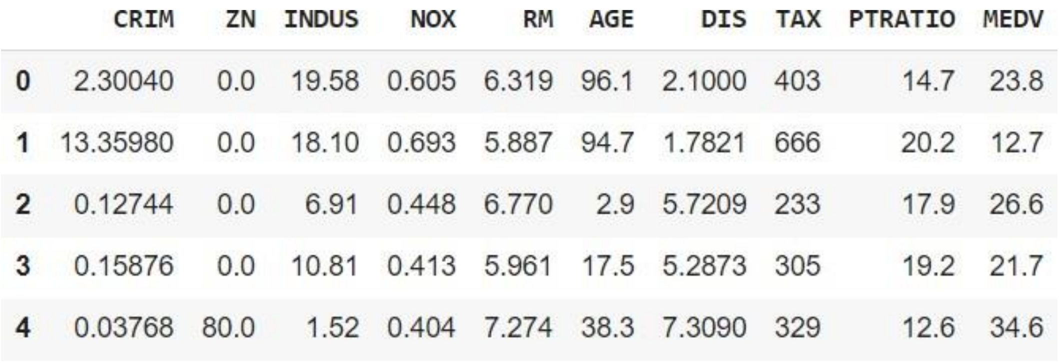
movies_df



	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	TAX	PTRATIO	MEDV
0	2.30040	0.0	19.58	0.605	6.319	96.1	2.1000	403	14.7	23.8
1	13.35980	0.0	18.10	0.693	5.887	94.7	1.7821	666	20.2	12.7
2	0.12744	0.0	6.91	0.448	6.770	2.9	5.7209	233	17.9	26.6
3	0.15876	0.0	10.81	0.413	5.961	17.5	5.2873	305	19.2	21.7
4	0.03768	80.0	1.52	0.404	7.274	38.3	7.3090	329	12.6	34.6
...	...	...	...	...	...	...	...	...	...	...
395	0.23912	0.0	9.69	0.585	6.019	65.3	2.4091	391	19.2	21.2
396	0.04560	0.0	13.89	0.550	5.888	56.0	3.1121	276	16.4	23.3
397	1.38799	0.0	8.14	0.538	5.950	82.0	3.9900	307	21.0	13.2
398	7.36711	0.0	18.10	0.679	6.193	78.1	1.9356	666	20.2	11.0
399	0.14150	0.0	6.91	0.448	6.169	6.6	5.7209	233	17.9	25.3

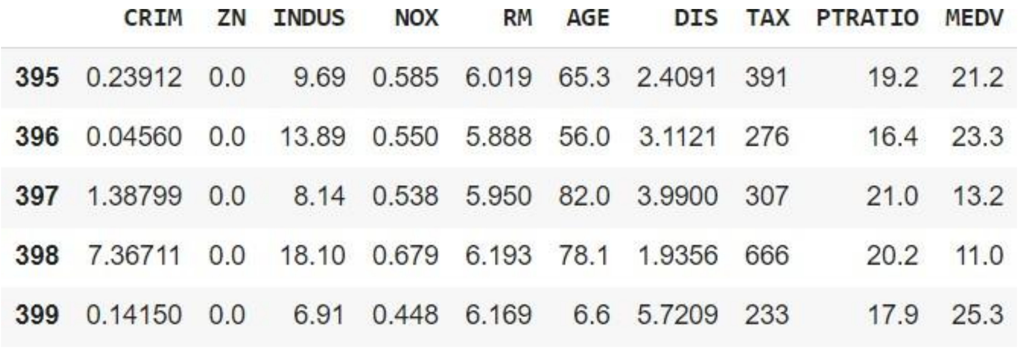
400 rows × 10 columns

movies_df.head()



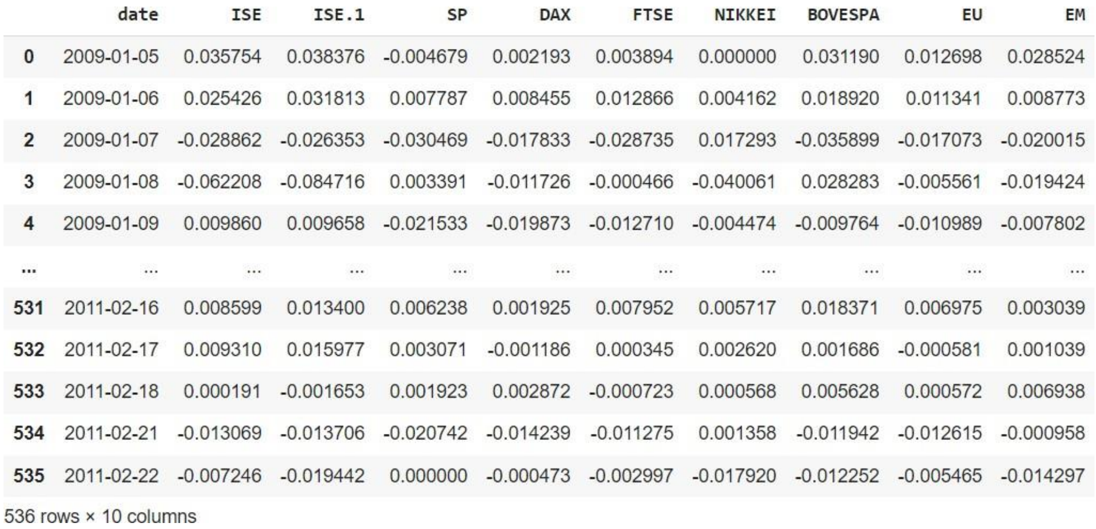
	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	TAX	PTRATIO	MEDV
0	2.30040	0.0	19.58	0.605	6.319	96.1	2.1000	403	14.7	23.8
1	13.35980	0.0	18.10	0.693	5.887	94.7	1.7821	666	20.2	12.7
2	0.12744	0.0	6.91	0.448	6.770	2.9	5.7209	233	17.9	26.6
3	0.15876	0.0	10.81	0.413	5.961	17.5	5.2873	305	19.2	21.7
4	0.03768	80.0	1.52	0.404	7.274	38.3	7.3090	329	12.6	34.6

movies_df.tail()



	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	TAX	PTRATIO	MEDV
395	0.23912	0.0	9.69	0.585	6.019	65.3	2.4091	391	19.2	21.2
396	0.04560	0.0	13.89	0.550	5.888	56.0	3.1121	276	16.4	23.3
397	1.38799	0.0	8.14	0.538	5.950	82.0	3.9900	307	21.0	13.2
398	7.36711	0.0	18.10	0.679	6.193	78.1	1.9356	666	20.2	11.0
399	0.14150	0.0	6.91	0.448	6.169	6.6	5.7209	233	17.9	25.3

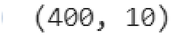
```
stock_data = pd.read_excel("https://github.com/ammishra08/MachineLearning/raw/master/Datasets/data_akbilgic.xlsx", header=1)
stock_data
```



	date	ISE	ISE.1	SP	DAX	FTSE	NIKKEI	BOVESPA	EU	EM
0	2009-01-05	0.035754	0.038376	-0.004679	0.002193	0.003894	0.000000	0.031190	0.012698	0.028524
1	2009-01-06	0.025426	0.031813	0.007787	0.008455	0.012866	0.004162	0.018920	0.011341	0.008773
2	2009-01-07	-0.028862	-0.026353	-0.030469	-0.017833	-0.028735	0.017293	-0.035899	-0.017073	-0.020015
3	2009-01-08	-0.062208	-0.084716	0.003391	-0.011726	-0.000466	-0.040061	0.028283	-0.005561	-0.019424
4	2009-01-09	0.009860	0.009658	-0.021533	-0.019873	-0.012710	-0.004474	-0.009764	-0.010989	-0.007802
...	...	...	...	...	...	...	...	...	...	...
531	2011-02-16	0.008599	0.013400	0.006238	0.001925	0.007952	0.005717	0.018371	0.006975	0.003039
532	2011-02-17	0.009310	0.015977	0.003071	-0.001186	0.000345	0.002620	0.001686	-0.000581	0.001039
533	2011-02-18	0.000191	-0.001653	0.001923	0.002872	-0.000723	0.000568	0.005628	0.000572	0.006938
534	2011-02-21	-0.013069	-0.013706	-0.020742	-0.014239	-0.011275	0.001358	-0.011942	-0.012615	-0.000958
535	2011-02-22	-0.007246	-0.019442	0.000000	-0.000473	-0.002997	-0.017920	-0.012252	-0.005465	-0.014297

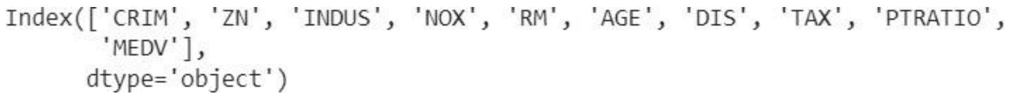
536 rows × 10 columns

```
movies_df.shape
```



```
(400, 10)
```

```
movies_df.columns
```

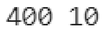


```
Index(['CRIM', 'ZN', 'INDUS', 'NOX', 'RM', 'AGE', 'DIS', 'TAX', 'PTRATIO',
       'MEDV'],
      dtype='object')
```

```
len(movies_df.columns)
```

```
10
```

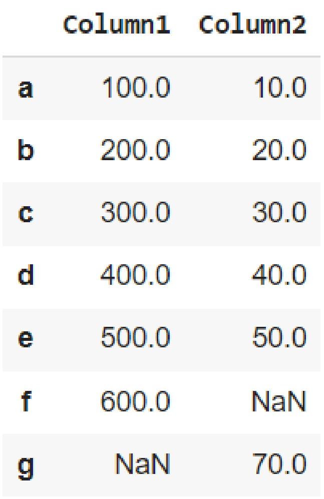
```
print(movies_df.shape[0], movies_df.shape[1])
```



```
400 10
```

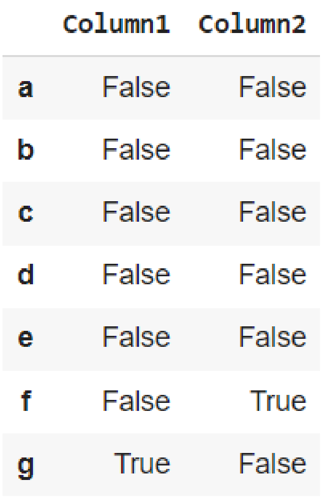
Data Manipulation

```
data_series = {
        'Column1' :  pd.Series([100, 200, 300, 400, 500, 600], index = ['a','b','c','d','e','f'], dtype = 'int16'),
        'Column2' :  pd.Series([10, 20, 30, 40, 50, 70], index = ['a','b','c','d','e','g'], dtype = 'int16')
       }
df = pd.DataFrame(data_series)
df
```



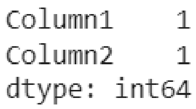
	Column1	Column2
a	100.0	10.0
b	200.0	20.0
c	300.0	30.0
d	400.0	40.0
e	500.0	50.0
f	600.0	NaN
g	NaN	70.0

df.isnull()



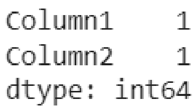
	Column1	Column2
a	False	False
b	False	False
c	False	False
d	False	False
e	False	False
f	False	True
g	True	False

df.isnull().sum()



```
Column1    1
Column2    1
dtype: int64
```

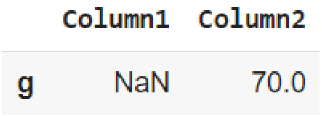
df.isna().sum()



```
Column1    1
Column2    1
dtype: int64
```

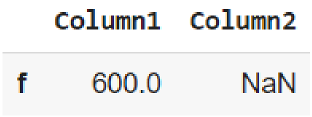
df.notnull()

df[df['Column1'].isnull() == True]



	Column1	Column2
g	NaN	70.0

df[df['Column2'].isnull() == True]



	Column1	Column2
f	600.0	NaN

RESULT:

Program Executed Successfully.

Experiment-2
KNN CLASSIFICATION

AIM: To implement decision tree using any standard dataset available in the public domain and find the accuracy of the algorithm.

KNN WITH DIABETES DATASET

ALGORITHM:

Step-1: Load the dataset- diabetes.csv

Step 2: Pre-process the dataset by replacing zeros suitable mean values.

Step 3: Perform the training and testing dataset splitting

Step 4: Determine the number of neighbors for the training dataset.

Step-5: Calculate the Euclidean distance of K number of neighbors

Step-6: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-7: Among these k neighbors, count the number of the data points in each category.

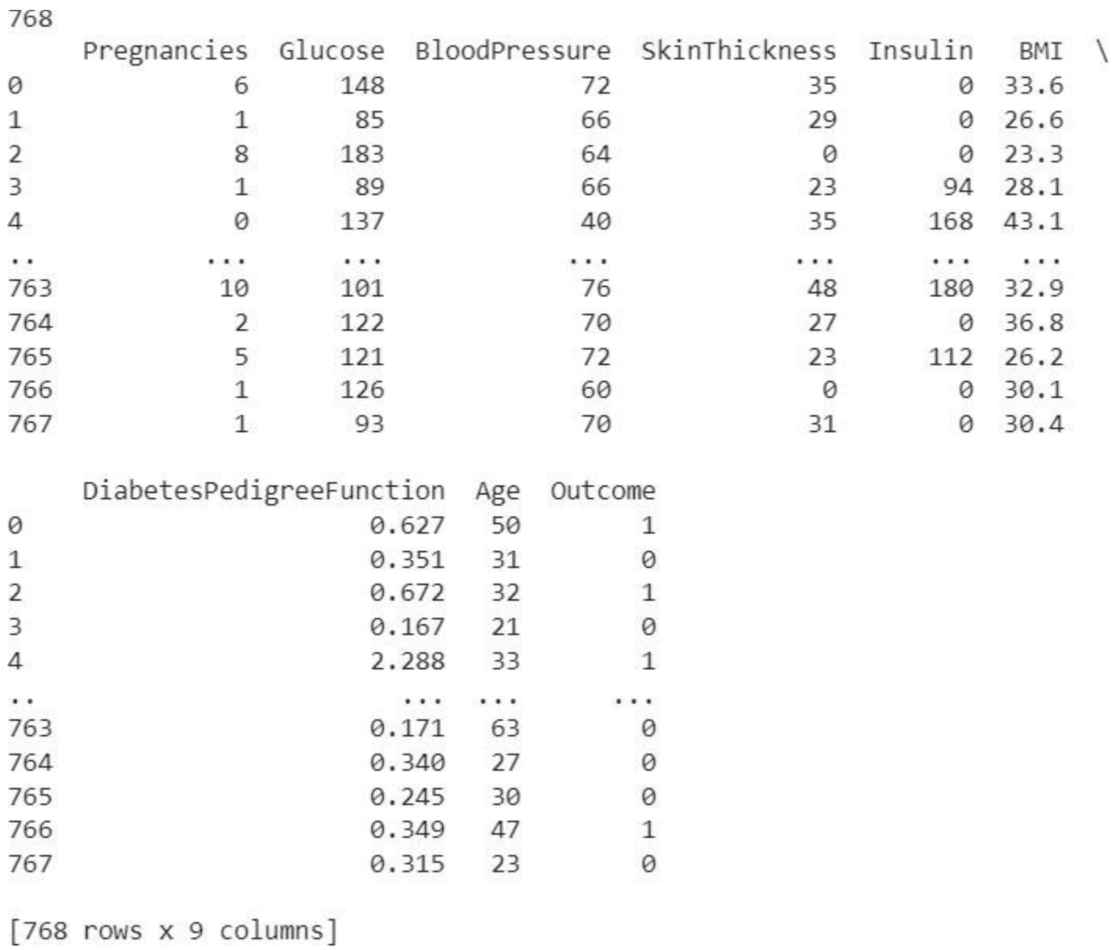
Step-8: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-9: Calculate the model performance by creating the confusion matrix using the test data and the predicted output

SOURCE CODE & OUTPUT:

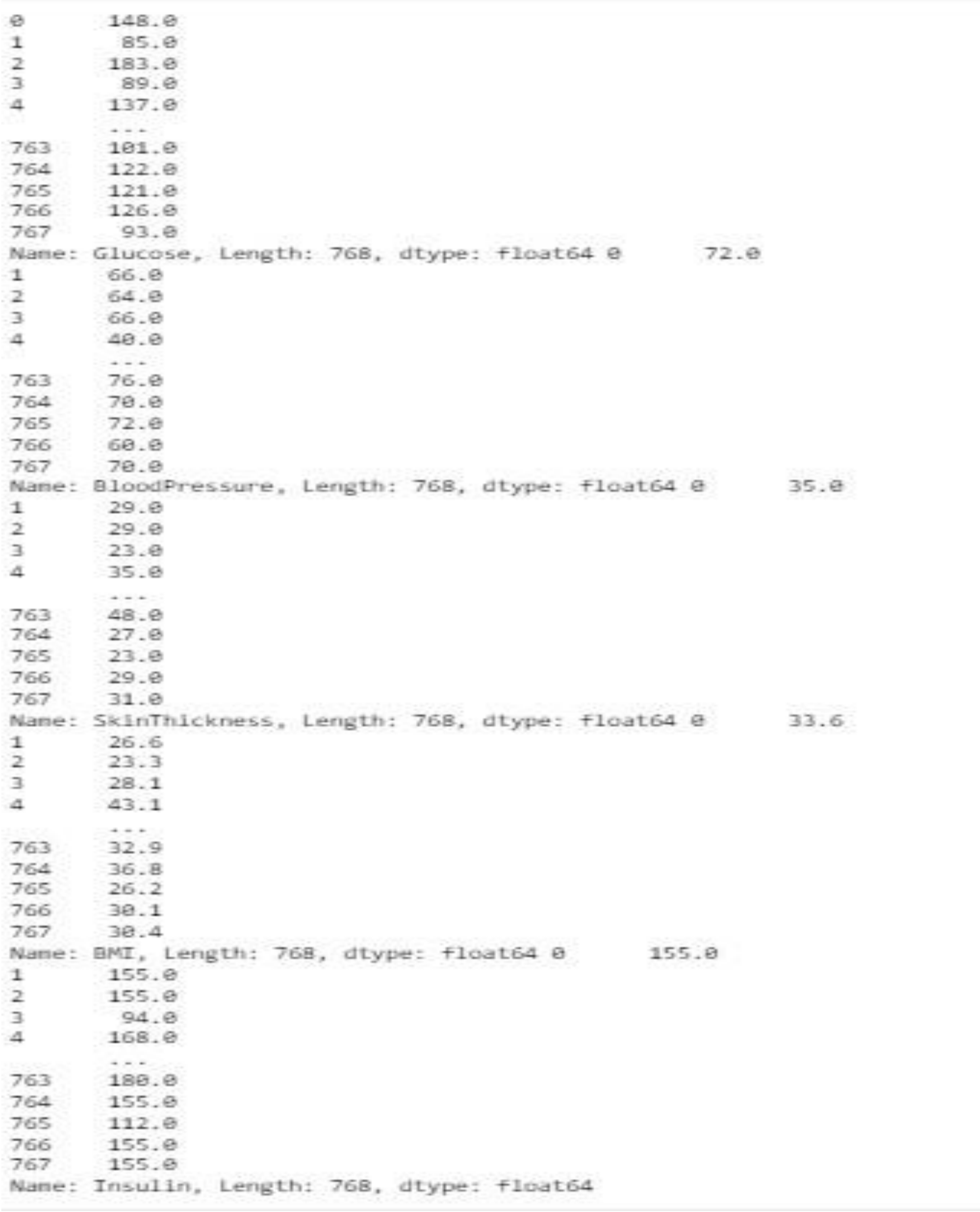
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

```
dataset = pd.read_csv('/content/diabetes.csv')
print(len(dataset))
print(dataset)
```



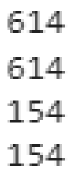
```
768
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

```
zero_not_accepted = ['Glucose', 'BloodPressure', 'SkinThickness', 'BMI', 'Insulin']
for column in zero_not_accepted:
    dataset[column] = dataset[column].replace(0, np.NaN)     mean = 
int(dataset[column].mean(skipna=True))# Calculate mean of dataset
    dataset[column] = dataset[column].replace(np.NaN, mean)
print(dataset['Glucose'],dataset['BloodPressure'],dataset['SkinThickness'],dataset['
BMI'],dataset['Insulin'])
```



```
0      148.0
1       85.0
2      183.0
3       89.0
4      137.0
       ...
763    101.0
764    122.0
765    121.0
766    126.0
767     93.0
Name: Glucose, Length: 768, dtype: float64 0      72.0
1      66.0
2      64.0
3      66.0
4      40.0
       ...
763    76.0
764    70.0
765    72.0
766    60.0
767    70.0
Name: BloodPressure, Length: 768, dtype: float64 0      35.0
1      29.0
2      29.0
3      23.0
4      35.0
       ...
763    48.0
764    27.0
765    23.0
766    29.0
767    31.0
Name: SkinThickness, Length: 768, dtype: float64 0      33.6
1      26.6
2      23.3
3      28.1
4      43.1
       ...
763    32.9
764    36.8
765    26.2
766    30.1
767    30.4
Name: BMI, Length: 768, dtype: float64 0      155.0
1      155.0
2      155.0
3       94.0
4      168.0
       ...
763    180.0
764    155.0
765    112.0
766    155.0
767    155.0
Name: Insulin, Length: 768, dtype: float64
```

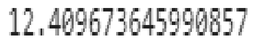
```
X = dataset.iloc[:, 0:8]
y = dataset.iloc[:, 8]
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=0,
test_size=0.2)
print(len(X_train))
print(len(y_train))
print(len(X_test))
print(len(y_test))
```



```
614
614
154
154
```

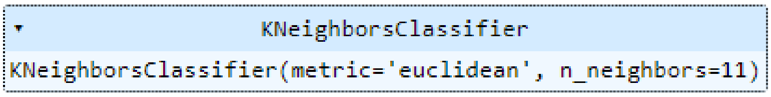
```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

```
import math
math.sqrt(len(y_test))
```



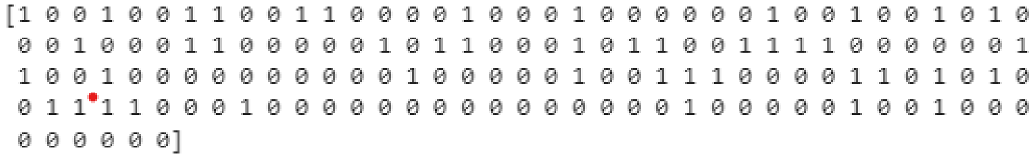
```
12.409673645990857
```

```
classifier = KNeighborsClassifier(n_neighbors=11,metric='euclidean')
classifier.fit(X_train, y_train)
```



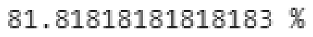
```
▾ KNeighborsClassifier
KNeighborsClassifier(metric='euclidean', n_neighbors=11)
```

```
y_pred=classifier.predict(X_test)
print(y_pred)
```



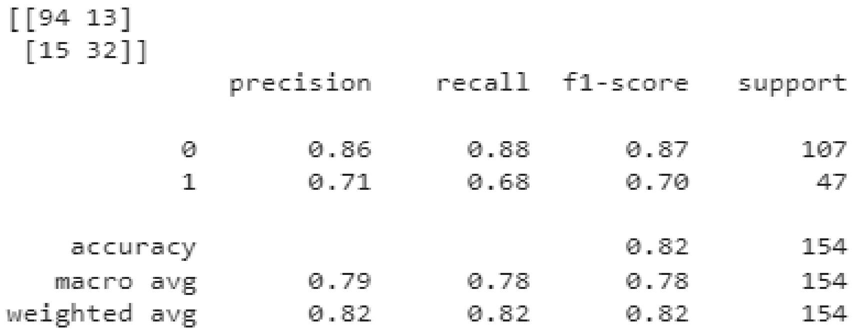
```
[1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0
 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 1
 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0
 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0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

```
print(accuracy_score(y_test,y_pred)*100,  '%')
```



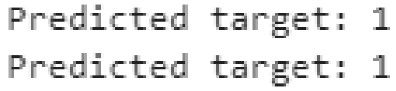
```
81.81818181818183 %
```

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



```
[[94 13]
 [15 32]]
              precision    recall  f1-score   support

           0       0.86      0.88      0.87       107
           1       0.71      0.68      0.70        47

    accuracy                           0.82       154
   macro avg       0.79      0.78      0.78       154
weighted avg       0.82      0.82      0.82       154
```

```
new_data = [[140, 72, 35, 0, 33.6, 0.627, 45, 1],
            [120, 70, 30, 1, 25.2, 0.2, 35, 0]]
predictions = classifier.predict(new_data)
for prediction in predictions:
    print(f"Predicted target: {prediction}")
```

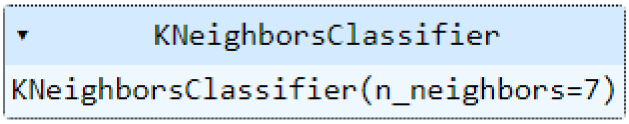


```
Predicted target: 1
Predicted target: 1
```

KNN WITH IRIS DATASET

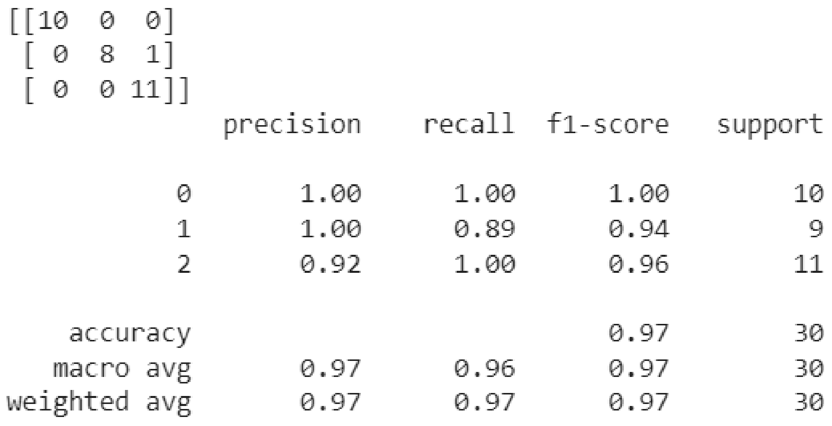
SOURCE CODE & OUTPUT:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
irisData = load_iris()

X = irisData.data
y = irisData.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state=42)
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
```



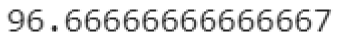
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=7)
```

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



```
[[10  0  0]
 [ 0  8  1]
 [ 0  0 11]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.89      0.94         9
           2       0.92      1.00      0.96        11

    accuracy                           0.97        30
   macro avg       0.97      0.96      0.97        30
weighted avg       0.97      0.97      0.97        30
```

```
print(accuracy_score(y_test,y_pred)*100)
```

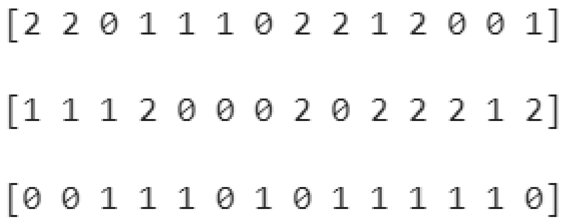


```
96.66666666666667
```

KNN WITH PREPROCESSING

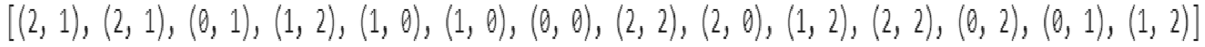
SOURCE CODE & OUTPUT:

```
weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny',
'Sunny','Rainy','Sunny','Overcast','Overcast','Rainy']

temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','
Hot','Mild']

play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']
from sklearn import preprocessing
le = preprocessing.LabelEncoder()

weather_encoded=le.fit_transform(weather)
print(weather_encoded)
print(" ")


temp_encoded=le.fit_transform(temp)
print(temp_encoded)
print(" ")

label=le.fit_transform(play)
print(label)
```



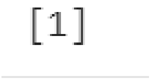
```
[2 2 0 1 1 1 0 2 2 1 2 0 0 1]
[1 1 1 2 0 0 0 2 0 2 2 2 1 2]
[0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

```
features=list(zip(weather_encoded,temp_encoded))
print(features)
```



```
[(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2), (2, 0), (1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]
```

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(features,label)
predicted= model.predict([[0,1]]) # 0:Overcast, 1:Hot
print(predicted)
```



```
[1]
```

RESULT

k-NN classification model on diabetes dataset is build and the accuracy of the algorithm is determined.

EXPERIMENT-3

NAIVE BAYES CLASSIFICATION ON IRIS DATASET

AIM: To implement Naive Bayes classification using any standard datasetavailable in the public domain and find the accuracy of the algorithm.

ALGORITHM:

Step 1:start

Step 2: Importing the standard libraries.

Step 3:Load the iris dataset-iris.csv The iris dataset contains the following data50 samples of 3 different species of iris (150 samples total) Measurements: sepal length, sepal width, petal length, petal width The format for the data: (sepal length, sepal width, petal length, petal width)

Step 4:Define x and y and label the fields

Step 5:Split the dataset into Training and testing

Step 6:Preprocess the dataset using StandardScaler StandardScaler removes the mean and scales each feature/variable to unit variance

Step 7:Train the data using GuassianNB model

Step 8:Test the data using Test set

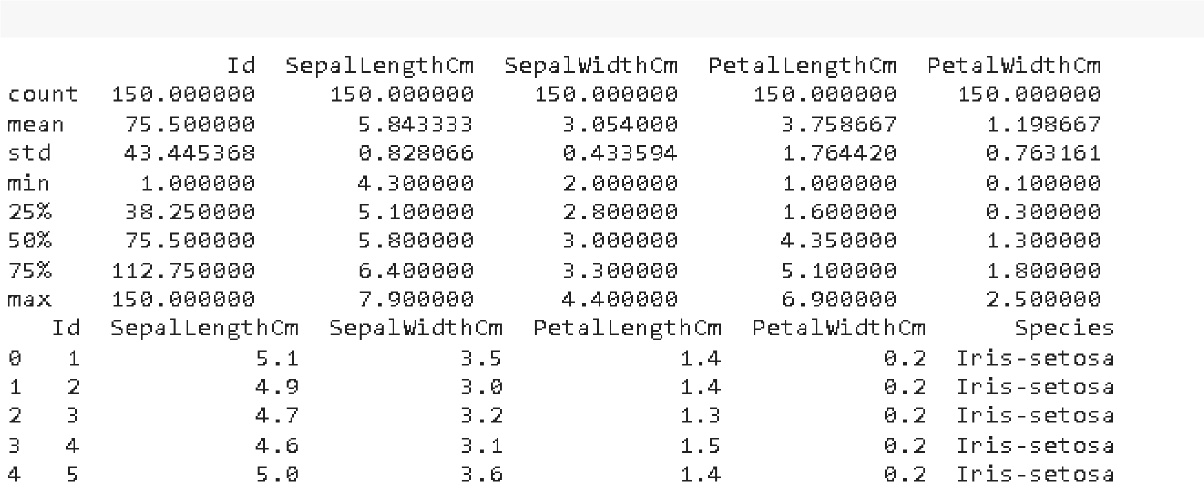
Step 9:Create the confusion matrix and Find the accuracy score

Step 10:Stop

SOURCE CODE & OUTPUT:

```
import pandas as pd
```

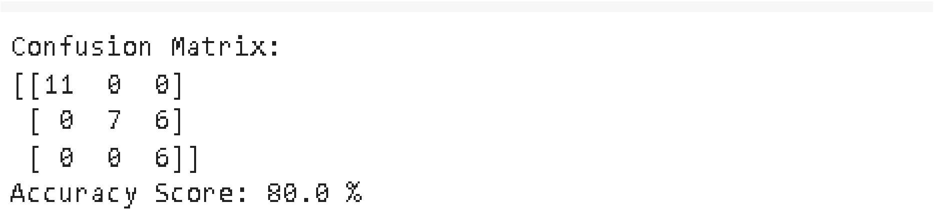
```
from sklearn.preprocessing import LabelEncoder from

sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.model_selection import train_test_split
import sklearn.naive_bayes
dataset = pd.read_csv('/content/Iris.csv')
print(dataset.describe())
print(dataset.head())
```



```
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

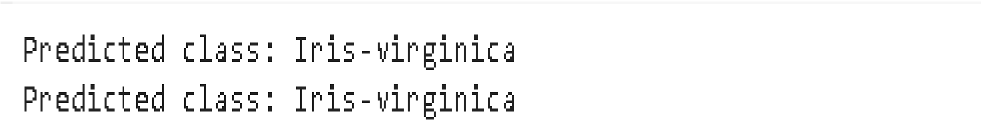
```
X = dataset.iloc[:, [1, 2, 3, 4]].valuesy =
dataset.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=0)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
classifier = sklearn.naive_bayes.GaussianNB()
```

```
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)ac

= accuracy_score(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
print("Accuracy Score:", ac*100,'%')
```



```
Confusion Matrix:
[[11  0  0]
 [ 0  7  6]
 [ 0  0  6]]
Accuracy Score: 80.0 %
```

```
new_data = [[5.1, 3.5, 1.4, 0.2],
            [6.2, 3.4, 5.4, 2.3]]
predictions = classifier.predict(new_data)

for prediction in predictions: print(f"Predicted
    class: {prediction}")
```



```
Predicted class: Iris-virginica
Predicted class: Iris-virginica
```

RESULT

Program executed successfully.

Experiment-4

Decision Tree Classifier

AIM: To implement decision tree using any standard dataset available in the public domain and find the accuracy of the algorithm.

ALGORITHM:

Step 1: Import the necessary packages and classes

Step 2: Load the Data Set

Step 3: Extract feature matrix and target from the data frame Step 4:

Split the data into training and testing sets

Step 5: Create a Decision Tree Classifier

Step 6: Train the classifier on the training data Step 7:

Make predictions on the test data

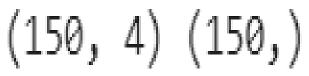
Step 8: Generate a confusion matrix and classification report Step

9: Visualize the decision tree

Step 10: Save the figure as an image

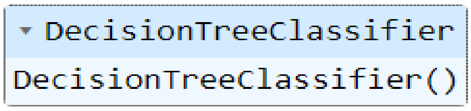
SOURCE CODE & OUTPUT:

```
import numpy as np import
pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split from
sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import plot_tree
from sklearn import tree
data=load_iris()
X=data.data y=data.target
print(X.shape,y.shape)
```



```
(150, 4) (150,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=10)
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
```

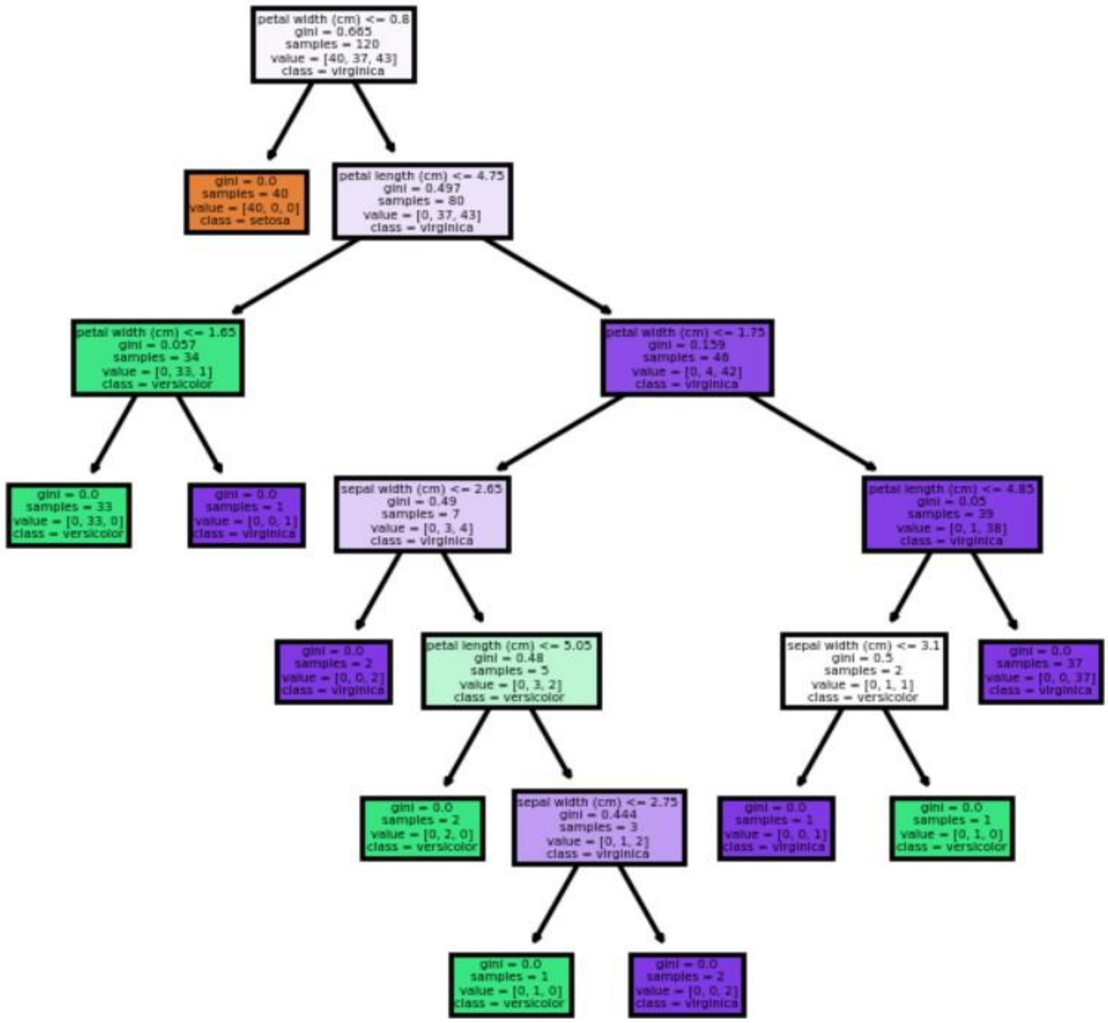


```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

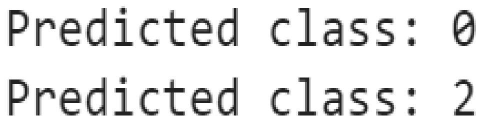
```
y_pred =dtc.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
print("Classification report - \n", classification_report(y_test,y_pred)) fig,
axes = plt.subplots(nrows=1, ncols=1, figsize=(4, 4), dpi=200)
tree.plot_tree(dtc, feature_names=data.feature_names,
class_names=data.target_names, filled=True)

plt.show()
```

```
fig.savefig("iris_tree.png")
```

```
Confusion Matrix:
[[10  0  0]
 [ 0 12  1]
 [ 0  0  7]]
Classification report -
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.92      0.96        13
           2       0.88      1.00      0.93         7

    accuracy                           0.97        30
   macro avg       0.96      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```



```
new_data = [[5.1, 3.5, 1.4, 0.2],[6.2, 3.4, 5.4, 2.3]]
predictions = dtc.predict(new_data)
for prediction in predictions:
    print(f"Predicted class: {prediction}")
```



```
Predicted class: 0
Predicted class: 2
```

RESULT

Program executed successfully.

Experiment-5

Simple Linear Regression

AIM: To predict the salary based on the number of years of experience.

ALGORITHM:

Step 1: Load the data set

Step 2: Extract the features and labels from the dataframe

Step 3: Split the dataset into the Training set and Test set

Step 4: Perform data visualization on train and test data

Step 5: Initialize a Linear Regression model and fit the model on the training data

Step 6: Predict on the test data

Step 7: Calculate the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R2)

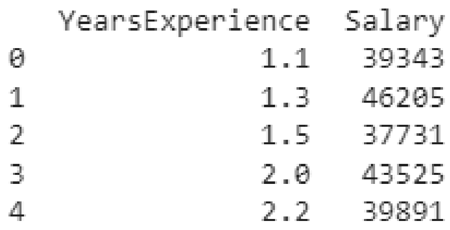
Step 8: Predict on a new data

Step 9: Stop

SOURCE CODE & OUTPUT:

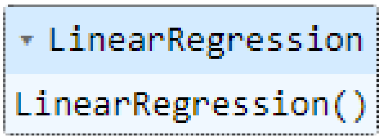
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
dataset = pd.read_csv('/content/Salary_data.csv')
X = dataset.iloc[:, :-1].values
```

```
y = dataset.iloc[:, 1].values
print(dataset.head())
```



```
   YearsExperience  Salary
0              1.1   39343
1              1.3   46205
2              1.5   37731
3              2.0   43525
4              2.2   39891
```

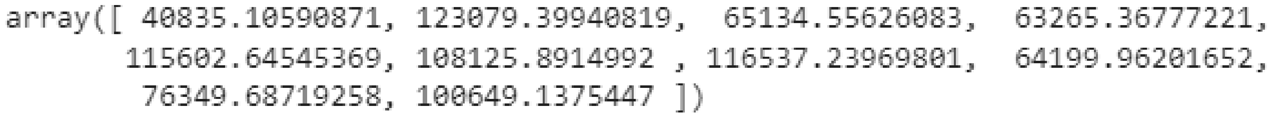
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=0)
plt.scatter(X_train, y_train, color='red', label='Actual')
plt.plot(X_train, regressor.predict(X_train), color='blue', label='Predicted')
plt.title('Salary VS Experience (Training set)')
plt.xlabel('Year of Experience')
plt.ylabel('Salary')
plt.legend()  # Add a legend to distinguish between actual and predicted data
plt.show()
plt.scatter(X_test, y_test, color='red', label='Actual')
plt.plot(X_train, regressor.predict(X_train), color='blue', label='Predicted')
plt.title('Salary VS Experience (Test set)')
plt.xlabel('Year of Experience')
plt.ylabel('Salary')
plt.legend()  # Add a legend to distinguish between actual and predicted data
plt.show()
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```



```
▾ LinearRegression
LinearRegression()
```

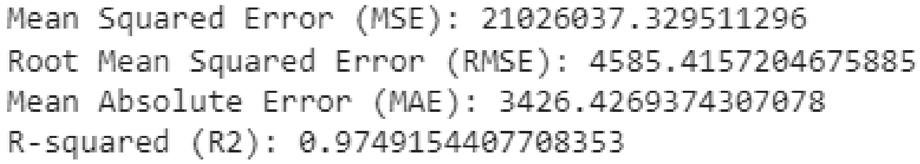
```
y_pred = regressor.predict(X_test)
```

```
y_pred
```



```
array([ 40835.10590871, 123079.39940819,  65134.55626083,  63265.36777221,
       115602.64545369, 108125.8914992 , 116537.23969801,  64199.96201652,
        76349.68719258, 100649.1375447 ])
```

```
mse = mean_squared_error(y_test, y_pred)

rmse = np.sqrt(mse)

mae = mean_absolute_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)

print("Root Mean Squared Error (RMSE):", rmse)

print("Mean Absolute Error (MAE):", mae)

print("R-squared (R2):", r2)
```



```
Mean Squared Error (MSE): 21026037.329511296
Root Mean Squared Error (RMSE): 4585.4157204675885
Mean Absolute Error (MAE): 3426.4269374307078
R-squared (R2): 0.9749154407708353
```

```
new_input = [[5]]

y_pred = regressor.predict(new_input)

print("Predicted Salary:", y_pred)
```

RESULT

Program executed successfully.

Experiment 5.B

Multiple Linear Regression

AIM: To predict the salary based on the number of years of experience.

ALGORITHM

Step 1: Load the data set

Step 2: Extract the features and labels from the dataframe, Define X and y

Step 3: Split the dataset into the Training set and Test set

Step 4: Initialize a Linear Regression model and fit the model on the training data

Step 5: Predict on the test data

Step 6: Perform data Visualization

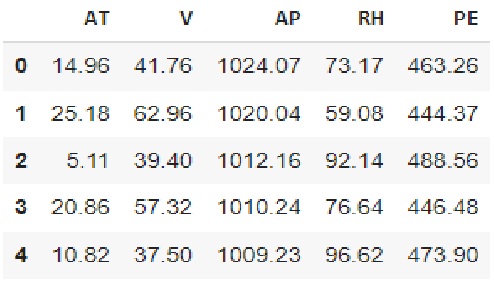
Step 7: Calculate the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R2)

Step 8: Predict on a new data

Step 9: Stop

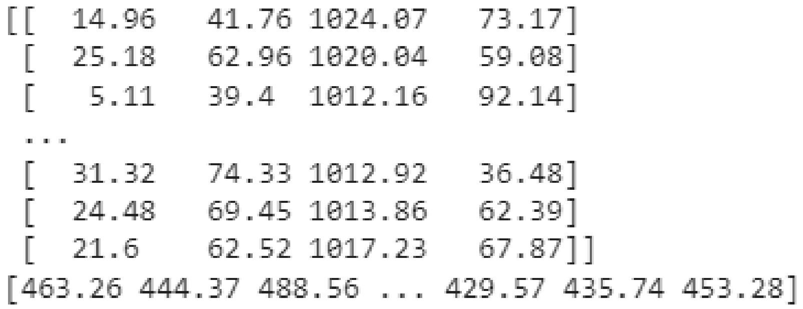
SOURCE CODE & OUTPUT:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
data_df = pd.read_excel('/content/CCCP.xlsx')
data_df.head()
```



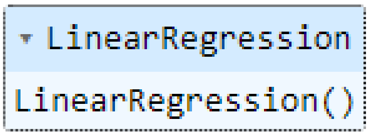
	AT	V	AP	RH	PE
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90

```
x = data_df.drop(['PE'], axis=1).values

print(x)

y = data_df['PE'].values

print(y)
```



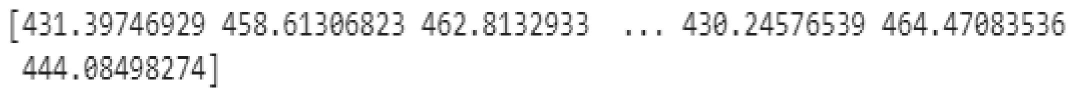
```
[[  14.96   41.76 1024.07   73.17]
 [  25.18   62.96 1020.04   59.08]
 [   5.11   39.4  1012.16   92.14]
 ...
 [  31.32   74.33 1012.92   36.48]
 [  24.48   69.45 1013.86   62.39]
 [  21.6    62.52 1017.23   67.87]]
[463.26 444.37 488.56 ... 429.57 435.74 453.28]
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=0)

regressor = LinearRegression()

regressor.fit(x_train, y_train)
```



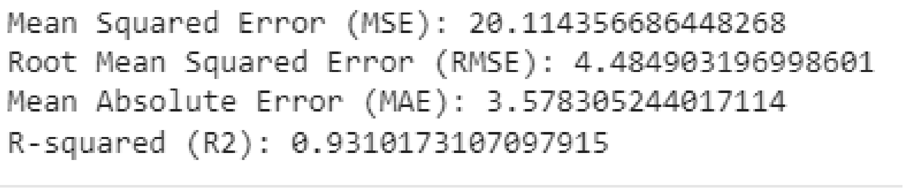
```
▾ LinearRegression
LinearRegression()
```

```
y_pred = regressor.predict(x_test)

print(y_pred)
```



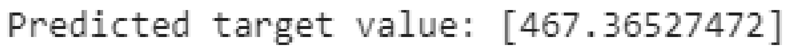
```
[431.39746929 458.61306823 462.8132933  ... 430.24576539 464.47083536
 444.08498274]
```

```
plt.figure(figsize=(15, 10))

plt.scatter(y_test, y_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('ACTUAL VS
PREDICTED')
plt.show()
mse = mean_squared_error(y_test,
y_pred)rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test,
y_pred)r2 = r2_score(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):",
rmse)print("Mean Absolute Error (MAE):",
mae) print("R-squared (R2):", r2)
```



```
Mean Squared Error (MSE): 20.114356686448268
Root Mean Squared Error (RMSE): 4.484903196998601
Mean Absolute Error (MAE): 3.578305244017114
R-squared (R2): 0.9310173107097915
```

```
new_input = [[14.96, 41.76, 1024.07, 73.17]]
y_pred = regressor.predict(new_input)
print("Predicted target value:", y_pred)
```



```
Predicted target value: [467.36527472]
```

RESULT

Program executed successfully.

EXPERIMENT-6

Convolutional Neural Network

AIM: Programs to implement Convolutional Neural Network to classify images from any standard dataset in the public domain using Keras framework.

ALGORITHM:

Step 1: Import Libraries:

Import the deep learning framework of your choice (e.g., TensorFlow, PyTorch).
Import other necessary libraries (e.g., NumPy for numerical operations).
Step 2:Load and Preprocess Data:

Load your dataset (images and corresponding labels).
Preprocess the data (normalize, resize, etc.).
Step 3: Define the CNN Architecture:

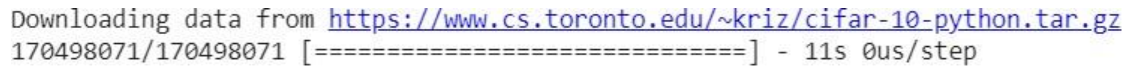
Define the layers of your CNN, including convolutional layers, pooling layers, fully connected layers, etc.
Step 4: Compile the Model:

Specify the optimizer, loss function, and metrics.
Step 5: Train the Model:
Feed the training data into the model and adjust the weights using backpropagation.
Step 6: Evaluate the Model:

Evaluate the performance of the trained model on the test set.
Step 7: Make Predictions:
Use the trained model to make predictions on new data.

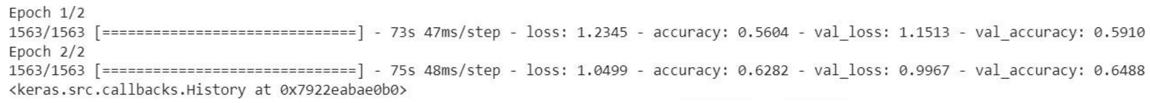
SOURCE CODE & OUTPUT:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import cifar10
from PIL import Image
import numpy as np
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```



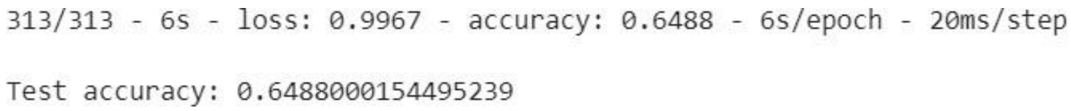
```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 11s 0us/step
```

```
X_train, X_test = X_train / 255.0, X_test / 255.0

model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10)  # 10 output classes
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.fit(X_train, y_train, epochs=2, validation_data=(X_test, y_test))
```



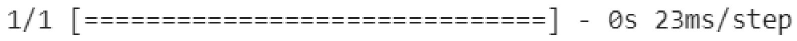
```
Epoch 1/2
1563/1563 [==============================] - 73s 47ms/step - loss: 1.2345 - accuracy: 0.5604 - val_loss: 1.1513 - val_accuracy: 0.5910
Epoch 2/2
1563/1563 [==============================] - 75s 48ms/step - loss: 1.0499 - accuracy: 0.6282 - val_loss: 0.9967 - val_accuracy: 0.6488
<keras.src.callbacks.History at 0x7922eabae0b0>
```

```
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print("\nTest accuracy:", test_acc)
```



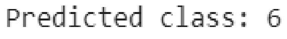
```
313/313 - 6s - loss: 0.9967 - accuracy: 0.6488 - 6s/epoch - 20ms/step

Test accuracy: 0.6488000154495239
```

```
image_path = '/content/new_image.jpeg'
image = Image.open(image_path).resize((32, 32))
image = np.array(image) / 255.0
image = np.expand_dims(image, axis=0)
predictions = model.predict(image)
```



```
1/1 [==============================] - 0s 23ms/step
```

```
predicted_class = np.argmax(predictions)
print(f'Predicted class: {predicted_class}')
```



```
Predicted class: 6
```

RESULT:

Program Executed Successfully.

EXPERIMENT-7

Support Vector Machine

AIM: Program to implement text classification using Support vector machine.

ALGORITHM:

1. Add the Required Libraries
2. Set random seed [This is used to reproduce the same result every time if the script is kept consistent otherwise each run will produce different results. The seed can be set to any number.]
3. Load the dataset "Corpus" [The data set can be easily added as a pandas Data Frame with the help of ‘read_csv’ function. I have set the encoding to ‘latin-1’ as the text had many special characters.]
4. Data pre-processing [This basically involves transforming raw data into an understandable format for NLP models.]

   (a) Remove Blank rows in Data, if any

   (b) Change all the text to lower case

   (c) Word Tokenization

   (d) Remove Stop words

   (e) Remove Non-alpha text

   (f) Word Lemmatization
5. Prepare Train and Test Data sets [This can be done through the *train_test_split* from the sklearn library. The Training Data will have 70% of the corpus and Test data will have the remaining 30% as we have

set the parameter test_size=0.3]

6. Encoding [Label encode the target variable — This is done to transform Categorical data of string type in the data set into numerical values which the model can understand.]
7. Word Vectorization [It is a general process of turning a collection of text documents into numerical feature vectors. Most popular method is called TF-IDF (“Term Frequency — Inverse Document” Frequency) which are the components of the resulting scores assigned to each word.]

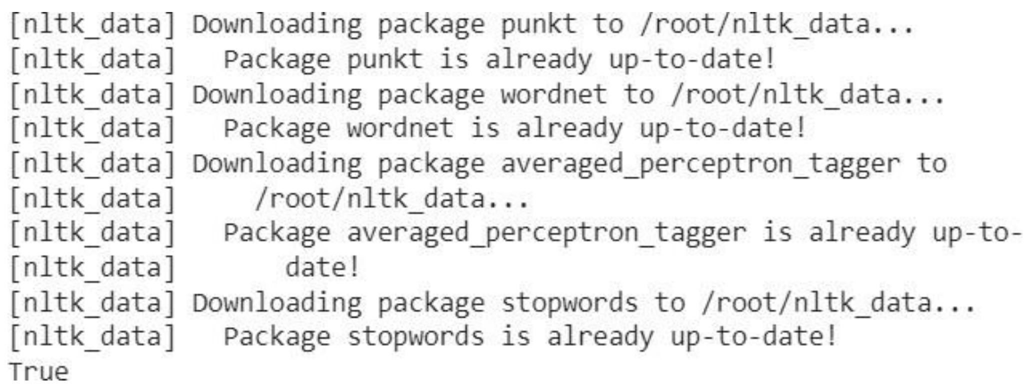
(a) Term Frequency: This summarizes how often a given word appears within a document.

(b) Inverse Document Frequency: This down scales words that appear alot across documents.

8. Use the SVM to Predict the outcome and display the accuracy of prediction.

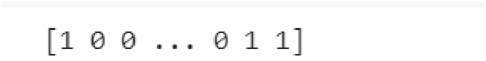
SOURCE CODE & OUTPUT:

```
import pandas as pd
import numpy as np
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
nltk.download('punkt')



nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
```



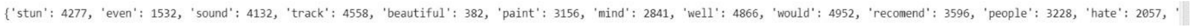
```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
np.random.seed(500)Corpus =
pd.read_csv("/content/corpus.csv",encoding='latin-1')
Corpus['text'].dropna(inplace=True)
Corpus['text'] = [entry.lower() for entry in Corpus['text']]
Corpus['text']= [word_tokenize(entry) for entry in Corpus['text']]
tag_map = defaultdict(lambda : wn.NOUN)
tag_map['J'] = wn.ADJ
tag_map['V'] = wn.VERB
tag_map['R'] = wn.ADV
for index,entry in enumerate(Corpus['text']):
 Final_words = []
 word_Lemmatized = WordNetLemmatizer()
   for word, tag in pos_tag(entry):
        if word not in stopwords.words('english') and word.isalpha():
            word_Final = word_Lemmatized.lemmatize(word,tag_map[tag[0]])
            Final_words.append(word_Final)
    Corpus.loc[index,'text_final'] = str(Final_words)



  Train_X, Test_X, Train_Y, Test_Y =
model_selection.train_test_split(Corpus['text_final'],Corpus['label'],test_size=0.3)
Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
Test_Y = Encoder.fit_transform(Test_Y)
print(Train_Y)
```



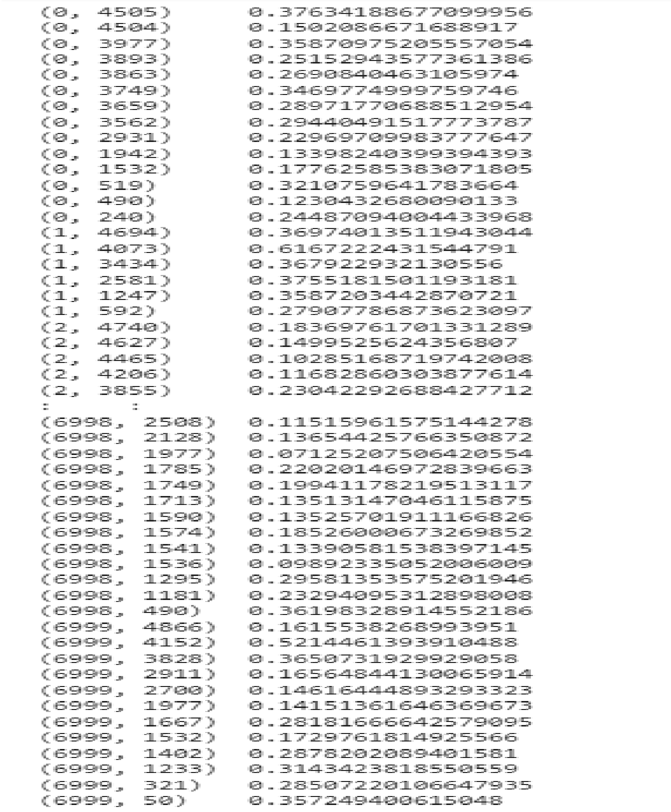
```
[1 0 0 ... 0 1 1]
```

```
Tfidf_vect = TfidfVectorizer(max_features=5000)
Tfidf_vect.fit(Corpus['text_final'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)
```



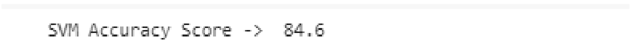
```
{'stun': 4277, 'even': 1532, 'sound': 4132, 'track': 4558, 'beautiful': 382, 'paint': 3156, 'mind': 2841, 'well': 4866, 'would': 4952, 'recomend': 3596, 'people': 3228, 'hate': 2057, '
```

```
print(Tfidf_vect.vocabulary_)
print(Train_X_Tfidf)
```



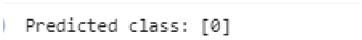
```
(0, 4505)         0.37634188677099956
(0, 4504)         0.1502086671688917
(0, 3977)         0.35870975205557054
(0, 3893)         0.25152943577361386
(0, 3863)         0.2690840463105974
(0, 3749)         0.3469774999759746
(0, 3659)         0.28971770688512954
(0, 3562)         0.29440491517773787
(0, 2931)         0.22969709983777647
(0, 1942)         0.13398240399394393
(0, 1532)         0.17762585383071805
(0, 519)          0.3210759641783664
(0, 490)          0.1230432680090133
(0, 240)          0.24487094004433968
(1, 4694)         0.36974013511943044
(1, 4073)         0.6167222431544791
(1, 3434)         0.367922932130556
(1, 2581)         0.3755181501193181
(1, 1247)         0.3587203442870721
(1, 592)          0.27907786873623097
(2, 4740)         0.18369761701331289
(2, 4627)         0.1499525624356807
(2, 4465)         0.10285168719742008
(2, 4206)         0.11682860303877614
(2, 3855)         0.23042292688427712
:      :
(6998, 2508)      0.11515961575144278
(6998, 2128)      0.13654425766350872
(6998, 1977)      0.07125207506420554
(6998, 1785)      0.22020146972839663
(6998, 1749)      0.19941178219513117
(6998, 1713)      0.13513147046115875
(6998, 1590)      0.13525701911166826
(6998, 1574)      0.18526000673269852
(6998, 1541)      0.13390581538397145
(6998, 1536)      0.09892335052006009
(6998, 1295)      0.29581353575201946
(6998, 1181)      0.23294095312898008
(6998, 490)       0.36198328914552186
(6999, 4866)      0.1615538268993951
(6999, 4152)      0.5214461393910488
(6999, 3828)      0.3650731929929058
(6999, 2911)      0.16564844130065914
(6999, 2700)      0.14616444893293323
(6999, 1977)      0.14151361646369673
(6999, 1667)      0.28181666642579095
(6999, 1532)      0.1729761814925566
(6999, 1402)      0.2878202089401581
(6999, 1233)      0.3143423818550559
(6999, 321)       0.28507220106647935
(6999, 50)        0.357249400615048
```

```
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM.fit(Train_X_Tfidf,Train_Y)
predictions_SVM = SVM.predict(Test_X_Tfidf)
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM,
Test_Y)*100)
```



```
SVM Accuracy Score ->  84.6
```

```
new_text = "This is a new text that needs to be classified."
tokens = word_tokenize(new_text)
tokens = [token.lower() for token in tokens]
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
stop_words = set(stopwords.words("english"))
filtered_tokens = [token for token in tokens if token not in stop_words]
preprocessed_text = ' '.join(tokens)
new_text_vector = Tfidf_vect.transform([preprocessed_text])
predicted_class = SVM.predict(new_text_vector)
print("Predicted class:", predicted_class)
```



```
Predicted class: [0]
```

RESULT:

Program Executed Successfully.

Experiment-8

K – Means Clustering

AIM: To implement k-means clustering technique using any standard dataset available in the public domain.

ALGORITHM:

Step 1: Load the Dataset

Step 2: Do the scatter plot and see that clusters are evident

Step 3: Create an instance of K-Means

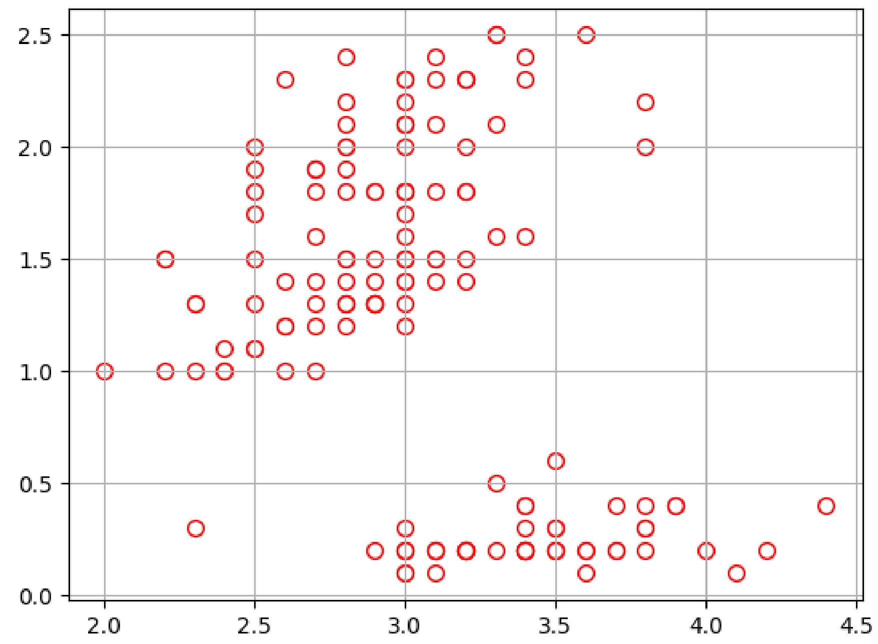
Step 4: Fit and make predictions

Step 5: Create the K-means cluster plot

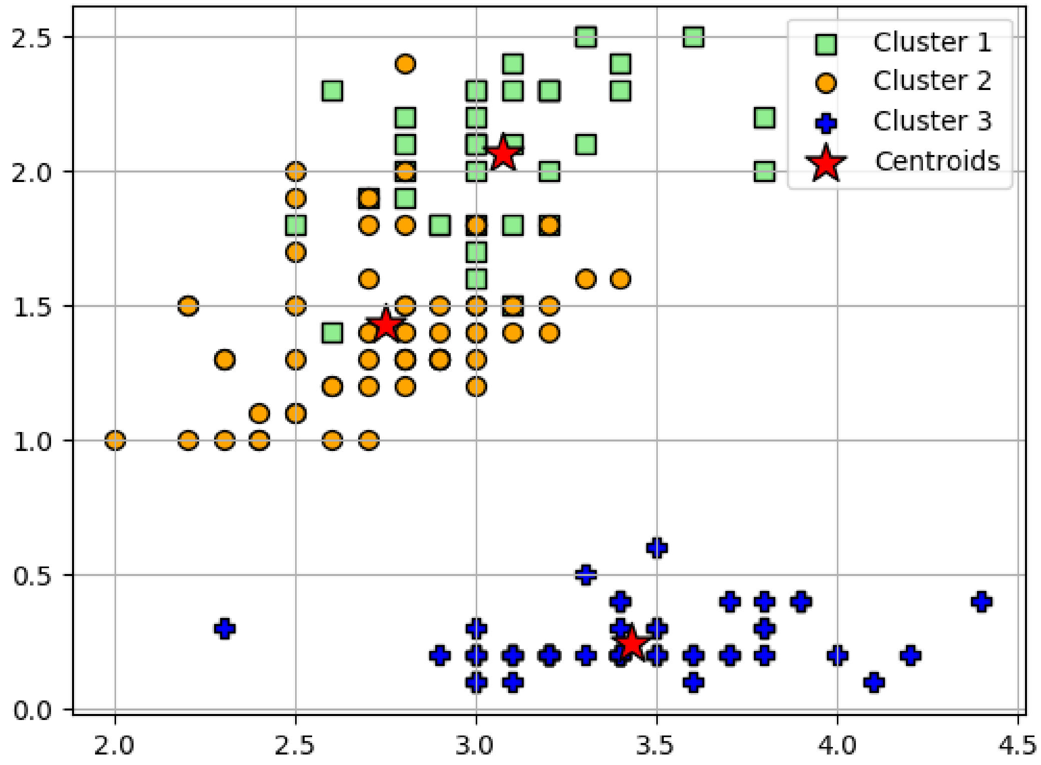
Step 6: Stop

SOURCE CODE & OUTPUT:

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
iris = datasets.load_iris()
X = iris.data
y = iris.target
plt.scatter(X[:,1], X[:,3], color='white', marker='o', edgecolor='red', s=50)
plt.grid()
plt.show()
```



```
kmc = KMeans(n_clusters=3)
y_kmc = kmc.fit_predict(X)
plt.scatter(X[y_kmc == 0, 1], X[y_kmc == 0, 3], s=50,c='lightgreen', marker='s',
edgecolor='black', label='Cluster 1')
plt.scatter(X[y_kmc == 1, 1], X[y_kmc == 1, 3], s=50,c='orange', marker='o',
edgecolor='black', label='Cluster 2')
plt.scatter(X[y_kmc == 2, 1], X[y_kmc == 2, 3], s=50,c='blue', marker='P',
edgecolor='black', label='Cluster 3')
plt.scatter(kmc.cluster_centers_[:, 1], kmc.cluster_centers_[:, 3],s=250,
marker='*', c='red', edgecolor='black', label='Centroids')
plt.legend()
plt.grid()
plt.show()
```



RESULT

Program executed successfully.

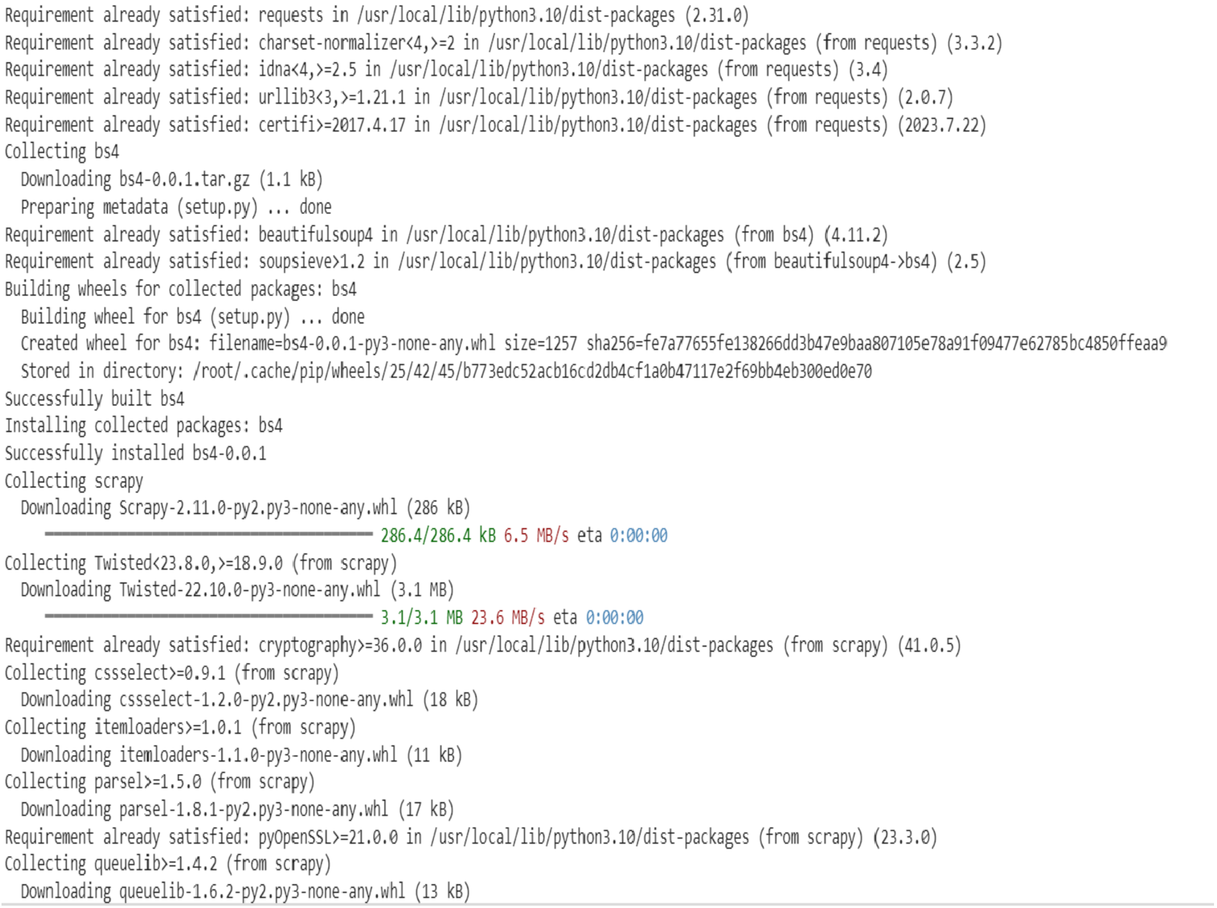
Experiment-9A

Web Crawler

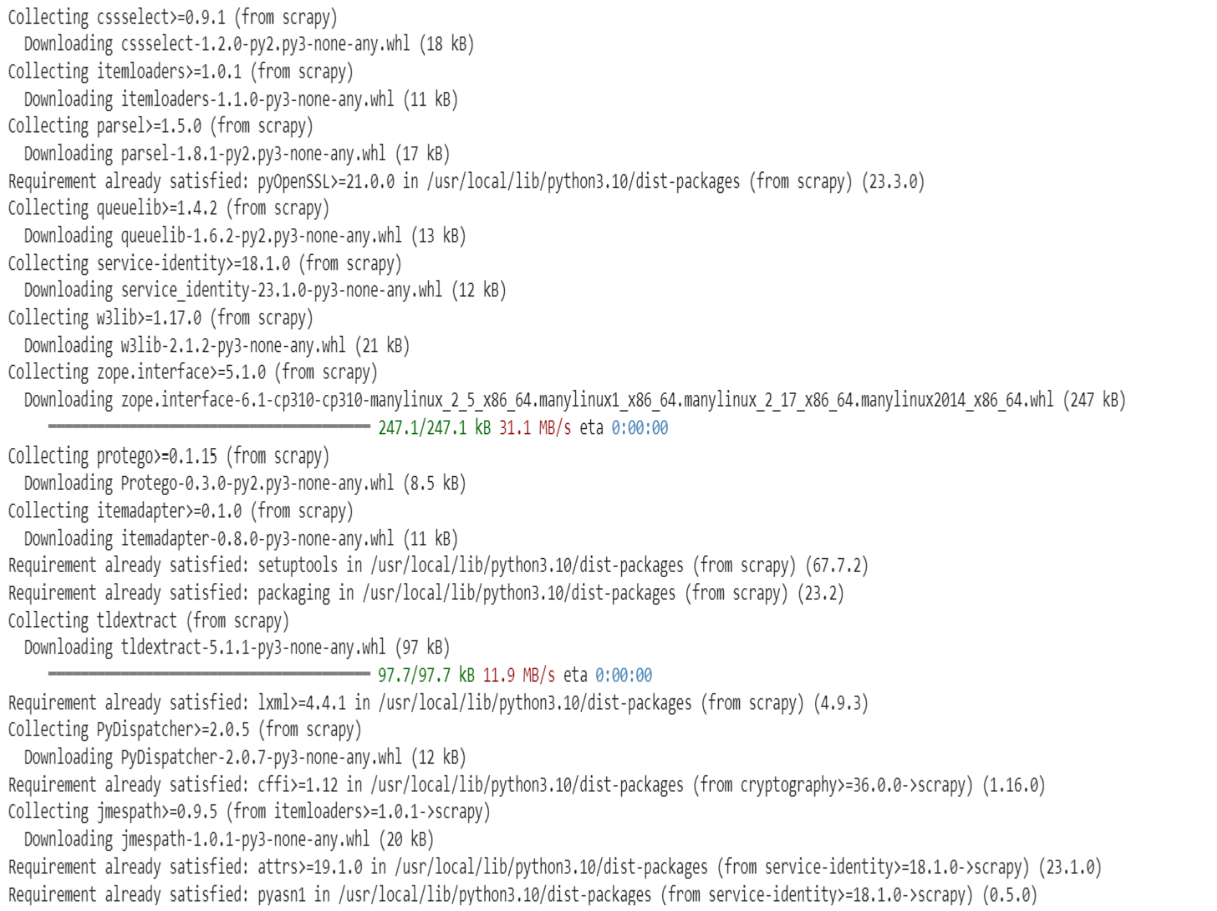
AIM: To implement a simple web crawler

SOURCE CODE & OUTPUT:

```
!pip install requests
!pip install bs4
!pip install scrapy
import logging
from urllib.parse import urljoin
import requests
from bs4 import BeautifulSoup
logging.basicConfig(
    format='%(asctime)s %(levelname)s:%(message)s',
    level=logging.INFO
 )
class Crawler:
    def__init__(self, urls=[]):
        self.visited_urls = []
        self.urls_to_visit = urls

    def download_url(self, url):
        return requests.get(url).text
    def get_linked_urls(self, url, html):
        soup = BeautifulSoup(html, 'html.parser')
        for link in soup.find_all('a'):
            path = link.get('href')
            if path and path.startswith('/'):
                path = urljoin(url, path)
                yield path
    def add_url_to_visit(self, url):
        if url not in self.visited_urls and url not in self.urls_to_visit:
            self.urls_to_visit.append(url)
    def crawl(self, url):
        html = self.download_url(url)
        for url in self.get_linked_urls(url, html):
            self.add_url_to_visit(url)
    def run(self):
        while self.urls_to_visit:
            url = self.urls_to_visit.pop(0)
            logging.info(f'Crawling: {url}')
            try:
                self.crawl(url)
            except Exception:
                logging.exception(f'Failed to crawl: {url}')
            finally:
                self.visited_urls.append(url)


if__name__== '__main__':
    Crawler(urls=['https://docs.python.org/']).run()
```



```
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2023.7.22)
Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from bs4) (4.11.2)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->bs4) (2.5)
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py) ... done
  Created wheel for bs4: filename=bs4-0.0.1-py3-none-any.whl size=1257 sha256=fe7a77655fe138266dd3b47e9baa807105e78a91f09477e62785bc4850ffeaa9
  Stored in directory: /root/.cache/pip/wheels/25/42/45/b773edc52acb16cd2db4cf1a0b47117e2f69bb4eb300ed0e70
Successfully built bs4
Installing collected packages: bs4
Successfully installed bs4-0.0.1
Collecting scrapy
  Downloading Scrapy-2.11.0-py2.py3-none-any.whl (286 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 286.4/286.4 kB 6.5 MB/s eta 0:00:00
Collecting Twisted<23.8.0,>=18.9.0 (from scrapy)
  Downloading Twisted-22.10.0-py3-none-any.whl (3.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 23.6 MB/s eta 0:00:00
Requirement already satisfied: cryptography>=36.0.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (41.0.5)
Collecting cssselect>=0.9.1 (from scrapy)
  Downloading cssselect-1.2.0-py2.py3-none-any.whl (18 kB)
Collecting itemloaders>=1.0.1 (from scrapy)
  Downloading itemloaders-1.1.0-py3-none-any.whl (11 kB)
Collecting parsel>=1.5.0 (from scrapy)
  Downloading parsel-1.8.1-py2.py3-none-any.whl (17 kB)
Requirement already satisfied: pyOpenSSL>=21.0.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (23.3.0)
Collecting queuelib>=1.4.2 (from scrapy)
  Downloading queuelib-1.6.2-py2.py3-none-any.whl (13 kB)
```



```
Collecting cssselect>=0.9.1 (from scrapy)
  Downloading cssselect-1.2.0-py2.py3-none-any.whl (18 kB)
Collecting itemloaders>=1.0.1 (from scrapy)
  Downloading itemloaders-1.1.0-py3-none-any.whl (11 kB)
Collecting parsel>=1.5.0 (from scrapy)
  Downloading parsel-1.8.1-py2.py3-none-any.whl (17 kB)
Requirement already satisfied: pyOpenSSL>=21.0.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (23.3.0)
Collecting queuelib>=1.4.2 (from scrapy)
  Downloading queuelib-1.6.2-py2.py3-none-any.whl (13 kB)
Collecting service-identity>=18.1.0 (from scrapy)
  Downloading service_identity-23.1.0-py3-none-any.whl (12 kB)
Collecting w3lib>=1.17.0 (from scrapy)
  Downloading w3lib-2.1.2-py3-none-any.whl (21 kB)
Collecting zope.interface>=5.1.0 (from scrapy)
  Downloading zope.interface-6.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (247 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247.1/247.1 kB 31.1 MB/s eta 0:00:00
Collecting protego>=0.1.15 (from scrapy)
  Downloading Protego-0.3.0-py2.py3-none-any.whl (8.5 kB)
Collecting itemadapter>=0.1.0 (from scrapy)
  Downloading itemadapter-0.8.0-py3-none-any.whl (11 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from scrapy) (67.7.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from scrapy) (23.2)
Collecting tldextract (from scrapy)
  Downloading tldextract-5.1.1-py3-none-any.whl (97 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.7/97.7 kB 11.9 MB/s eta 0:00:00
Requirement already satisfied: lxml>=4.4.1 in /usr/local/lib/python3.10/dist-packages (from scrapy) (4.9.3)
Collecting PyDispatcher>=2.0.5 (from scrapy)
  Downloading PyDispatcher-2.0.7-py3-none-any.whl (12 kB)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-packages (from cryptography>=36.0.0->scrapy) (1.16.0)
Collecting jmespath>=0.9.5 (from itemloaders>=1.0.1->scrapy)
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Requirement already satisfied: attrs>=19.1.0 in /usr/local/lib/python3.10/dist-packages (from service-identity>=18.1.0->scrapy) (23.1.0)
Requirement already satisfied: pyasn1 in /usr/local/lib/python3.10/dist-packages (from service-identity>=18.1.0->scrapy) (0.5.0)
```

RESULT

Program executed successfully.

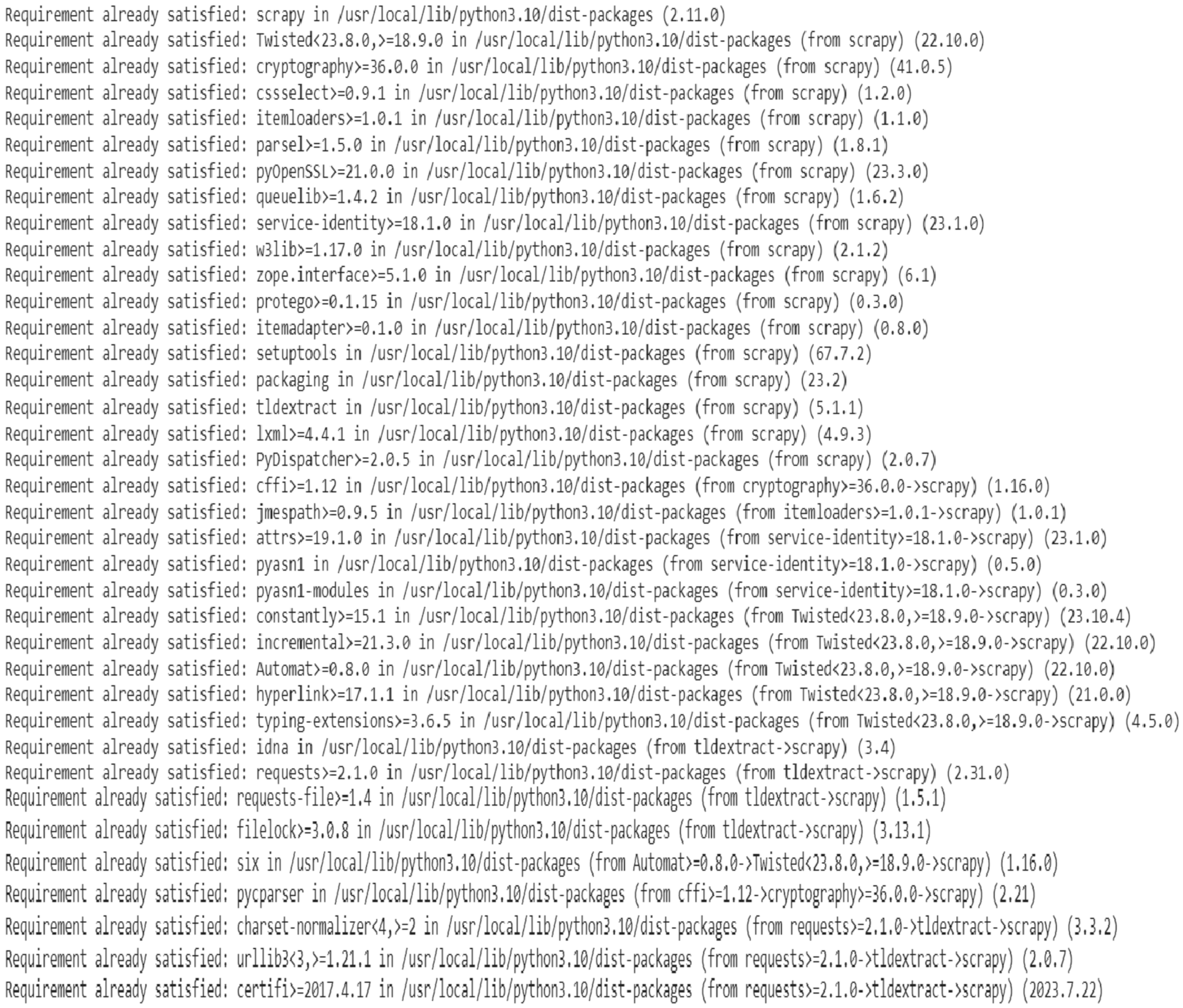
Experiment-9B

Web Scrapping

AIM: To implement a program to scrap a web page of any website.

SOURCE CODE & OUTPUT:

```
!pip install scrapy
import scrapy
from scrapy.crawler import CrawlerProcess


  class QuotesSpider(scrapy.Spider):
      name = 'quotes'
      start_urls = ['http://quotes.toscrape.com/']

      def parse(self, response):
          for quote in response.css('div.quote'):
              text = quote.css('span.text::text').get()
              author = quote.css('small::text').get()
              print(f'Text: {text}\nAuthor: {author}\n{"-"*40}')

          next_page = response.css('li.next a::attr(href)').get()
          if next_page:
              yield response.follow(next_page, self.parse)


  if__name__== "__main__":
      process = CrawlerProcess({
        'USER_AGENT': 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)',
    })


process.crawl(QuotesSpider)
process.start()
```



```
Requirement already satisfied: scrapy in /usr/local/lib/python3.10/dist-packages (2.11.0)
Requirement already satisfied: Twisted<23.8.0,>=18.9.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (22.10.0)
Requirement already satisfied: cryptography>=36.0.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (41.0.5)
Requirement already satisfied: cssselect>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from scrapy) (1.2.0)
Requirement already satisfied: itemloaders>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from scrapy) (1.1.0)
Requirement already satisfied: parsel>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (1.8.1)
Requirement already satisfied: pyOpenSSL>=21.0.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (23.3.0)
Requirement already satisfied: queuelib>=1.4.2 in /usr/local/lib/python3.10/dist-packages (from scrapy) (1.6.2)
Requirement already satisfied: service-identity>=18.1.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (23.1.0)
Requirement already satisfied: w3lib>=1.17.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (2.1.2)
Requirement already satisfied: zope.interface>=5.1.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (6.1)
Requirement already satisfied: protego>=0.1.15 in /usr/local/lib/python3.10/dist-packages (from scrapy) (0.3.0)
Requirement already satisfied: itemadapter>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from scrapy) (0.8.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from scrapy) (67.7.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from scrapy) (23.2)
Requirement already satisfied: tldextract in /usr/local/lib/python3.10/dist-packages (from scrapy) (5.1.1)
Requirement already satisfied: lxml>=4.4.1 in /usr/local/lib/python3.10/dist-packages (from scrapy) (4.9.3)
Requirement already satisfied: PyDispatcher>=2.0.5 in /usr/local/lib/python3.10/dist-packages (from scrapy) (2.0.7)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-packages (from cryptography>=36.0.0->scrapy) (1.16.0)
Requirement already satisfied: jmespath>=0.9.5 in /usr/local/lib/python3.10/dist-packages (from itemloaders>=1.0.1->scrapy) (1.0.1)
Requirement already satisfied: attrs>=19.1.0 in /usr/local/lib/python3.10/dist-packages (from service-identity>=18.1.0->scrapy) (23.1.0)
Requirement already satisfied: pyasn1 in /usr/local/lib/python3.10/dist-packages (from service-identity>=18.1.0->scrapy) (0.5.0)
Requirement already satisfied: pyasn1-modules in /usr/local/lib/python3.10/dist-packages (from service-identity>=18.1.0->scrapy) (0.3.0)
Requirement already satisfied: constantly>=15.1 in /usr/local/lib/python3.10/dist-packages (from Twisted<23.8.0,>=18.9.0->scrapy) (23.10.4)
Requirement already satisfied: incremental>=21.3.0 in /usr/local/lib/python3.10/dist-packages (from Twisted<23.8.0,>=18.9.0->scrapy) (22.10.0)
Requirement already satisfied: Automat>=0.8.0 in /usr/local/lib/python3.10/dist-packages (from Twisted<23.8.0,>=18.9.0->scrapy) (22.10.0)
Requirement already satisfied: hyperlink>=17.1.1 in /usr/local/lib/python3.10/dist-packages (from Twisted<23.8.0,>=18.9.0->scrapy) (21.0.0)
Requirement already satisfied: typing-extensions>=3.6.5 in /usr/local/lib/python3.10/dist-packages (from Twisted<23.8.0,>=18.9.0->scrapy) (4.5.0)
Requirement already satisfied: idna in /usr/local/lib/python3.10/dist-packages (from tldextract->scrapy) (3.4)
Requirement already satisfied: requests>=2.1.0 in /usr/local/lib/python3.10/dist-packages (from tldextract->scrapy) (2.31.0)
Requirement already satisfied: requests-file>=1.4 in /usr/local/lib/python3.10/dist-packages (from tldextract->scrapy) (1.5.1)
Requirement already satisfied: filelock>=3.0.8 in /usr/local/lib/python3.10/dist-packages (from tldextract->scrapy) (3.13.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from Automat>=0.8.0->Twisted<23.8.0,>=18.9.0->scrapy) (1.16.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.12->cryptography>=36.0.0->scrapy) (2.21)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.1.0->tldextract->scrapy) (3.3.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.1.0->tldextract->scrapy) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.1.0->tldextract->scrapy) (2023.7.22)
```

RESULT

Program executed successfully.

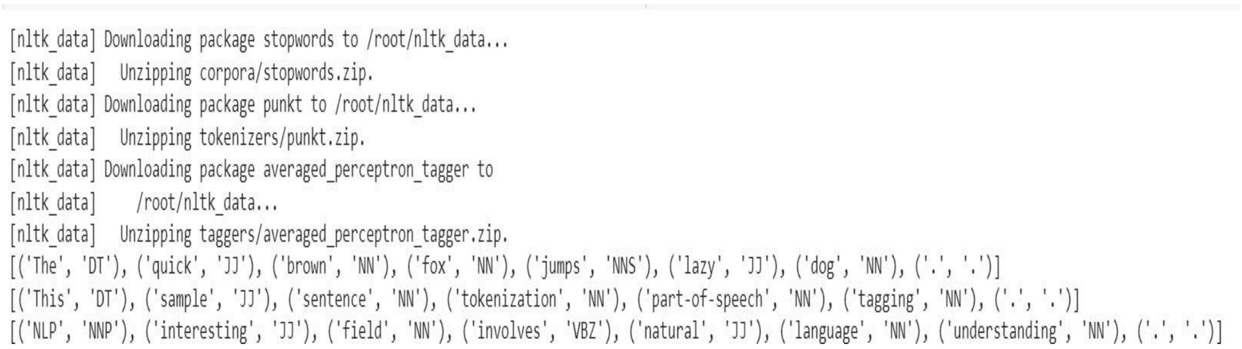
Experiment-10A

Parts of Speech Tagging

AIM: To demonstrate how to preprocess and analyze text data by tokenizing, removing stopwords, and performing part-of-speech tagging.

SOURCE CODE & OUTPUT:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
stop_words = set(stopwords.words('english'))
txt = "The quick brown fox jumps over the lazy dog. " \
    "This is a sample sentence for tokenization and part-of-speech tagging. " \
    "NLP is an interesting field that involves natural language understanding."
tokenized = sent_tokenize(txt)
for i in tokenized:
  wordsList = nltk.word_tokenize(i)
  wordsList = [w for w in wordsList if not w in stop_words]
  tagged = nltk.pos_tag(wordsList)
  print(tagged)
```



```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'NNS'), ('lazy', 'JJ'), ('dog', 'NN'), ('.', '.')]
[('This', 'DT'), ('sample', 'JJ'), ('sentence', 'NN'), ('tokenization', 'NN'), ('part-of-speech', 'NN'), ('tagging', 'NN'), ('.', '.')]
[('NLP', 'NNP'), ('interesting', 'JJ'), ('field', 'NN'), ('involves', 'VBZ'), ('natural', 'JJ'), ('language', 'NN'), ('understanding', 'NN'), ('.', '.')]
```

RESULT

Program executed successfully.

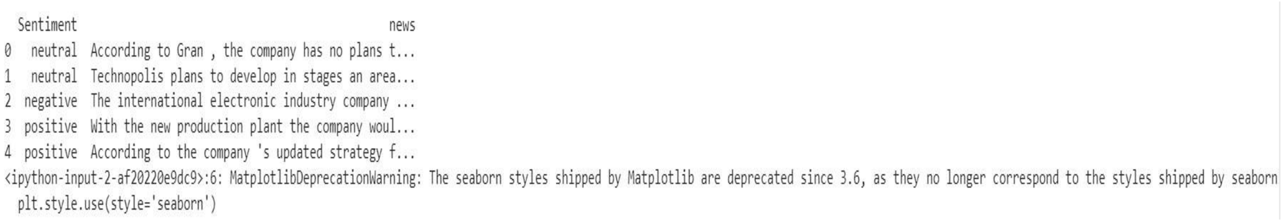
Experiment-10B

N-gram generation

AIM: The program to preprocess sentiment-labeled financial news data, including loading the dataset, splitting it into training and testing sets, removing punctuation, and demonstrating the generation of N-grams for text classification tasks.

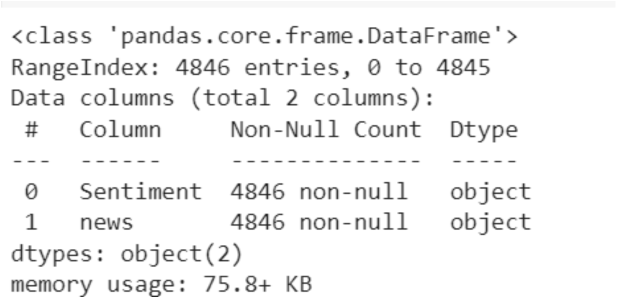
SOURCE CODE & OUTPUT:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use(style='seaborn')
colnames = ['Sentiment', 'news']
df = pd.read_csv('/content/all-data - all-data.csv', encoding="ISO-8859-1",
names=colnames, header=None)
print(df.head())
```



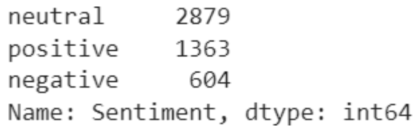
```
  Sentiment                                               news
0   neutral  According to Gran , the company has no plans t...
1   neutral  Technopolis plans to develop in stages an area...
2  negative  The international electronic industry company ...
3  positive  With the new production plant the company woul...
4  positive  According to the company 's updated strategy f...
<ipython-input-2-af20220e9dc9>:6: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn
  plt.style.use(style='seaborn')
```

```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4846 entries, 0 to 4845
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Sentiment  4846 non-null   object
 1   news       4846 non-null   object
dtypes: object(2)
memory usage: 75.8+ KB
```

df['Sentiment'].value_counts()



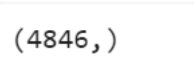
```
neutral     2879
positive    1363
negative     604
Name: Sentiment, dtype: int64
```

y = df['Sentiment'].values

y.shape

x = df['news'].values

x.shape



```
(4846,)
```

from sklearn.model_selection import train_test_split

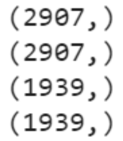
(x_train, x_test, y_train, y_test) = train_test_split(x, y, test_size=0.4)

print(x_train.shape)

print(y_train.shape)

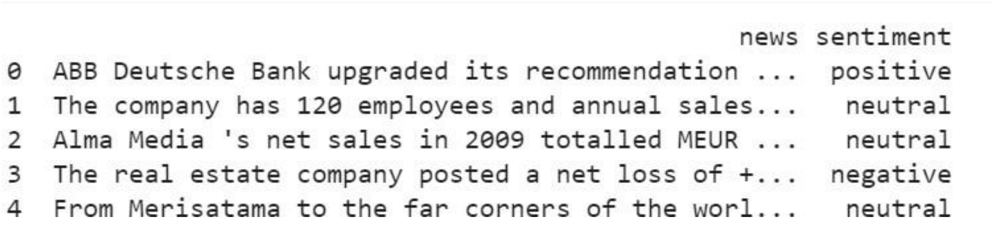
print(x_test.shape)

print(y_test.shape)



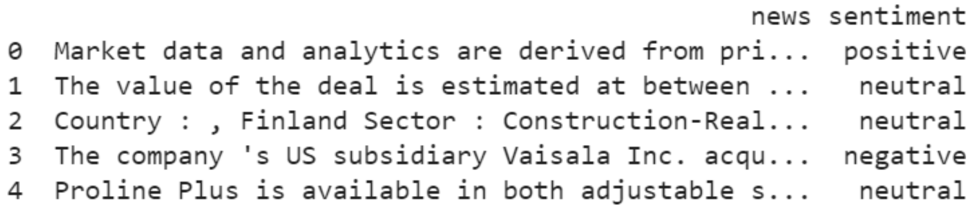
```
(2907,)
(2907,)
(1939,)
(1939,)
```

```
df1 = pd.DataFrame(x_train)
df1 = df1.rename(columns={0: 'news'})
df2 = pd.DataFrame(y_train)
df2 = df2.rename(columns={0: 'sentiment'})
df_train = pd.concat([df1, df2], axis=1)
print(df_train.head())
```



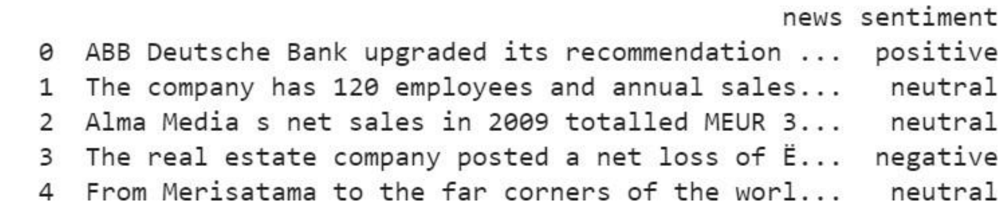
```
                                                news sentiment
0  ABB Deutsche Bank upgraded its recommendation ...  positive
1  The company has 120 employees and annual sales...   neutral
2  Alma Media 's net sales in 2009 totalled MEUR ...   neutral
3  The real estate company posted a net loss of +...  negative
4  From Merisatama to the far corners of the worl...   neutral
```

```
df3 = pd.DataFrame(x_test)
df3 = df3.rename(columns={0: 'news'})
df4 = pd.DataFrame(y_test)
df4 = df2.rename(columns={0: 'sentiment'})
df_test = pd.concat([df3, df4], axis=1)
print(df_test.head())
```



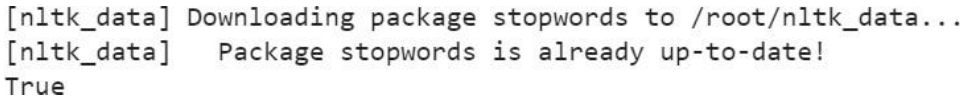
```
                                                news sentiment
0  Market data and analytics are derived from pri...  positive
1  The value of the deal is estimated at between ...   neutral
2  Country : , Finland Sector : Construction-Real...   neutral
3  The company 's US subsidiary Vaisala Inc. acqu...  negative
4  Proline Plus is available in both adjustable s...   neutral
```

```
import string
def remove_punctuation(text):
    if type(text) == float:
        return text
    ans = ""
    for i in text:
        if i not in string.punctuation:
            ans += i
    return ans
df_train['news'] = df_train['news'].apply(lambda x: remove_punctuation(x))
df_test['news'] = df_test['news'].apply(lambda x: remove_punctuation(x))
print(df_train.head())
```



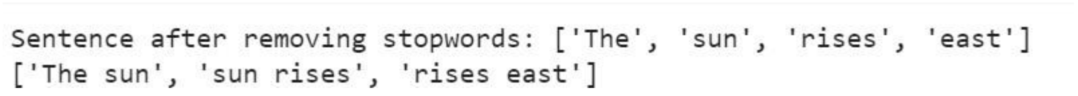
```
                                                news sentiment
0  ABB Deutsche Bank upgraded its recommendation ...  positive
1  The company has 120 employees and annual sales...   neutral
2  Alma Media s net sales in 2009 totalled MEUR 3...   neutral
3  The real estate company posted a net loss of Ë...  negative
4  From Merisatama to the far corners of the worl...   neutral
```

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
```



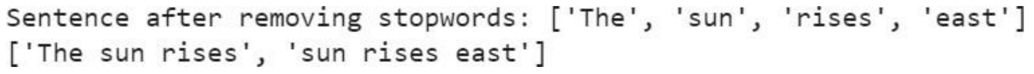
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
def generate_N_grams(text, ngram=1):
    words = [word for word in text.split(" ") if word not in
set(stopwords.words('english'))]
    print("Sentence after removing stopwords:", words)
    temp = zip(*[words[i:] for i in range(0, ngram)])
    ans = [' '.join(ngram) for ngram in temp]
    return ans
print(generate_N_grams("The sun rises in the east", 2))
```



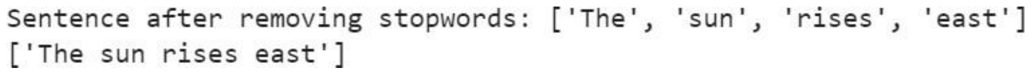
```
Sentence after removing stopwords: ['The', 'sun', 'rises', 'east']
['The sun', 'sun rises', 'rises east']
```

```
print(generate_N_grams("The sun rises in the east", 3))
```



```
Sentence after removing stopwords: ['The', 'sun', 'rises', 'east']
['The sun rises', 'sun rises east']
```

```
print(generate_N_grams("The sun rises in the east", 4))
```



```
Sentence after removing stopwords: ['The', 'sun', 'rises', 'east']
['The sun rises east']
```

RESULT

Program executed successfully.

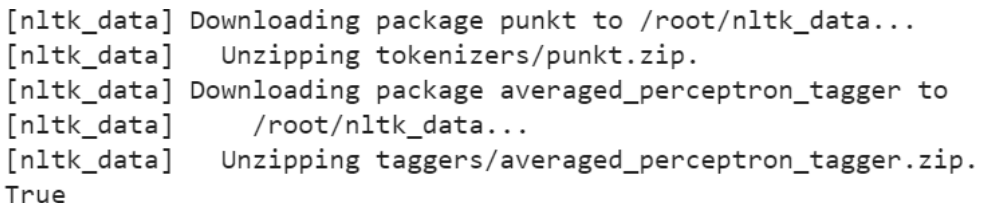
Experiment-10C

Chunking

AIM: The program to read sentences from the 'news' column of the 'all-data.csv' file and perform Noun Phrase (NP) chunking on each sentence using natural language processing techniques.

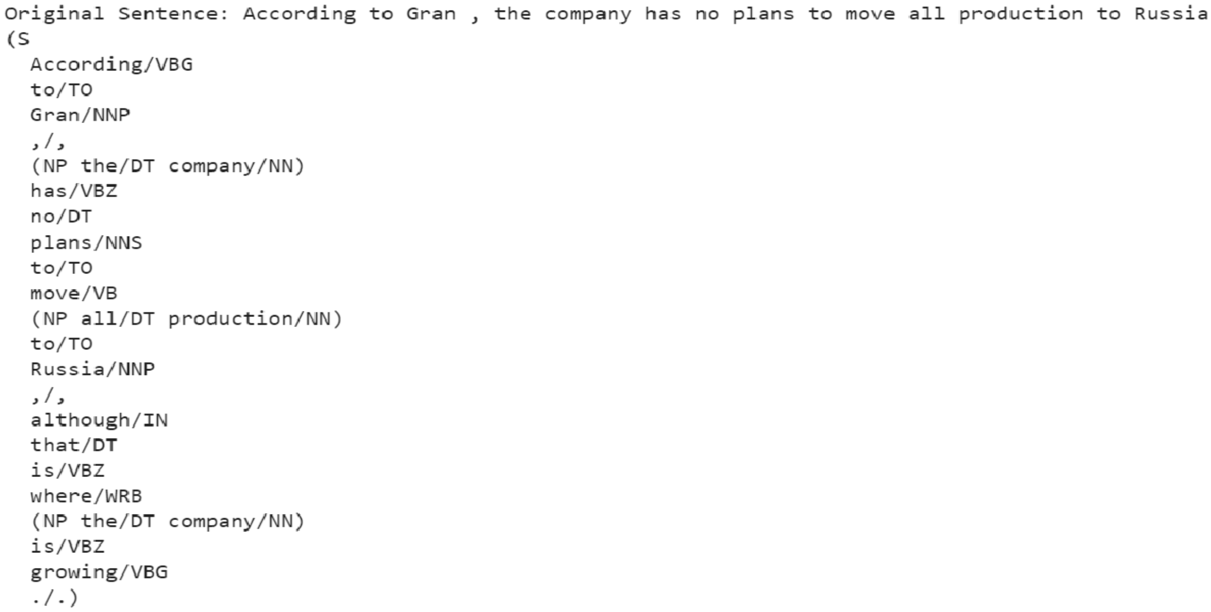
SOURCE CODE & OUTPUT:

```
import pandas as pd
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

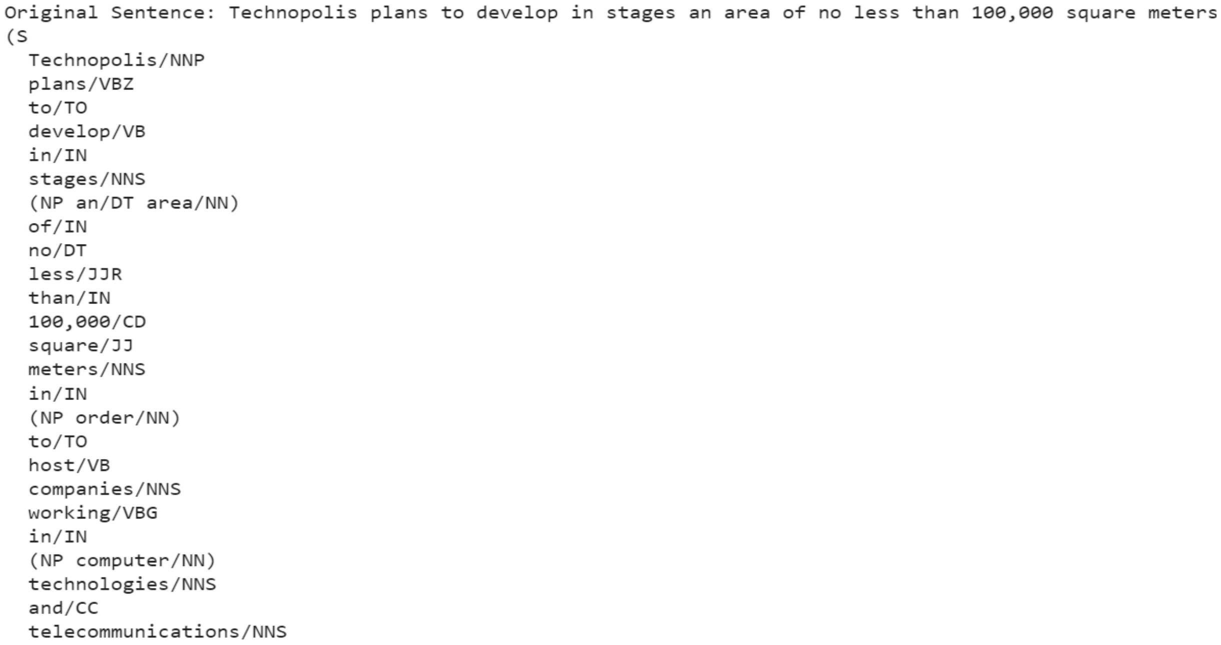


```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

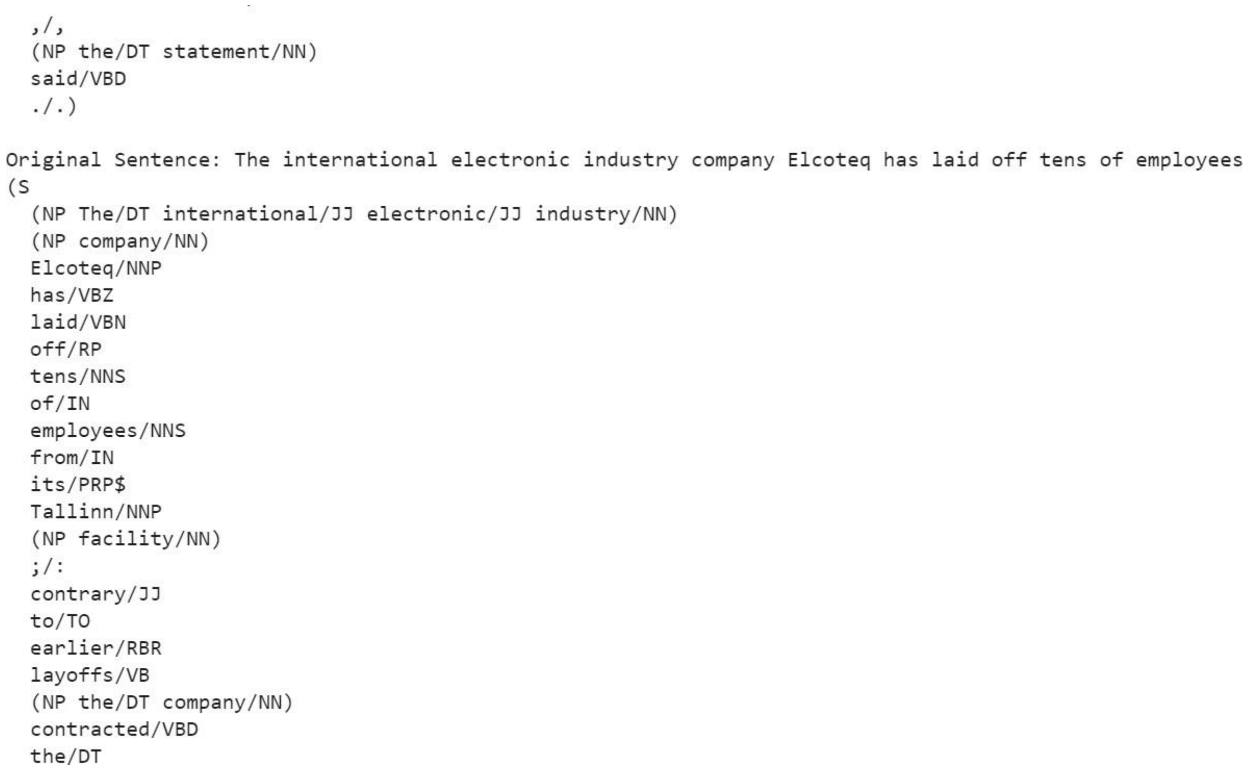
```
colnames = ['Sentiment', 'news']
df = pd.read_csv('/content/all-data - all-data.csv', encoding="ISO-8859-1",
names=colnames, header=None)
sentences_for_chunking = df['news'].head(3)
def perform_chunking(sentence):
    tokens = nltk.word_tokenize(sentence)
    pos_tags = nltk.pos_tag(tokens)
    grammar = "NP: {<DT>?<JJ>*<NN>}"
    chunk_parser = nltk.RegexpParser(grammar)
    chunks = chunk_parser.parse(pos_tags)
    print(chunks)
for sentence in sentences_for_chunking:
    print("\nOriginal Sentence:", sentence)
    perform_chunking(sentence)
```



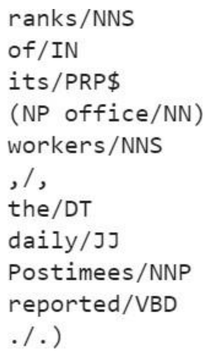
```
Original Sentence: According to Gran , the company has no plans to move all production to Russia
(S
  According/VBG
  to/TO
  Gran/NNP
  ,/,
  (NP the/DT company/NN)
  has/VBZ
  no/DT
  plans/NNS
  to/TO
  move/VB
  (NP all/DT production/NN)
  to/TO
  Russia/NNP
  ,/,
  although/IN
  that/DT
  is/VBZ
  where/WRB
  (NP the/DT company/NN)
  is/VBZ
  growing/VBG
  ./.)
```



```
Original Sentence: Technopolis plans to develop in stages an area of no less than 100,000 square meters
(S
  Technopolis/NNP
  plans/VBZ
  to/TO
  develop/VB
  in/IN
  stages/NNS
  (NP an/DT area/NN)
  of/IN
  no/DT
  less/JJR
  than/IN
  100,000/CD
  square/JJ
  meters/NNS
  in/IN
  (NP order/NN)
  to/TO
  host/VB
  companies/NNS
  working/VBG
  in/IN
  (NP computer/NN)
  technologies/NNS
  and/CC
  telecommunications/NNS
```



```
  ,/,
  (NP the/DT statement/NN)
  said/VBD
  ./.)

Original Sentence: The international electronic industry company Elcoteq has laid off tens of employees
(S
  (NP The/DT international/JJ electronic/JJ industry/NN)
  (NP company/NN)
  Elcoteq/NNP
  has/VBZ
  laid/VBN
  off/RP
  tens/NNS
  of/IN
  employees/NNS
  from/IN
  its/PRP$
  Tallinn/NNP
  (NP facility/NN)
  ;/:
  contrary/JJ
  to/TO
  earlier/RBR
  layoffs/VB
  (NP the/DT company/NN)
  contracted/VBD
  the/DT
```



```
ranks/NNS
of/IN
its/PRP$
(NP office/NN)
workers/NNS
,/,
the/DT
daily/JJ
Postimees/NNP
reported/VBD
./.)
```

RESULT

Program executed successfully.