

LATE BHAUSAHEB HIRAY S.S. TRUST'S INSTITUTE OF COMPUTER APPLICATION ISO 9001-2008 CERTIFIED

S.N. 341, Next to New English School, Govt.Colony, Bandra (East), Mumbai – 400051, Tel: 91-22-26570892/3181

Date: 12/08/2022

CERTIFICATE

This is to certify that Mr. Saurabh Bhausaheb Gawali

Roll No. 2021105 is a student of MCA of 1st year Semester-II has completed successfully full-semester practical/assignments of subject AIML Lab for the academic year 2021-22

Subject In-Charge

External Examiner

Director

INDEX

Sr.no.	Name Of Practical	Date	Sign
1	Implementation of Logic	27-jun-2022	
	programming using PROLOG		
2	Introduction to Python	4-july-2022	
	Programming: Learn the different		
	libraries –		
	1)NumPy		
	2)Pandas		
	3)Matplotlib.		
3	Implementation of	11-july-2022	
	1)Linear Regression		
	2)Logistic regression		
	3)KNN- classification.		
4	Implementation of Bagging	13-july-2022	
	Algorithm: Decision Tree.		
5	Implementation of K-mean	14-july-2022	
	clustering		
6	Implementation of dimensionality	18-july-2022	
	reduction techniques:		
	1)Normalization		
	2)Principal Components Analysis.		
7	Implementation of Classifying	21-july-2022	
	data using Support Vector		
	Machines (SVMs).		

Practical 1

Aim: Study of Logical Programming with Prolog.

Steps: - 1. Create one file in notepad with .pl extension and **file type =All type.**

```
1 studnet(neeta).
2 loves_to_eat(noodles).
3 loves_to_eat(vijeta,noodles).
4 intelligent(suchita).
5 cat(tom).
6 freinds(jack,jill).
```

Go to GNU \rightarrow File \rightarrow Change dir \rightarrow Select the folder where notepad file is saved.

```
SNU Prolog console

File Edit Terminal Prolog Help

GNU Prolog 1.4.5 (64 bits)

Compiled Jul 14 2018, 12:58:46 with cl

By Baniel Diaz

Copyright (C) 1999-2018 Daniel Diaz
```

Code:-

 \rightarrow [p1]. \rightarrow Student(X). Intelligent(Y).

```
GNU Prolog console

File Edit Terminal Prolog Help

GNU Prolog 1.4.5 (64 bits)

Compiled Jul 14 2018, 12:58:46 with cl

By Daniel Diaz

Copyright (C) 1999-2018 Daniel Diaz

| ?- change_directory('D:/AI & ML').

yes

| ?- [p1].

compiling D:/AI & ML/p1.pl for byte code...

D:/AI & ML/p1.pl:1: warning: singleton variables [Sameer] for student/1

D:/AI & ML/p1.pl:3: warning: singleton variables [Krishna] for loves_to_eat/2

D:/AI & ML/p1.pl:4: warning: singleton variables [Ritvik] for intelligent/1

D:/AI & ML/p1.pl compiled, 5 lines read - 826 bytes written, 390 ms

yes

| ?- student(X).

yes

| ?- intelligent(Y)

.

yes

| ?- |
```

2. AND(;) & OR(,) Function

Create new notepad file.

```
→likes(pooja,geeta).
likes(geeta,pooja).
likes(neha,aliya).
freindship(X,Y):-likes(X,Y);likes(Y,X).
```

```
| ?- [p2].
compiling D:/AI & ML/p2.pl for byte code...
D:/AI & ML/p2.pl compiled, 5 lines read - 884 bytes written, 17 ms
?- freindship(X,Y).
X = pooja
Y = geeta ? ;
X = geeta
Y = pooja ? ;
X = neha
Y = aliya ? ;
X = geeta
Y = pooja ? ;
X = pooja
Y = geeta ? ;
X = aliya
Y = neha
(46 ms) yes
| ?-|
```

→ likes(pooja,geeta). likes(geeta,pooja). likes(neha,aliya).

```
| ?- [p2].
compiling D:/AI & ML/p2.pl for byte code...
D:/AI & ML/p2.pl compiled, 5 lines read - 763 bytes written, 16 ms
yes
?- freindship(X,Y).
X = pooja
Y = geeta ? ;
X = geeta
Y = pooja ? ;
no
|| ?- |
→ next_to(mumbai,pune).
  next_to(pune,satara).
  next to(mumbai,nashik).
  travel(A,C) :- next_to(A,B), next_to(B,C).
compiling D:/AI & ML/p2.pl for byte code...
D:/AI & ML/p2.pl compiled, 5 lines read - 738 bytes written, 16 ms
| ?- freindship(X,Y).
X = pooja
Y = geeta ? ;
X = geeta
Y = pooja ? ;
(31 ms) no
```

3. Relationship in prolog: - specify relationship between object and properties of objects. Relationship can also be a rule.

Create new notepad file.

\rightarrow p3.pl

```
female(scarlet).
female(alice).
female(katherine).
female(fiona).
male(bob).
male(sean).
male(chris).
male(dravis).
```

```
parent(bob, alice).
parent(bob, sean).
parent(scarlet,alice).
parent(scarlet, sean).
parent(alice, katherine).
parent(sean, chris).
parent(katherine,fiona).
parent(chris,dravis).
granparent(X,Y) := parent(X,Z), parent(Z,Y).
sister(X,Y) := parent(Z,X), parent(Z,Y), female(X), X \subseteq Y.
brother(X,Y) := parent(Z,X), parent(Z,Y), male(X), female(Y).
uncle(X,Y) := parent(Z,Y), brother(X,Z).
aunt(X,Y) :- parent(Z,Y), sister(X,Z).
daughter(X,Y) :- parent(Y,X), female(X).
son(X,Y) :- parent(Y,X), male(X).
mother(X,Y) := parent(X,Y), female(X).
father(X,Y) :- parent(X,Y), male(X).
```

```
S GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.4.5 (64 bits)
Compiled Jul 14 2018, 12:58:46 with cl
By Daniel Diaz
Copyright (C) 1999-2018 Daniel Diaz
| ?- change_directory('D:/AI & ML').
yes
| ?- [p3].
compiling D:/AI & ML/p3.pl for byte code...
D:/AI & ML/p3.pl compiled, 38 lines read - 4475 bytes written, 354 ms
(31 ms) yes
| ?- parent(X,Y)
X = bob
Y = alice ? ;
X = bob
Y = sean ? ;
X = scarlet
Y = alice ? ;
X = scarlet
Y = sean ? ;
X = alice
Y = katherine ? ;
X = sean
Y = chris ? ;
X = katherine
Y = fiona ?;
X = chris
Y = dravis
```

```
| ?- brother(X,Y).
X = sean
Y = alice ?;
X = sean
Y = alice ? ;
no
| ?- sister(X,Y).
X = alice
Y = sean ? ;
X = alice
Y = sean ? ;
(16 ms) no
?- grandparent(X,Y).
X = bob
Y = katherine ? ;
X = bob
Y = chris ? ;
X = scarlet
Y = katherine ? ;
X = scarlet
Y = chris ? ;
X = alice
Y = fiona ? ;
X = sean
Y = dravis ? ;
```

```
?- father(X,Y).
X = bob
Y = alice ? ;
X = bob
Y = sean ? ;
X = sean
Y = chris ? ;
X = chris
Y = dravis
| ?- son(X,Y).
X = sean
Y = bob ? ;
X = sean
Y = scarlet ? ;
X = chris
Y = sean ? ;
X = dravis
Y = chris
| ?- daughter(X,Y).
X = alice
Y = bob ? ;
X = alice
Y = scarlet ? ;
X = katherine
Y = alice ? ;
X = fiona
Y = katherine ?;
(31 ms) no
```

| ?-| |

Practical NO: 2

Aim: Study of Python Libraries

```
a) NumPy
                                b) Pandas
                                                c)Matplotlib.
NumPy:
import numpy as np
l = ['dog', 'cat', 'horse']
1
Output: ['dog', 'cat', 'horse']
type(1)
Output: list
1.sort()
1
Output: ['cat', 'dog', 'horse']
li = list(range(6))
li
Output: [0, 1, 2, 3, 4, 5]
while li:
  p=li.pop()
  print('p:', p)
  print('li:', li)
Output:
p: 5
li: [0, 1, 2, 3, 4]
p: 4
li: [0, 1, 2, 3]
p: 3
li: [0, 1, 2]
p: 2
li: [0, 1]
p: 1
li: [0]
p: 0
li: []
a = ('Ryan', 33, True)
```

b = 'Takaya', 25, False

```
type(b)
Output: tuple
type(a)
type(b)
Output: tuple
print(a[1])
Output: 33
print(b[0])
Output: Takaya
a = np.array([2,4,6,8])
Output: array([2, 4, 6, 8])
a.dtype
Output: dtype('int32')
a = np.array([2,4,6,8], np.int64)
Output: array([2, 4, 6, 8], dtype=int64)
a = np.array([[2,4,6,8]])
Output: array([[2, 4, 6, 8]])
a[0][3]
Output: 8
a.shape
Output: (1, 4)
listarr = np.array([[1,1,1],[2,2,2],[3,3,3]])
listarr
Output:
array([[1, 1, 1],
    [2, 2, 2],
    [3, 3, 3]]
listarr.shape
Output: (3, 3)
listarr.size
Output: 9
z = np.zeros((2,4))
\mathbf{Z}
Output:
array([[0., 0., 0., 0.],
    [0., 0., 0., 0.]]
z.shape
Output: (2, 4)
y = np.ones((3,4))
Output:
array([[1., 1., 1., 1.],
    [1., 1., 1., 1.],
```

```
[1., 1., 1., 1.]])
y = np.ones((2,3,4))
Output:
array([[[1., 1., 1., 1.],
     [1., 1., 1., 1.],
     [1., 1., 1., 1.]],
    [[1., 1., 1., 1.],
     [1., 1., 1., 1.],
     [1., 1., 1., 1.]])
x = np.arange(10)
Output: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
m = np.linspace(1,5,4)
m
Output: array([1., 2.33333333, 3.66666667, 5.])
m = np.linspace(1,7,3)
Output: Type Markdown and LaTeX: α2
Output: array([1., 4., 7.])
Output:
array([[[1., 1., 1., 1.],
     [1., 1., 1., 1.],
     [1., 1., 1., 1.]],
    [[1., 1., 1., 1.],
     [1., 1., 1., 1.],
     [1., 1., 1., 1.]])
c = np.ones\_like(y)
Output:
array([[[1., 1., 1., 1.],
     [1., 1., 1., 1.],
     [1., 1., 1., 1.]],
    [[1., 1., 1., 1.],
     [1., 1., 1., 1.],
     [1., 1., 1., 1.]])
g = np.ones((2,3,4))
Output:
array([[[1., 1., 1., 1.],
     [1., 1., 1., 1.],
     [1., 1., 1., 1.]], [[1., 1., 1., 1.], [1., 1., 1., 1.],
     [1., 1., 1., 1.]])
g.reshape
Output: <function ndarray.reshape>
```

```
Output:
array([[[1., 1., 1., 1.], [1., 1., 1., 1.],
     [1., 1., 1., 1.], [[1., 1., 1., 1.],
     [1., 1., 1., 1.],
     [1., 1., 1., 1.]]
h = np.arange(50)
h
Output:
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
    17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
    34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])
h.reshape(2,25)
Output:
array([[ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
     16, 17, 18, 19, 20, 21, 22, 23, 24],
    [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
     41, 42, 43, 44, 45, 46, 47, 48, 49]])
h.ravel()
Output:
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
    17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
    34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])
b = np.arange(3,10,2, dtype=np.int32)
b.itemsize
b
Output: 4
b = np.arange(3.4,10,2)
b.itemsize
Output: 8
b.shape
Output: (4,)
b.itemsize
Output: 8
t = np.linspace(3,10,3, dtype=np.int32)
Output: array([ 3. , 4.75, 6.5 , 8.25, 10. ])
t = np.linspace(3,10,5, dtype=np.int32)
Output: array([ 3, 4, 6, 8, 10])
m = np.arange(6)
Output: array([0, 1, 2, 3, 4, 5])
m.reshape(2,3)
Output:
array([[0, 1, 2],
    [3, 4, 5]]
m.reshape(3,2)
Output:
array([[0, 1],[2, 3], [4, 5]])
```

```
Pandas:
import numpy as np
import pandas as pd
dict = { "name":['aa', 'bb', 'cc'],
    "class":['fy','sy','ty'],
    "roll":[11, 22, 33]}
dict
Output: {'name': ['aa', 'bb', 'cc'], 'class': ['fy', 'sy', 'ty'], 'roll': [11, 22, 33]}
df = pd.DataFrame(dict)
df
Output:
name class
              roll
                      11
0
              fy
       aa
1
       bb
                     22
              sy
2
       cc
              ty
                     33
df.to_csv('student.csv')
df.to_csv('index_false_student.csv', index=False)
df.head()
Output:
name class
             roll
0
       aa
              fy
                     11
1
       bb
                     22
              sy
2
       cc
                     33
              ty
df.tail()
Output:
name class
              roll
0
              fy
                     11
       aa
                     22
1
       bb
              sy
2
                     33
       cc
              ty
df.describe()
Output:
roll
count 3.0
mean 22.0
std
       11.0
min
       11.0
25%
      16.5
50%
       22.0
       27.5
75%
       33.0
max
df.head(3)
Output:
       name class
                     roll
0
       aa
              fy
                     11
                     22
1
       bb
              sy
2
                     33
       cc
              ty
df.to_csv('index.csv', index=False)
Output:
```

name class roll

```
0
                     11
       aa
             fy
1
                    22
       bb
             sy
2
      cc
             ty
                    33
df.to_csv('index1.csv', index=False)
demo = pd.read_csv('index2.csv')
demo
Output:
       prod_id
                    name area
       2200 apple andheri
0
1
       3300
             mango parle
2
      4400
             orange santacruz
demo['name']
Output:
   apple
0
1
    mango
   orange
Name: name, dtype: object
demo['name'][1]
Output: 'mango'
demo['prod_id']
Output:
0 2200
1
   3300
2 4400
Name: prod_id, dtype: int64
demo['prod_id'][2]
Output: 4400
demo['prod_id'][2] = 4004
Output: warning
<ipython-input-64-0c3a9eb8bc8c>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
demo['prod_id'][2] = 4004
demo['prod_id']
Output:
0 2200
   3300
1
2 4004
Name: prod_id, dtype: int64
demo.to_csv('new.csv')
demo
Output:
             name area
prod_id
0
      2200
             apple andheri
1
      3300 mango parle
2
      4004 orange santacruz
demo.index = ['one', 'two', 'three']
demo
Output:
      prod_id
                    name area
```

```
2200
             apple andheri
one
             mango parle
      3300
two
three
      4004
             orange Santacruz
s = pd.Series([2,3,4,5,6,7,8,9,10])
Output:
   2
0
   3
1
2
   4
3
   5
4
   6
5
   7
   8
6
7
   9
8
  10
dtype: int64
s1 = pd.Series(np.random.rand(20))
s1
Output:
   0.476242
1
   0.332118
2
   0.265113
3
   0.722535
4
   0.210917
5
   0.204344
6
   0.557794
7
   0.585600
8
   0.775989
9
   0.555856
10 0.669544
11
    0.874442
12 0.534156
13 0.260446
14 0.519634
15
    0.776713
16 0.660476
17
    0.748030
18
   0.814161
19 0.366974
dtype: float64
df1 = pd.DataFrame(np.random.rand(20,10))
df1
Output:
      0
             1
                    2
                          3
                                       5
                                                     7
                                                                  9
0
      0.889829
                    0.217723
                                 0.950464
                                                            0.175260
                                              0.114454
                                                                         0.171785
      0.502882
                   0.431306
                                 0.585802
                                              0.824907
1
      0.815695
                   0.961605
                                 0.734357
                                              0.617062
                                                            0.778672
                                                                         0.737305
      0.224034
                    0.792681
                                 0.043488
                                              0.755798
2
      0.300321
                   0.297326
                                                           0.954124
                                 0.667170
                                              0.810632
                                                                         0.527148
      0.697780
                   0.679426
                                 0.251948
                                              0.124489
3
                                                                         0.602699
      0.648760
                   0.770672
                                 0.254008
                                              0.025945
                                                           0.110265
```

0.498752

0.413338

0.312994

0.293970

4	0.538527	0.630472	0.851454	0.061778	0.659211	0.565140
	0.876626	0.598274	0.997209	0.087594		
5	0.541544	0.934696	0.424254	0.602228	0.491561	0.614428
	0.120711	0.491124	0.204725	0.973860		
6	0.628961	0.302158	0.846598	0.068880	0.285089	0.233620
	0.408571	0.277139	0.119807	0.524263		
7	0.120473	0.407693	0.207758	0.042455	0.203260	0.605364
	0.230598	0.450066	0.450713	0.003687		
8	0.558722	0.927035	0.777533	0.483478	0.847846	0.096667
	0.910407	0.327488	0.254891	0.337679		
9	0.427066	0.629416	0.845941	0.008152	0.927802	0.945599
	0.783255	0.626967	0.922936	0.155402		
10	0.748707	0.909395	0.492470	0.046778	0.203244	0.102367
	0.242721	0.370299	0.525937	0.410644		
11	0.190404	0.602494	0.196155	0.650595	0.986109	0.680599
	0.886406	0.262964	0.956797	0.719145		
12	0.240944	0.520401	0.174845	0.756972	0.198388	0.355310
	0.419668	0.514867	0.761939	0.560055		
13	0.627101	0.535762	0.842373	0.963862	0.816623	0.052924
	0.211294	0.368572	0.167157	0.388588		
14	0.978139	0.237486	0.077492	0.209904	0.650783	0.663827
	0.352613	0.130673	0.536371	0.074908		
15	0.488940	0.336477	0.495782	0.341456	0.425742	0.461244
	0.142852	0.294217	0.499867	0.226806		
16	0.024142	0.726993	0.602587	0.815984	0.753234	0.515214
	0.982483	0.124366	0.452646	0.757576		
17	0.428680	0.481441	0.671396	0.437300	0.565147	0.387528
	0.174145	0.295377	0.683534	0.326617		
18	0.529209	0.236979	0.605650	0.002481	0.898732	0.043005
10	0.464004	0.849748	0.056447	0.424221	0.640006	0.400054
19	0.884170	0.725553	0.001559	0.273916	0.643806	0.102261
	0.280440	0.360105	0.760108	0.674790		
type(d	lf1)					
• •	escribe()					
Outpu	ut:					
_	0 1	2 3	4 5	6 7	8 9	
count	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000
	20.000000	20.000000	20.000000	20.000000		
mean	0.530517	0.569589	0.535992	0.366716	0.578745	0.423202
	0.470512	0.432950	0.477266	0.432250		
std	0.263312	0.252397	0.294265	0.323563	0.296716	0.265350
	0.286197	0.197626	0.300232	0.281850		
min	0.024142	0.217723	0.001559	0.002481	0.110265	0.043005
	0.120711	0.124366	0.043488	0.003687		
25%	0.395380	0.327897	0.242445	0.058028	0.264632	0.154430
	0.228957	0.295087	0.240142	0.208955		
50%	0.540035	0.569128	0.604118	0.307686	0.647295	0.488229
7. 5. 1	0.414119	0.391818	0.476257	0.399616	0.004400	0.607.600
75%	0.673747	0.737913	0.793743	0.625445	0.824429	0.607630
	0.719149	0.535718	0.702677	0.685879	0.006100	0.045500
max	0.978139	0.961605	0.950464	0.963862	0.986109	0.945599
	0.982483	0.849748	0.997209	0.973860		

df1 [0][1] = "abc" df1.head(10)

Outp	out:							
0	1 2	3 4	5 6	7 8	9			
0	0.889829	0.217723	0.950464	0.114454	0.175260	0.171785		
	0.502882	0.431306	0.585802	0.824907				
1	abc 0.96	1605 0.734	1357 0.617	7062 0.778	8672 0.73	7305		
	0.224034	0.792681	0.043488	0.755798				
2	0.300321	0.297326	0.667170	0.810632	0.954124	0.527148		
	0.697780	0.679426	0.251948	0.124489				
3	0.64876	0.770672	0.254008	0.025945	0.110265	0.602699		
	0.498752	0.413338	0.312994	0.293970				
4	0.538527	0.630472	0.851454	0.061778	0.659211	0.565140		
	0.876626	0.598274	0.997209	0.087594				
5	0.541544	0.934696	0.424254	0.602228	0.491561	0.614428		
	0.120711	0.491124	0.204725	0.973860				
6	0.628961	0.302158	0.846598	0.068880	0.285089	0.233620		
	0.408571	0.277139	0.119807	0.524263				
7	0.120473	0.407693	0.207758	0.042455	0.203260	0.605364		
	0.230598	0.450066	0.450713	0.003687				
8	0.558722	0.927035	0.777533	0.483478	0.847846	0.096667		
	0.910407	0.327488	0.254891	0.337679				
9	0.427066	0.629416	0.845941	0.008152	0.927802	0.945599		
	0.783255	0.626967	0.922936	0.155402				
df1.h	ead(4)							
Outp	out:							
0	1 2	3 4	5 6	7 8	9			
0	0.889829	efg 0.950	0.114	1454 0.175	5260 0.17	1785		
	0.502882	0.431306	0.585802	0.824907				
1	abc 0.96	1605 0.734	1357 0.617	7062 0.778	3672 0.73°	7305		
	0.224034	0.792681	0.043488	0.755798				
2	pqr 0.29	7326 0.667	7170 0.810	0.954	1124 0.52	7148		
	0.697780	0.679426	0.251948	0.124489				
3	0.64876	0.770672	0.254008	0.025945	0.110265	0.602699		
	0.498752	0.413338	0.312994	0.293970				
df1[2	df1[2][1]="aaa"							
Outp	out:							
-:	han innut 05 (02440-055464	. 1. Cottina Witt	Conv.Womin				

<ipython-input-95-93449a955d64>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

df1

Out	nut:
Out	pu.

Outp	u.					
0	1 2	3 4	5 6	7 8	9	
0	0.889829	efg 0.95	50464 0.11	4454 0.1	75260 0.17	1785
	0.502882	0.431306	0.585802	0.824907		
1	abc 0.9	061605 aaa	0.617062	0.778672	0.737305	0.224034
	0.792681	0.043488	0.755798			
2	pqr 0.2	297326 0.66	5717 0.81	0632 0.95	54124 0.52	7148
	0.697780	0.679426	0.251948	0.124489		
3	0.64876	0.770672	0.254008	0.025945	0.110265	0.602699
	0.498752	0.413338	0.312994	0.293970		
4	0.538527	0.630472	0.851454	0.061778	0.659211	0.565140
	0.876626	0.598274	0.997209	0.087594		

5	0.541544	0.934696	0.424254	0.602228	0.491561	0.614428
	0.120711	0.491124	0.204725	0.973860		
6	0.628961	0.302158	0.846598	0.068880	0.285089	0.233620
	0.408571	0.277139	0.119807	0.524263		
7	0.120473	0.407693	0.207758	0.042455	0.203260	0.605364
	0.230598	0.450066	0.450713	0.003687		
8	0.558722	0.927035	0.777533	0.483478	0.847846	0.096667
	0.910407	0.327488	0.254891	0.337679		
9	0.427066	0.629416	0.845941	0.008152	0.927802	0.945599
	0.783255	0.626967	0.922936	0.155402		
10	0.748707	0.909395	0.49247	0.046778	0.203244	0.102367
	0.242721	0.370299	0.525937	0.410644		
11	0.190404	0.602494	0.196155	0.650595	0.986109	0.680599
	0.886406	0.262964	0.956797	0.719145		
12	0.240944	0.520401	0.174845	0.756972	0.198388	0.355310
	0.419668	0.514867	0.761939	0.560055		
13	0.627101	0.535762	0.842373	0.963862	0.816623	0.052924
	0.211294	0.368572	0.167157	0.388588		
14	0.978139	0.237486	0.077492	0.209904	0.650783	0.663827
	0.352613	0.130673	0.536371	0.074908		
15	0.48894	0.336477	0.495782	0.341456	0.425742	0.461244
	0.142852	0.294217	0.499867	0.226806		
16	0.024142	0.726993	0.602587	0.815984	0.753234	0.515214
	0.982483	0.124366	0.452646	0.757576		
17	0.42868	0.481441	0.671396	0.437300	0.565147	0.387528
	0.174145	0.295377	0.683534	0.326617		
18	0.529209	0.236979	0.60565	0.002481	0.898732	0.043005
	0.464004	0.849748	0.056447	0.424221		
19	0.88417	0.725553	0.001559	0.273916	0.643806	0.102261
	0.280440	0.360105	0.760108	0.674790		

demo

Output:

prod_id name area one 2200 apple andheri two 3300 mango parle three 4004 grapes santacruz

 $demo['prod_id'][1] = 5005$

<ipython-input-98-547956110199>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

 $demo['prod_id'][1] = 5005$ demo

Output:

prod_id name area
one 2200 apple andheri
two 5005 mango parle
three 4004 grapes santacruz

demo.dtypes

Output:
prod_id int64
name object
area object

dtype: object

df1.dtypes Output: 0 object 1 object 2 object 3 float64 4 float64

- 5 float64
- 6 float64
- 7 float64
- 8 float64 9 float64
- dtype: object

df1

Outp		_					_			
0		2 3	4	5	6	7	8	9		
0	0.88982	\mathcal{C}	0.950			4454		5260	0.17	1785
	0.50288		1306	0.583			4907			
1		0.961605	aaa	0.61		0.77	8672	0.737	305	0.224034
	0.79268		3488	0.75						
2	1 1	0.297326	0.667			0632		4124	0.52°	7148
	0.69778		9426	0.25			4489			
3	0.64876		0672	0.25			5945	0.110	1265	0.602699
	0.49875		3338	0.312			3970			
4	0.53852		0472	0.85			1778	0.659	211	0.565140
	0.87662		8274		7209		7594			
5	0.54154		4696		4254		2228	0.491	561	0.614428
	0.12071		1124		4725		3860			
6	0.62896		2158		6598		8880	0.285	089	0.233620
	0.40857		7139	0.119			4263			
7	0.12047		7693	0.20			2455	0.203	260	0.605364
	0.23059		0066		0713		3687			
8	0.55872		7035	0.77'			3478	0.847	846	0.096667
	0.91040		7488	0.25			7679			
9	0.42706		9416	0.84			8152	0.927	802	0.945599
	0.78325		6967		2936		5402			
10	0.74870		9395	0.492			6778	0.203	244	0.102367
	0.24272		0299	0.52			0644			
11	0.19040		2494	0.19			0595	0.986	5109	0.680599
	0.88640		2964	0.95			9145			
12	0.24094		0401	0.174			6972	0.198	388	0.355310
	0.41966		4867		1939		0055			
13	0.62710		5762		2373		3862	0.816	623	0.052924
	0.21129		8572	0.16'			8588			
14	0.97813		7486	0.07'			9904	0.650	1783	0.663827
	0.35261		0673	0.53			4908			
15	0.48894		6477		5782		1456	0.425	742	0.461244
	0.14285		4217	0.49			6806			
16	0.02414		6993	0.602			5984	0.753	234	0.515214
	0.98248		4366		2646		7576			
17	0.42868		1441		1396		7300	0.565	147	0.387528
	0.17414	5 0.29	5377	0.683	3534	0.32	6617			

18	0.529209	0.236979	0.60565	0.002481	0.898732	0.043005
10	0.464004	0.849748	0.056447	0.424221	0.070732	0.015005
19	0.88417	0.725553	0.001559	0.273916	0.643806	0.102261
	0.280440	0.360105	0.760108	0.674790		**********
df1.t	o_numpy()					
Out						
_	/([[0.889829058	37574625, 'efg'	. 0.9504637009	9415536.		
	0.11445350753				545231497.	
	0.50288248981	,		,		
	0.82490717628	,	,		,	
	['abc', 0.961604		aa', 0.6170616	309044902.		
	0.77867157780				59455848.	
	0.79268078367	,		,		
	['pqr', 0.297326	,		*	<u>1</u> ,	
	0.81063247665				5664397.	
	0.69778007571	,		•	,	
	0.12448932434			,	, , ,	
	[0.64875996319		715150467816	5, 0.254007670	1928634,	
	0.02594477872					
	0.49875172287	,		*	*	
	0.29397027281	,		•	,	
	[0.53852692882		722606997493	3, 0.851454200	1716628,	
	0.06177752739	395137, 0.659	210898121525	2, 0.56514002	20152262,	
	0.87662588422	206889, 0.5982	735952347097	, 0.9972089870	0133152,	
	0.08759388187	7127126],				
	[0.74870695273	367622, 0.9093	947236386258	3, 0.492469776	63430314,	
	0.04677816297	7810866, 0.203	243749991742	266, 0.1023672	526008913,	
	0.24272067170	750022, 0.370	298670831912	9, 0.52593741	57967004,	
	0.41064368110	0825533],				
	[0.4889402089	2946477, 0.33	647710082718	496, 0.4957819	9333797754,	
	0.226806181	7643775],				
	[0.02414249471	1968255, 0.726	5993384950277	1, 0.60258661	34022627,	
	0.81598387251	63027, 0.7532	337808624099	, 0.5152143189	9684777,	
	0.98248327563	390718, 0.1243	662853957240	03, 0.45264646	68408925,	
	0.75757587319					
	[0.42868008325					
	0.43730039897	,		,	,	
	0.17414543667	,	770888681998	3, 0.683533520	5019395,	
	0.32661688397	=:				
	[0.52920856303	,		*	,	
	0.00248139224	141588865 0.8	987318909075	309 0 043004	55769905975	

0.0024813922441588865, 0.8987318909075309, 0.04300455769905975,0.46400399742872456, 0.8497477915770419, 0.05644742404326131,

0.4242205865280846],

[0.8841697571458046, 0.7255526457480526, 0.0015591409819755153,0.2739157678767543, 0.6438064650372433, 0.10226113788566826,

0.2804404425187115, 0.36010547725733033, 0.7601084640428719,0.6747895650580462]], dtype=object)

df1

	0 1	2	3 4	5	6	7	8	9
0	0.889829	efg (0.950464	0.114	4454	0.17	5260	0.171785
	0.502882	0.43130	0.58	35802	0.82	4907		

1	abc	0.961	605	aaa	0.6170	062	0.7786	572	0.7373	805	0.224034
	0.792	681	0.043	488	0.7557	798					
2	pqr	0.2973	326	0.6671	7	0.8106	32	0.9541	24	0.5271	48
	0.697	780	0.679	426	0.2519	948	0.1244	189			
3	0.648	76	0.770	672	0.2540	800	0.0259	945	0.1102	265	0.602699
	0.498	752	0.413	338	0.3129	994	0.2939	970			
4	0.538	527	0.630	472	0.8514	454	0.0617	778	0.6592	211	0.565140
	0.876	626	0.598	274	0.9972	209	0.0875	594			
5	0.541	544	0.934	696	0.4242	254	0.6022	228	0.4915	561	0.614428
	0.120°	711	0.491	124	0.2047	725	0.9738	360			
6	0.628	961	0.302	158	0.8465	598	0.0688	380	0.2850)89	0.233620
	0.408	571	0.277	139	0.1198	307	0.5242	263			
7	0.120	473	0.407	693	0.207	758	0.0424	155	0.2032	260	0.605364
	0.230	598	0.450	066	0.450	713	0.0036	587			
10	0.748	707	0.909	395	0.4924	1 7	0.0467	778	0.2032	244	0.102367
	0.242	721	0.370	299	0.5259	937	0.4106	544			
11	0.190	404	0.602	494	0.196	155	0.6505	595	0.9861	09	0.680599
	0.886	406	0.262	964	0.956	797	0.7191	145			
12	0.240	944	0.520	401	0.1748	345	0.7569	972	0.1983	388	0.355310
	0.419	668	0.514	867	0.7619	939	0.5600)55			
13	0.627	101	0.535	762	0.8423	373	0.9638	362	0.8166	523	0.052924
	0.211	294	0.368	572	0.167	157	0.3885	588			
14	0.978	139	0.237	486	0.0774	192	0.2099	904	0.6507	783	0.663827
	0.352	613	0.130	673	0.5363	371	0.0749	808			
15	0.488	94	0.336	477	0.4957	782	0.3414	156	0.4257	742	0.461244
	0.142	852	0.294	217	0.4998	367	0.2268	306			
16	0.024	142	0.726	993	0.6025	587	0.8159	984	0.7532	234	0.515214
	0.982	483	0.124	366	0.4526	546	0.7575	576			
17	0.428	68	0.481	441	0.6713	396	0.4373	300	0.5651	47	0.387528
	0.174	145	0.295		0.6835		0.3266				
18	0.529		0.236		0.6056		0.0024		0.8987	732	0.043005
	0.464	004	0.849	748	0.0564	147	0.4242	221			
19	0.884	17	0.725	553	0.0013	559	0.2739	916	0.6438	306	0.102261
	0.280	440	0.360	105	0.760	108	0.6747	790			

demo

Output:

prod_id name area one 2200 apple andheri two 5005 mango parle three 4004 grapes santacruz

demo.T

Output:

One two three prod_id 2200 5005 4004 name apple mango grapes area andheriparle santacruz

$df2 = pd.DataFrame(np.random.rand(10,5)) \\ df2$

0	1 2	3 4			
0	0.988782	0.155982	0.163659	0.216378	0.338656
1	0.922171	0.810851	0.249822	0.283435	0.181059

2	0.069235	0.844811	0.165427	0.086819	0.301486
3	0.789741	0.358560	0.738854	0.373372	0.934196
4	0.405396	0.146483	0.516349	0.259770	0.846987
5	0.929204	0.212274	0.604740	0.422453	0.722843
6	0.247970	0.452907	0.853457	0.639186	0.590882
7	0.672903	0.397623	0.773096	0.071042	0.135975
8	0.139015	0.843306	0.936715	0.941274	0.551718
9	0.052673	0.486642	0.234463	0.257344	0.981282

df2.sort_index(axis=1, ascending=False)

Output:

4	3 2	1 0			
0	0.338656	0.216378	0.163659	0.155982	0.988782
1	0.181059	0.283435	0.249822	0.810851	0.922171
2	0.301486	0.086819	0.165427	0.844811	0.069235
3	0.934196	0.373372	0.738854	0.358560	0.789741
4	0.846987	0.259770	0.516349	0.146483	0.405396
5	0.722843	0.422453	0.604740	0.212274	0.929204
6	0.590882	0.639186	0.853457	0.452907	0.247970
7	0.135975	0.071042	0.773096	0.397623	0.672903
8	0.551718	0.941274	0.936715	0.843306	0.139015
9	0.981282	0.257344	0.234463	0.486642	0.052673

demo

Output:

prod_id name area one 2200 apple andheri two 5005 mango parle three 4004 grapes santacruz

p = demo.sort_values('name')

p

Output:

prod_id name area one 2200 apple andheri three 4004 grapes santacruz two 5005 mango parle

q = demo.groupby('prod_id')

q

Output:

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001EE05FE2A30>

import numpy as np

from matplotlib import pyplot as plt

demo.hist('prod_id', bins=20)

ax = plt.gca()

ax.set_yscale('log')

Output: (graph)

demo.info()

Output:

<class 'pandas.core.frame.DataFrame'>

Index: 3 entries, one to three
Data columns (total 3 columns):
Column Non-Null Count Dtype

-- ----- -----

0 prod_id 3 non-null int64 1 name 3 non-null object 2 area 3 non-null object dtypes: int64(1), object(2) memory usage: 204.0+ bytes

df1

Output:

Outp	uı.					
0	1 2	3 4	5 6	7 8	9	
0	0.889829	efg 0.950	464 0.11	4454 0.17	5260 0.17	1785
	0.502882	0.431306	0.585802	0.824907		
1	abc 0.96	1605 aaa	0.617062	0.778672	0.737305	0.224034
	0.792681	0.043488	0.755798			
2	pqr 0.29'			0.954	4124 0.52	7148
	0.697780	0.679426	0.251948	0.124489		
3	0.64876	0.770672	0.254008	0.025945	0.110265	0.602699
	0.498752	0.413338	0.312994	0.293970		
4	0.538527	0.630472	0.851454	0.061778	0.659211	0.565140
	0.876626	0.598274	0.997209	0.087594		
5	0.541544	0.934696	0.424254	0.602228	0.491561	0.614428
	0.120711	0.491124	0.204725	0.973860		
6	0.628961	0.302158	0.846598	0.068880	0.285089	0.233620
	0.408571	0.277139	0.119807	0.524263		
7	0.120473	0.407693	0.207758	0.042455	0.203260	0.605364
	0.230598	0.450066	0.450713	0.003687		
8	0.558722	0.927035	0.777533	0.483478	0.847846	0.096667
	0.910407	0.327488	0.254891	0.337679		
9	0.427066	0.629416	0.845941	0.008152	0.927802	0.945599
	0.783255	0.626967	0.922936	0.155402		
10	0.748707	0.909395	0.49247	0.046778	0.203244	0.102367
	0.242721	0.370299	0.525937	0.410644		
11	0.190404	0.602494	0.196155	0.650595	0.986109	0.680599
	0.886406	0.262964	0.956797	0.719145		
12	0.240944	0.520401	0.174845	0.756972	0.198388	0.355310
	0.419668	0.514867	0.761939	0.560055		
13	0.627101	0.535762	0.842373	0.963862	0.816623	0.052924
	0.211294	0.368572	0.167157	0.388588		
14	0.978139	0.237486	0.077492	0.209904	0.650783	0.663827
	0.352613	0.130673	0.536371	0.074908		
15	0.48894	0.336477	0.495782	0.341456	0.425742	0.461244
	0.142852	0.294217	0.499867	0.226806		
16	0.024142	0.726993	0.602587	0.815984	0.753234	0.515214
	0.982483	0.124366	0.452646	0.757576		
17	0.42868	0.481441	0.671396	0.437300	0.565147	0.387528
	0.174145	0.295377	0.683534	0.326617		
18	0.529209	0.236979	0.60565	0.002481	0.898732	0.043005
4.5	0.464004	0.849748	0.056447	0.424221	0.4.500	0.4055
19	0.88417	0.725553	0.001559	0.273916	0.643806	0.102261
	0.280440	0.360105	0.760108	0.674790		

Output: 0 1 2 3 4 5 6 7 8 9 0.502882 0.502862 0.502862 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.502882 0.5028 0.60228 0.6028	ull												
0 990 efg 0.431306 0.585802 0.824907 0.175260 0.171785 0.502882 1 abc 0.961605 aaa (617062 0.778672 0.737305 0.224034 1 abc 0.961605 aaa (617062 0.778672 0.737305 0.224034 2 pqr 0.297326 0.66717 0.810632 0.954124 0.527148 3 0.64876 0.770672 0.254008 0.025945 0.110265 0.602699 4 0.538527 0.630472 0.851454 0.061778 0.659211 0.565140 0.876626 0.598274 0.997209 0.887594 0.60228 0.491561 0.614428 1 0.120711 0.491124 0.204725 0.973860 0.491561 0.614428 0.120473 0.302158 0.846598 0.068880 0.285089 0.233620 0.120473 0.407693 0.207758 0.042455 0.203260 0.65364 0.230598 0.450066 0.450713 0.003687 0.247066 0.62941 0.084599 <t< th=""><th>_</th><th></th><th>_</th><th>_</th><th></th><th>_</th><th>_</th><th>_</th><th></th><th>_</th><th></th><th></th></t<>	_		_	_		_	_	_		_			
1 abc 0.961605 aaa 0.617062 aaa 0.617062 0.778672 0.737305 0.224034 0.792681 0.043488 0.755798 0.755798 2 pqr 0.297326 0.66717 0.810632 0.954124 0.527148 3 0.64876 0.770672 0.254008 0.025945 0.110265 0.602699 0.498752 0.413338 0.312994 0.293970 0.498752 0.43338 0.312994 0.293970 4 0.538527 0.630472 0.851454 0.061778 0.655211 0.565140 0.876626 0.598274 0.997209 0.087594 0.10711 0.491124 0.204725 0.973860 0.10711 0.491124 0.204725 0.973860 0.285089 0.233620 0.408571 0.277139 0.119807 0.524263 0.408571 0.277139 0.119807 0.524263 0.0408571 0.277139 0.119807 0.524263 0.0408571 0.277139 0.119807 0.524263 0.0408571 0.277139 0.119807 0.524263 0.040457 0.230598 0.450066 0.450713 0.003687 0.230598 0.450066 0.450713 0.003687 0.230598 0.450066 0.450713 0.003687 0.910407 0.327488 0.254891 0.337679 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 0.783255 0.626967 0.922936 0.155402 0.733255 0.626967 0.922936 0.155402 0.73221 0.370299 0.525937 0.410644 0.008152 0.927802 0.945599 0.783255 0.626967 0.922936 0.155402 0.73221 0.370299 0.525937 0.410644 0.008152 0.92808 0.386406 0.262964 0.956797 0.719145 0.242721 0.370299 0.525937 0.410644 0.008152 0.98609 0.680599 0.886406 0.262964 0.956797 0.719145 0.242721 0.370299 0.525937 0.410644 0.19688 0.514867 0.761939 0.550055 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.240944 0.368572 0.167157 0.388588 0.355310 0.419688 0.514867 0.761939 0.550055 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.240944 0.368572 0.167157 0.388588 0.050055 0.982483 0.124366 0.452646 0.757576 0.48894 0.336477 0.499887 0.226806 0.4242 0.726993 0.602587 0.815984 0.753234 0.515214 0.988417 0.725533 0.001559 0.273916 0.643806 0.102261 0.280440 0.360105 0.760108 0.674790 0.500481 0.898732 0.043005 0.760108 0.674790 0.28044		_		_									
abc	0		_					0.1752	260	0.17178	85	0.502882	
Post	1			05	aaa	0.6170	062	0.7786	572	0.73730	05	0.224034	
0.697780 0.679426 0.251948 0.124489 3 0.64876 0.770672 0.254008 0.025945 0.110265 0.602699 0.498752 0.413338 0.312994 0.293970 0.602699 0.602699 4 0.538527 0.630472 0.851454 0.061778 0.655211 0.565140 0.876626 0.598274 0.997209 0.087594 0.991561 0.614428 0.120711 0.491124 0.204725 0.973860 0.88680 0.285089 0.233620 0.408571 0.302158 0.846598 0.068880 0.285089 0.233620 0.408571 0.302158 0.846598 0.06880 0.285089 0.233620 0.120473 0.407693 0.207758 0.042455 0.203260 0.605364 0.230598 0.450066 0.450713 0.003687 0.847846 0.84947 0.910407 0.327488 0.254891 0.337679 0.9247802 0.948599 0.7832555 0.626967 0.922936 0.155402 </th <th>2</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th>22</th> <th>0.0541</th> <th>2.4</th> <th>0.5071</th> <th>40</th>	2							22	0.0541	2.4	0.5071	40	
3 0.64876 0.498752 0.770672 0.413338 0.312994 0.293970 0.293970 0.293970 0.659211 0.565140 0.602699 0.6038227 0.602178 0.997209 0.087594 0.087594 0.659211 0.565140 0.565140 0.672866 5 0.541544 0.120711 0.491124 0.204725 0.602228 0.973860 0.491561 0.602228 0.491561 0.614428 0.614428 0.408571 0.77139 0.277139 0.119807 0.230598 0.45066 0.450713 0.230598 0.450666 0.450713 0.230598 0.450666 0.450713 0.230598 0.450666 0.450713 0.230598 0.450666 0.450713 0.033687 0.847846 0.847846 0.096667 0.910407 0.327488 0.254891 0.254891 0.783255 0.629416 0.845941 0.008152 0.927802 0.927802 0.945599 0.783255 0.626967 0.922936 0.155402 0.15402 0.203244 0.102367 0.242721 0.370299 0.525937 0.410644 0.203244 0.106447 0.102367 0.242721 0.370299 0.525937 0.886406 0.602494 0.956797 0.719145 0.1074888 0.256407 0.149668 0.514867 0.761939 0.560055 0.986109 0.680599 0.886406 0.602494 0.956797 0.719145 0.198388 0.355310 0.352613 0.130673 0.352613 0.167157 0.49867 0.142852 0.167157 0.388588 0.356712 0.42868 0.452742 0.461244 0.978139 0.237486 0.142852 0.4507490 0.226806 0.461244 0.726993 0.605055 0.650783 0.663371 0.789749 0.66857 0.75	2									24	0.5271	48	
0.498752 0.413338 0.312994 0.293970 4 0.538527 0.630472 0.851454 0.061778 0.659211 0.565140 0.876626 0.598274 0.997209 0.087594 0.602228 0.491561 0.614428 5 0.541544 0.934696 0.424254 0.602228 0.491561 0.614428 6 0.628961 0.302158 0.846598 0.068880 0.285089 0.233620 6 0.628961 0.302158 0.846598 0.068880 0.285089 0.233620 7 0.120473 0.407693 0.207758 0.042455 0.203260 0.605364 0.230598 0.450066 0.450713 0.003687 0.847846 0.096667 0.910407 0.327488 0.254891 0.337679 0.9247802 0.945599 0.783255 0.626967 0.922936 0.155402 0.927802 0.945599 10 0.748707 0.909395 0.49247 0.046778 0.203244 0.102367 11	2									0.1100	- - -	0.602600	
4 0.538527 0.630472 0.851454 0.061778 0.659211 0.565140 0.876626 0.598274 0.997209 0.087594 0.42154 0.602228 0.491561 0.614428 5 0.541544 0.934696 0.424254 0.602228 0.491561 0.614428 6 0.628961 0.302158 0.846598 0.068880 0.285089 0.233620 7 0.120473 0.407693 0.207758 0.042455 0.203260 0.605364 8 0.230598 0.450066 0.450713 0.003687 0.203260 0.605364 8 0.558722 0.927035 0.7777533 0.483478 0.847846 0.096667 9 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 0.783255 0.626967 0.922936 0.155402 0.046778 0.203244 0.102367 10 0.748707 0.90395 0.49247 0.046778 0.203244 0.102367 11 0.194040 <t< th=""><th>3</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>0.1102</th><th>65</th><th>0.602699</th></t<>	3									0.1102	65	0.602699	
0.876626 0.598274 0.997209 0.087594 5 0.541544 0.934696 0.424254 0.602228 0.491561 0.614428 0.120711 0.491124 0.204725 0.973860 0.628961 0.302158 0.846598 0.068880 0.285089 0.233620 6 0.628961 0.302158 0.846598 0.068880 0.285089 0.233620 7 0.120473 0.407693 0.207758 0.042455 0.203260 0.605364 8 0.558722 0.927035 0.777533 0.483478 0.847846 0.096667 9 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 10 0.748707 0.909395 0.49247 0.046778 0.203244 0.102367 11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 12 0.24721 0.370299 0.525937 0.410644 0.196155 0.650595 0.986109 0.680599 12													
5 0.541544 0.934696 0.424254 0.602228 0.491561 0.614428 0.120711 0.491124 0.204725 0.973860 0.285089 0.233620 6 0.628961 0.302158 0.846598 0.068880 0.285089 0.233620 7 0.120473 0.407693 0.207758 0.042455 0.203260 0.605364 8 0.558722 0.927035 0.777533 0.483478 0.847846 0.096667 9 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 0.783255 0.626967 0.922936 0.155402 0.046778 0.203244 0.102367 10 0.748707 0.909395 0.49247 0.046778 0.203244 0.102367 11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.224721 0.368572 0.167157 0.388588 0.355310 0.353762 0.842373 0.	4									0.6592	11	0.565140	
0.120711 0.491124 0.204725 0.973860 6 0.628961 0.302158 0.846598 0.068880 0.285089 0.233620 7 0.120473 0.407693 0.207758 0.042455 0.203260 0.605364 8 0.230598 0.450066 0.450713 0.003687 0.847846 0.096667 9 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 9 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 9 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 0.783255 0.626967 0.922936 0.155402 0.92770 0.24771 0.370299 0.525937 0.410644 0.102367 11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.204904 0.520401 0.174845 0.756972 0.198388 0.355310 <t< th=""><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></t<>													
6 0.628961 0.302158 0.846598 0.068880 0.285089 0.233620 7 0.120473 0.407693 0.207758 0.042455 0.203260 0.605364 8 0.2530598 0.450066 0.450713 0.003687 0.847846 0.096667 8 0.558722 0.927035 0.777533 0.483478 0.847846 0.096667 0.910407 0.327488 0.254891 0.337679 0.927802 0.945599 9 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 10 0.748707 0.909395 0.49247 0.046778 0.203244 0.102367 11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 12 0.240721 0.37372099 0.525937 0.410444 0.520401 0.174845 0.756972 0.198388 0.355310 12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 13 <t< th=""><th>5</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>0.4915</th><th>61</th><th>0.614428</th></t<>	5									0.4915	61	0.614428	
0.408571 0.277139 0.119807 0.524263 7 0.120473 0.407693 0.207758 0.042455 0.203260 0.605364 0.230598 0.450066 0.450713 0.003687 0.847846 0.096667 8 0.558722 0.927035 0.7777533 0.483478 0.847846 0.096667 9 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 10 0.748707 0.909395 0.49247 0.046778 0.203244 0.102367 11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.198388 0.355310 12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 13 0.627101 0.535762 0.842373 0.963862 0.816623 0.052924 0.211294 0.368572 0.167157 0.388588 0.352613 0.130673 0.536371 0.0749	_												
7 0.120473 0.407693 0.207758 0.042455 0.203260 0.605364 8 0.230598 0.450066 0.450713 0.003687 0.847846 0.096667 8 0.558722 0.927035 0.777533 0.483478 0.847846 0.096667 0.910407 0.327488 0.254891 0.337679 0.927802 0.945599 0.783255 0.626967 0.922936 0.155402 0.203244 0.102367 10 0.748707 0.909395 0.49247 0.046778 0.203244 0.102367 11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.198388 0.355310 12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 13 0.627101 0.535762 0.842373 0.963862 0.816623 0.052924 14 0.978139 0.2337486 0.077492 0.209904 0.650783 <th>6</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th>0.2850</th> <th>89</th> <th>0.233620</th>	6									0.2850	89	0.233620	
0.230598 0.450066 0.450713 0.003687 8 0.558722 0.927035 0.777533 0.483478 0.847846 0.096667 0.910407 0.327488 0.254891 0.337679 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 0.783255 0.626967 0.922936 0.155402 0.102367 0.242721 0.370299 0.525937 0.410644 0.102367 0.242721 0.370299 0.525937 0.410644 11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.2049044 0.520401 0.174845 0.756972 0.198388 0.355310 12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 13 0.627101 0.535762 0.842373 0.963862 0.816623 0.052924 14 0.978139 0.237486 0.077492 0.209904 0.650783 0.663827 15<											_		
8 0.558722 0.927035 0.777533 0.483478 0.847846 0.096667 9 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 10 0.748707 0.90395 0.49247 0.046778 0.203244 0.102367 10 0.748707 0.90395 0.49247 0.046778 0.203244 0.102367 11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.198388 0.355310 12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 13 0.627101 0.535762 0.842373 0.963862 0.816623 0.052924 0.211294 0.368572 0.167157 0.388588 0.352613 0.130673 0.536371 0.074908 15 0.48894 0.336477 0.495782 0.341456 0.425742 0.461244 0.142852 0.294217 0.499867	7									0.2032	60	0.605364	
0.910407 0.327488 0.254891 0.337679 9 0.427066 0.629416 0.845941 0.008152 0.927802 0.945599 0.783255 0.626967 0.922936 0.155402 0.203244 0.102367 10 0.748707 0.909395 0.49247 0.046778 0.203244 0.102367 0.242721 0.370299 0.525937 0.410644 0.1910644 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.198388 0.355310 12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 13 0.627101 0.535762 0.842373 0.963862 0.816623 0.052924 14 0.978139 0.237486 0.077492 0.209904 0.650783 0.663827 0.352613 0.130673 0.536371 0.074908 0.425742 0.461244 15 0.48894 0.336477 0.495782 <th< th=""><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></th<>													
9	8									0.8478	46	0.096667	
0.783255 0.626967 0.922936 0.155402 10 0.748707 0.909395 0.49247 0.046778 0.203244 0.102367 0.242721 0.370299 0.525937 0.410644 11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.10236 0.198388 0.355310 12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 0.419668 0.514867 0.761939 0.560055 0.198388 0.355310 13 0.627101 0.538572 0.167157 0.388588 0.816623 0.052924 0.211294 0.368572 0.167157 0.388588 0.650783 0.663827 14 0.978139 0.237486 0.077492 0.209904 0.650783 0.663827 0.345251 0.142852 0.294217 0.499867 0.226806 0.425742 0.461244 0.142852 0.294217													
10	9									0.9278	02	0.945599	
0.242721 0.370299 0.525937 0.410644 11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 0.886406 0.262964 0.956797 0.719145 0.756972 0.198388 0.355310 12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 0.419668 0.514867 0.761939 0.560055 0.352613 0.355762 0.842373 0.963862 0.816623 0.052924 0.211294 0.368572 0.167157 0.388588 0.3552613 0.130673 0.536371 0.074908 0.650783 0.663827 14 0.978139 0.237486 0.077492 0.209904 0.650783 0.663827 0.352613 0.130673 0.536371 0.074908 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.425742 0.461244 16 0.024142 0.726993 0.602587 0.815984 0.753234 0.515214 17												0.4000.5	
11 0.190404 0.602494 0.196155 0.650595 0.986109 0.680599 12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 13 0.627101 0.535762 0.842373 0.963862 0.816623 0.052924 0.211294 0.368572 0.167157 0.388588 0.4978139 0.237486 0.077492 0.209904 0.650783 0.663827 15 0.48894 0.336477 0.495782 0.341456 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.452646 0.757576 16 0.024142 0.726993 0.602587 0.815984 0.753234 0.515214 0.982483 0.124366 0.452646 0.757576 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 0.174145 0.295377 0.683534 0.326617 0.898732 0.043005 19 0.88417 0.725553 0.001559 0.273916 0.643806	10									0.2032	44	0.102367	
0.886406 0.262964 0.956797 0.719145 12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 13 0.627101 0.535762 0.842373 0.963862 0.816623 0.052924 0.211294 0.368572 0.167157 0.388588 0.650783 0.663827 14 0.978139 0.237486 0.077492 0.209904 0.650783 0.663827 0.352613 0.130673 0.536371 0.074908 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.425742 0.461244 0.142852 0.294217 0.499867 0.26806 0.757576 0.753234 0.515214 16 0.024142 0.726993 0.602587 0.815984 0.753234 0.515214 17 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 19										0.0044	0.0	0.400.400	
12 0.240944 0.520401 0.174845 0.756972 0.198388 0.355310 0.419668 0.514867 0.761939 0.560055 0.816623 0.052924 13 0.627101 0.535762 0.842373 0.963862 0.816623 0.052924 0.211294 0.368572 0.167157 0.388588 0.077492 0.209904 0.650783 0.663827 0.352613 0.130673 0.536371 0.074908 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.757576 0.726993 0.602587 0.815984 0.753234 0.515214 0.982483 0.124366 0.452646 0.757576 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 17 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 19 0.88417 0.725553 0.001559 <th>11</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th>0.9861</th> <th>09</th> <th>0.680599</th>	11									0.9861	09	0.680599	
0.419668 0.514867 0.761939 0.560055 13 0.627101 0.535762 0.842373 0.963862 0.816623 0.052924 0.211294 0.368572 0.167157 0.388588 0.077492 0.209904 0.650783 0.663827 14 0.978139 0.237486 0.077492 0.209904 0.650783 0.663827 0.352613 0.130673 0.536371 0.074908 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.753234 0.515214 0.982483 0.124366 0.452646 0.757576 0.387528 17 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 0.174145 0.295377 0.683534 0.326617 0.898732 0.043005 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 0 0.4464004	10									0.4000	0.0	0.055010	
13 0.627101 0.535762 0.842373 0.963862 0.816623 0.052924 14 0.978139 0.237486 0.077492 0.209904 0.650783 0.663827 15 0.48894 0.336477 0.495782 0.341456 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.753234 0.515214 16 0.024142 0.726993 0.602587 0.815984 0.753234 0.515214 0.982483 0.124366 0.452646 0.757576 0.387528 17 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 0.174145 0.295377 0.683534 0.326617 0.898732 0.043005 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 19 0.88417 0.725553 0.001559 0.273916 0.643806 0.102261 df2 Output: 0 1 2 3 4 0 0.674790 0.643806 0.102261 <td co<="" th=""><th>12</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>0.1983</th><th>88</th><th>0.355310</th></td>	<th>12</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th>0.1983</th> <th>88</th> <th>0.355310</th>	12									0.1983	88	0.355310
0.211294 0.368572 0.167157 0.388588 14 0.978139 0.237486 0.077492 0.209904 0.650783 0.663827 0.352613 0.130673 0.536371 0.074908 0.425742 0.461244 15 0.48894 0.336477 0.495782 0.341456 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.024142 0.726993 0.602587 0.815984 0.753234 0.515214 0.982483 0.124366 0.452646 0.757576 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 0.174145 0.295377 0.683534 0.326617 0.898732 0.043005 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 0.464004 0.849748 0.056447 0.424221 0.643806 0.102261 0tf2 Output: 0 1 2 3 4 0.069235 0.844811 0.163659 </th <th>10</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th>0.01.66</th> <th></th> <th>0.050004</th>	10									0.01.66		0.050004	
14 0.978139 0.237486 0.077492 0.209904 0.650783 0.663827 0.352613 0.130673 0.536371 0.074908 0.425742 0.461244 15 0.48894 0.336477 0.495782 0.341456 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.024142 0.726993 0.602587 0.815984 0.753234 0.515214 0.982483 0.124366 0.452646 0.757576 0.565147 0.387528 0.174145 0.295377 0.683534 0.326617 0.898732 0.043005 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 0.464004 0.849748 0.056447 0.424221 0.643806 0.102261 19 0.88417 0.725553 0.001559 0.273916 0.643806 0.102261 df2 Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 0.181059 0.2069235 0.844811 0.165427	13									0.8166	23	0.052924	
0.352613 0.130673 0.536371 0.074908 15 0.48894 0.336477 0.495782 0.341456 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.024142 0.726993 0.602587 0.815984 0.753234 0.515214 0.982483 0.124366 0.452646 0.757576 0.757576 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 0.174145 0.295377 0.683534 0.326617 0.898732 0.043005 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 0.464004 0.849748 0.056447 0.424221 0.643806 0.102261 19 0.88417 0.725553 0.001559 0.273916 0.643806 0.102261 df2 Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.0692	1.4									0.6507	0.2	0.662027	
15 0.48894 0.336477 0.495782 0.341456 0.425742 0.461244 0.142852 0.294217 0.499867 0.226806 0.753234 0.515214 16 0.024142 0.726993 0.602587 0.815984 0.753234 0.515214 0.982483 0.124366 0.452646 0.757576 0.733234 0.515214 17 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 0.174145 0.295377 0.683534 0.326617 0.898732 0.043005 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 0.464004 0.849748 0.056447 0.424221 0.643806 0.102261 19 0.88417 0.725553 0.001559 0.273916 0.643806 0.102261 df2 Output: 0 1 2 3 4 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 0.181059 0.2069235 0.844811 0.165427 0.086819 0.301486 0.789	14									0.6507	83	0.663827	
0.142852 0.294217 0.499867 0.226806 16 0.024142 0.726993 0.602587 0.815984 0.753234 0.515214 0.982483 0.124366 0.452646 0.757576 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 0.174145 0.295377 0.683534 0.326617 0.898732 0.043005 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 0.464004 0.849748 0.056447 0.424221 0.643806 0.102261 19 0.88417 0.725553 0.001559 0.273916 0.643806 0.102261 0.280440 0.360105 0.760108 0.674790 0.643806 0.102261 df2 O.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 </th <th>1.5</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th>0.4057</th> <th>40</th> <th>0.461044</th>	1.5									0.4057	40	0.461044	
16 0.024142 0.726993 0.602587 0.815984 0.753234 0.515214 0.982483 0.124366 0.452646 0.757576 0.753234 0.515214 17 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 0.174145 0.295377 0.683534 0.326617 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 0.464004 0.849748 0.056447 0.424221 0.643806 0.102261 19 0.88417 0.725553 0.001559 0.273916 0.643806 0.102261 0.280440 0.360105 0.760108 0.674790 0.643806 0.102261 df2 Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.992171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 <	15									0.4257	42	0.461244	
0.982483 0.124366 0.452646 0.757576 17 0.42868 0.481441 0.671396 0.437300 0.565147 0.387528 0.174145 0.295377 0.683534 0.326617 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 0.464004 0.849748 0.056447 0.424221 0.643806 0.102261 19 0.88417 0.725553 0.001559 0.273916 0.643806 0.102261 0.280440 0.360105 0.760108 0.674790 0.643806 0.102261 df2 Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 <t< th=""><th>1.0</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>0.7520</th><th>2.4</th><th>0.515014</th></t<>	1.0									0.7520	2.4	0.515014	
17	10									0.7532.	34	0.515214	
0.174145 0.295377 0.683534 0.326617 18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 0.464004 0.849748 0.056447 0.424221 0.643806 0.102261 19 0.88417 0.725553 0.001559 0.273916 0.643806 0.102261 0.280440 0.360105 0.760108 0.674790 0.643806 0.102261 df2 Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987	17									0.5651	47	0.207520	
18 0.529209 0.236979 0.60565 0.002481 0.898732 0.043005 0.464004 0.849748 0.056447 0.424221 0.643806 0.102261 19 0.88417 0.725553 0.001559 0.273916 0.643806 0.102261 0.280440 0.360105 0.760108 0.674790 0.643806 0.102261 df2 Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987	1 /									0.30314	4/	0.38/328	
0.464004 0.849748 0.056447 0.424221 19 0.88417 0.725553 0.001559 0.273916 0.643806 0.102261 0.280440 0.360105 0.760108 0.674790 0.643806 0.102261 Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 0.338656 0.9922171 0.810851 0.249822 0.283435 0.181059 0.301486 0.789741 0.358560 0.738854 0.373372 0.934196 0.405396 0.146483 0.516349 0.259770 0.846987	10									0.0007	22	0.042005	
19	18									0.8987.	32	0.043005	
0.280440 0.360105 0.760108 0.674790 df2 Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987	10									0.6429	06	0.102261	
df2 Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987	19									0.0438	06	0.102201	
Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987		0.2804	40	0.3001	.03	0.760	108	0.0747	790				
Output: 0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987	df2												
0 1 2 3 4 0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987		t:											
0 0.988782 0.155982 0.163659 0.216378 0.338656 1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987	_		2	3	4								
1 0.922171 0.810851 0.249822 0.283435 0.181059 2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987		0.9887		_		0.163	559	0.2163	378	0.3386	56		
2 0.069235 0.844811 0.165427 0.086819 0.301486 3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987													
3 0.789741 0.358560 0.738854 0.373372 0.934196 4 0.405396 0.146483 0.516349 0.259770 0.846987		0.0692	:35	0.8448	311	0.1654	427	0.0868	319				
4 0.405396 0.146483 0.516349 0.259770 0.846987		0.7897	41	0.3585	660	0.7388	854	0.3733	372	0.9341	96		
5 0.929204 0.212274 0.604740 0.422453 0.722843		0.4053	96	0.1464	83	0.5163	349	0.2597	770	0.8469	87		
	5	0.9292	.04	0.2122	274	0.604	740	0.4224	153	0.7228	43		

6	0.247970	0.452907	0.853457	0.639186	0.590882
7	0.672903	0.397623	0.773096	0.071042	0.135975
8	0.139015	0.843306	0.936715	0.941274	0.551718
9	0.052673	0.486642	0.234463	0.257344	0.981282
df2.10	c[0,0]=990				
df2	L / J				
Outp		2 4			
0	1 2	3 4	0.4.0.5	0.01.40=0	0.000.
0	990.000000	0.155982	0.163659	0.216378	0.338656
1	0.922171	0.810851	0.249822	0.283435	0.181059
2	0.069235	0.844811	0.165427	0.086819	0.301486
3	0.789741	0.358560	0.738854	0.373372	0.934196
4	0.405396	0.146483	0.516349	0.259770	0.846987
5	0.929204	0.212274	0.604740	0.422453	0.722843
6	0.247970	0.452907	0.853457	0.639186	0.590882
7	0.672903	0.397623	0.773096	0.071042	0.135975
8	0.139015	0.843306	0.936715	0.941274	0.551718
9	0.052673	0.486642	0.234463	0.257344	0.981282
9	0.032073	0.480042	0.234403	0.23/344	0.961262
df2.c	olumns = list("A	ABCDE")			
df2					
Outp	nit:				
A	ВС	D E			
0	990.000000	0.155982	0.163659	0.216378	0.338656
1	0.922171	0.810851	0.249822	0.283435	0.181059
2	0.069235	0.844811	0.165427	0.086819	0.301486
3	0.789741	0.358560	0.738854	0.373372	0.934196
4	0.405396	0.146483	0.516349	0.259770	0.846987
5	0.929204	0.212274	0.604740	0.422453	0.722843
6	0.247970	0.452907	0.853457	0.639186	0.722843
7	0.672903	0.397623	0.773096	0.071042	0.135975
8	0.139015	0.843306	0.936715	0.941274	0.551718
9	0.052673	0.486642	0.234463	0.257344	0.981282
df2.1d	oc[0,'A']=89				
df2	3 c [0,11]-07				
	4				
Outp		ъ п			
Α	ВС	D E			
0	89.000000	899.000000	0.163659	0.216378	0.338656
1	0.922171	0.810851	0.249822	0.283435	0.181059
2	0.069235	0.844811	0.165427	0.086819	0.301486
3	0.789741	0.358560	0.738854	0.373372	0.934196
4	0.405396	0.146483	0.516349	0.259770	0.846987
5	0.929204	0.212274	0.604740	0.422453	0.722843
6	0.247970	0.452907	0.853457	0.639186	0.590882
7	0.672903	0.397623	0.773096	0.071042	0.135975
8	0.139015	0.843306	0.936715	0.941274	0.551718
9	0.052673	0.486642	0.234463	0.257344	0.981282
,	0.052015	0. 1 000 1 2	0.237 7 03	0.231377	0.701202
•	15 (5		(10.5)		
	pd.DataFrame(n	p.random.rand	(10,5))		
dt					
Oup	ut:				
0	1 2	3 4			
J		- '			

0 1 2 3	0.514973 0.505812 0.446541	0.132473 0.655760 0.850593	0.662300 0.709748 0.959236	0.870011 0.459002 0.653753	0.099254 0.258930 0.742279
	0.364539	0.001264	0.233297	0.904143	0.396865
4 5	0.214473 0.543493	0.344468 0.511075	0.010521 0.517688	0.403364 0.971037	0.834405 0.386030
<i>5</i>	0.343493	0.311073	0.317688	0.971037	0.537692
7	0.737970	0.310084	0.438818	0.767323	0.483544
8	0.383618	0.366597	0.258645	0.600649	0.044865
9	0.649240	0.894046	0.534226	0.551215	0.025614
dt[0][0	0]=88				
Outpu					
0	1 2	3 4	0.662200	0.050011	0.000274
0	88.000000	0.132473	0.662300	0.870011	0.099254
1	0.505812	0.655760	0.709748	0.459002	0.258930
2 3	0.446541 0.364539	0.850593	0.959236 0.233297	0.653753	0.742279
3 4	0.304339	0.001264 0.344468	0.233297	0.904143 0.403364	0.396865 0.834405
5	0.214473	0.544408	0.517688	0.403304	0.386030
6	0.757976	0.311073	0.385691	0.767525	0.537692
7	0.532578	0.294248	0.438818	0.581528	0.483544
8	0.383618	0.366597	0.258645	0.600649	0.044865
9	0.649240	0.894046	0.534226	0.551215	0.025614
dt.sort	t_index(axis=1	, ascending=Fa	lse)		
Outpu	*		,		
4	3 2	1 0			
0	0.099254	0.870011	0.662300	0.132473	88.000000
1	0.258930	0.459002	0.709748	0.655760	0.505812
1 2	0.258930 0.742279	0.459002 0.653753	0.709748 0.959236	0.655760 0.850593	0.505812 0.446541
1 2 3	0.258930 0.742279 0.396865	0.459002 0.653753 0.904143	0.709748 0.959236 0.233297	0.655760 0.850593 0.001264	0.505812 0.446541 0.364539
1 2 3 4	0.258930 0.742279 0.396865 0.834405	0.459002 0.653753 0.904143 0.403364	0.709748 0.959236 0.233297 0.010521	0.655760 0.850593 0.001264 0.344468	0.505812 0.446541 0.364539 0.214473
1 2 3 4 5	0.258930 0.742279 0.396865 0.834405 0.386030	0.459002 0.653753 0.904143 0.403364 0.971037	0.709748 0.959236 0.233297 0.010521 0.517688	0.655760 0.850593 0.001264 0.344468 0.511075	0.505812 0.446541 0.364539 0.214473 0.543493
1 2 3 4 5 6	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976
1 2 3 4 5 6 7	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578
1 2 3 4 5 6	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976
1 2 3 4 5 6 7 8 9	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618
1 2 3 4 5 6 7 8 9	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618
1 2 3 4 5 6 7 8 9	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649 0.551215	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618
1 2 3 4 5 6 7 8 9 dt[0][0 dt Outpu 0	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614 0]=0.9	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649 0.551215	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645 0.534226	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597 0.894046	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618 0.649240
1 2 3 4 5 6 7 8 9 dt[0][0 dt Outpu 0	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614 0]=0.9	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649 0.551215	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645 0.534226	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597 0.894046	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618 0.649240
1 2 3 4 5 6 7 8 9 dt[0][0 dt Outpu 0 0	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614 0]=0.9	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649 0.551215 3 4 0.132473 0.655760	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645 0.534226 0.662300 0.709748	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597 0.894046	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618 0.649240 0.099254 0.258930
1 2 3 4 5 6 7 8 9 dt[0][0 dt Outpu 0 0 1 2	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614 0]=0.9 ut: 1 2 0.900000 0.505812 0.446541	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649 0.551215 3 4 0.132473 0.655760 0.850593	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645 0.534226 0.662300 0.709748 0.959236	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597 0.894046 0.870011 0.459002 0.653753	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618 0.649240 0.099254 0.258930 0.742279
1 2 3 4 5 6 7 8 9 dt[0][0 dt Outpu 0 0 1 2 3	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614 0]=0.9 ut: 1 2 0.900000 0.505812 0.446541 0.364539	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649 0.551215 3 4 0.132473 0.655760 0.850593 0.001264	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645 0.534226 0.662300 0.709748 0.959236 0.233297	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597 0.894046 0.870011 0.459002 0.653753 0.904143	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618 0.649240 0.099254 0.258930 0.742279 0.396865
1 2 3 4 5 6 7 8 9 dt[0][0 dt Outpu 0 0 1 2 3 4	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614 0]=0.9 ut: 1 2 0.900000 0.505812 0.446541 0.364539 0.214473	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649 0.551215 3 4 0.132473 0.655760 0.850593 0.001264 0.344468	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645 0.534226 0.662300 0.709748 0.959236 0.233297 0.010521	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597 0.894046 0.870011 0.459002 0.653753 0.904143 0.403364	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618 0.649240 0.099254 0.258930 0.742279 0.396865 0.834405
1 2 3 4 5 6 7 8 9 dt[0][0 dt Outpu 0 0 1 2 3 4 5 5	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614 0]=0.9 ut: 1 2 0.900000 0.505812 0.446541 0.364539 0.214473 0.543493	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649 0.551215 3 4 0.132473 0.655760 0.850593 0.001264 0.344468 0.511075	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645 0.534226 0.662300 0.709748 0.959236 0.233297 0.010521 0.517688	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597 0.894046 0.870011 0.459002 0.653753 0.904143 0.403364 0.971037	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618 0.649240 0.099254 0.258930 0.742279 0.396865 0.834405 0.386030
1 2 3 4 5 6 7 8 9 dt[0][0 dt Outpu 0 0 1 2 3 4 5 6	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614 0]=0.9 ut: 1 2 0.900000 0.505812 0.446541 0.364539 0.214473 0.543493 0.757976	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649 0.551215 3 4 0.132473 0.655760 0.850593 0.001264 0.344468 0.511075 0.310684	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645 0.534226 0.662300 0.709748 0.959236 0.233297 0.010521 0.517688 0.385691	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597 0.894046 0.870011 0.459002 0.653753 0.904143 0.403364 0.971037 0.767525	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618 0.649240 0.099254 0.258930 0.742279 0.396865 0.834405 0.386030 0.537692
1 2 3 4 5 6 7 8 9 dt[0][0 dt Outpu 0 0 1 2 3 4 5 5	0.258930 0.742279 0.396865 0.834405 0.386030 0.537692 0.483544 0.044865 0.025614 0]=0.9 ut: 1 2 0.900000 0.505812 0.446541 0.364539 0.214473 0.543493	0.459002 0.653753 0.904143 0.403364 0.971037 0.767525 0.581528 0.600649 0.551215 3 4 0.132473 0.655760 0.850593 0.001264 0.344468 0.511075	0.709748 0.959236 0.233297 0.010521 0.517688 0.385691 0.438818 0.258645 0.534226 0.662300 0.709748 0.959236 0.233297 0.010521 0.517688	0.655760 0.850593 0.001264 0.344468 0.511075 0.310684 0.294248 0.366597 0.894046 0.870011 0.459002 0.653753 0.904143 0.403364 0.971037	0.505812 0.446541 0.364539 0.214473 0.543493 0.757976 0.532578 0.383618 0.649240 0.099254 0.258930 0.742279 0.396865 0.834405 0.386030

dt.columns = list("abcde") **Output:** a c d e 0.132473 0.099254 0 0.900000 0.662300 0.870011 1 0.655760 0.505812 0.709748 0.459002 0.258930 2 0.446541 0.850593 0.959236 0.653753 0.742279 3 0.364539 0.001264 0.233297 0.904143 0.396865 4 0.403364 0.834405 0.214473 0.344468 0.010521 5 0.543493 0.511075 0.517688 0.971037 0.386030 6 0.757976 0.310684 0.385691 0.767525 0.537692 7 0.532578 0.294248 0.438818 0.581528 0.483544 8 0.383618 0.366597 0.258645 0.600649 0.044865 9 0.649240 0.894046 0.534226 0.551215 0.025 dt.loc[0,b']=68dt **Output:** b c d a 0 0.900000 68.000000 0.662300 0.870011 0.099254 1 0.505812 0.655760 0.709748 0.459002 0.258930 2 0.446541 0.850593 0.959236 0.653753 0.742279 3 0.364539 0.001264 0.233297 0.904143 0.396865 4 0.834405 0.344468 0.010521 0.403364 0.214473 5 0.971037 0.511075 0.517688 0.386030 0.543493 6 0.757976 0.310684 0.385691 0.767525 0.537692 7 0.294248 0.483544 0.532578 0.438818 0.581528 0.600649 8 0.383618 0.366597 0.258645 0.044865 9 0.649240 0.894046 0.534226 0.551215 0.025614 dt.loc[0,0]=98dt **Output:** d 0 a c e 0 0.900000 68.000000 0.662300 0.870011 0.099254 98.0 NaN 1 0.505812 0.655760 0.709748 0.459002 0.258930 2 NaN 0.446541 0.850593 0.959236 0.653753 0.742279 3 NaN 0.364539 0.001264 0.233297 0.904143 0.396865 4 NaN 0.214473 0.344468 0.010521 0.403364 0.834405 5 NaN 0.543493 0.511075 0.517688 0.971037 0.386030 6 0.757976 0.310684 0.385691 0.767525 0.537692 NaN 7 NaN 0.532578 0.294248 0.438818 0.581528 0.483544 0.600649 8 0.383618 0.258645 0.044865 NaN 0.366597 9 0.649240 0.894046 0.534226 0.551215 0.025614 NaN

dt.drop(0,axis=1)						
	put:					
a	b c	d e				
0	0.900000	68.000000	0.662300	0.870011	0.099254	
1	0.505812	0.655760	0.709748	0.459002	0.258930	
2	0.446541	0.850593	0.959236	0.653753	0.742279	
3	0.364539	0.001264	0.233297	0.904143	0.396865	
4	0.214473	0.344468	0.010521	0.403364	0.834405	

5	0.543493	0.511075	0.517688	0.971037	0.386030
6	0.757976	0.310684	0.385691	0.767525	0.537692
7	0.532578	0.294248	0.438818	0.581528	0.483544
8	0.383618	0.366597	0.258645	0.600649	0.044865
9	0.649240	0.894046	0.534226	0.551215	0.025614

newdt = dt.drop(0,axis=1)
newdt

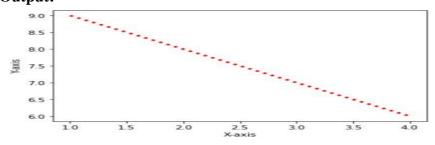
Output:

a	b c	d e			
0	0.900000	68.000000	0.662300	0.870011	0.099254
1	0.505812	0.655760	0.709748	0.459002	0.258930
2	0.446541	0.850593	0.959236	0.653753	0.742279
3	0.364539	0.001264	0.233297	0.904143	0.396865
4	0.214473	0.344468	0.010521	0.403364	0.834405
5	0.543493	0.511075	0.517688	0.971037	0.386030
6	0.757976	0.310684	0.385691	0.767525	0.537692
7	0.532578	0.294248	0.438818	0.581528	0.483544
8	0.383618	0.366597	0.258645	0.600649	0.044865
9	0.649240	0.894046	0.534226	0.551215	0.025614

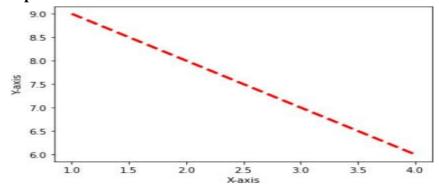
Matplotlib:

```
import matplotlib.pyplot as plx = [1, 2, 3, 4]y = [9, 8, 7, 6]pl.xlabel("X-axis")pl.ylabel("Y-axis")pl.plot(x,y,'r',linewidth = 3, linestyle = 'dashdot')pl.show()
```

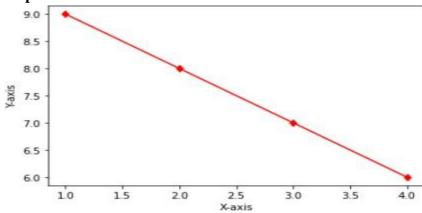
Output:



```
pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.plot(x,y,'r',linewidth = 3, linestyle = 'dashed')
pl.show()
```

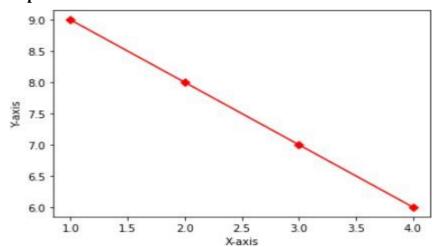


```
pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.plot(x,y,'r',marker='d')
pl.show()
```

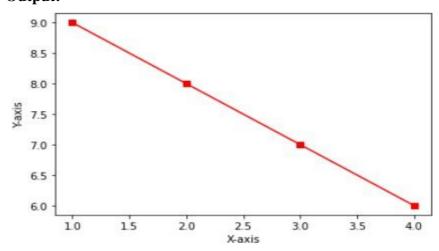


pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.plot(x,y,'r',marker='D')
pl.show()

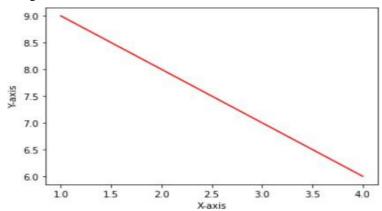
Output:



pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.plot(x,y,'r',marker='s')
pl.show()

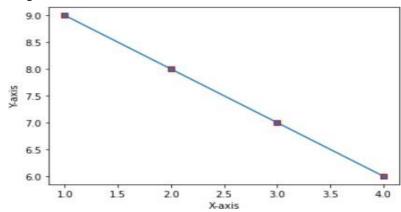


```
pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.plot(x,y,'r',marker=")
pl.show()
```

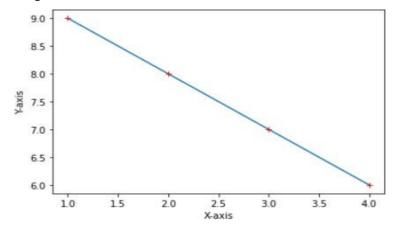


```
pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.plot(x,y,marker='s', markeredgecolor='red')
pl.show()
```

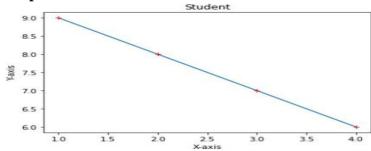
Output:



```
pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.plot(x,y,marker='+', markeredgecolor='red')
pl.show()
```

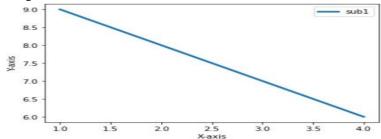


```
pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.title("Student")
pl.plot(x,y,marker='+', markeredgecolor='red')
pl.show()
```



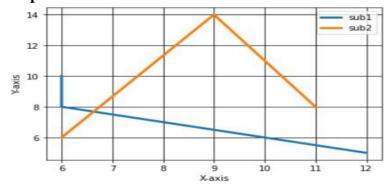
pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.plot(x,y,linewidth=3, label="sub1")
pl.legend()
pl.show()

Output:



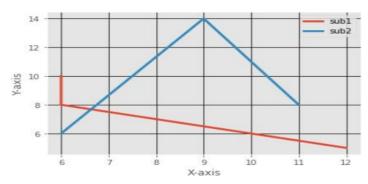
x2 = [6, 9, 11] y = [5, 8, 10] x = [12, 6, 6] y2 = [6, 14, 8] pl.xlabel("X-axis") pl.ylabel("Y-axis") pl.plot(x,y,linewidth=3, label="sub1") pl.plot(x2,y2,linewidth=3, label="sub2") pl.legend() pl.show()

Output:



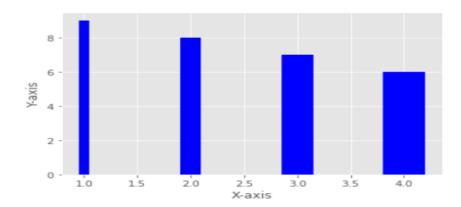
from matplotlib import style style.use('ggplot') pl.xlabel("X-axis")

```
pl.ylabel("Y-axis")
pl.plot(x,y,linewidth=3, label="sub1")
pl.plot(x2,y2,linewidth=3, label="sub2")
pl.legend()
pl.grid(True, color = 'k')
pl.show()
```

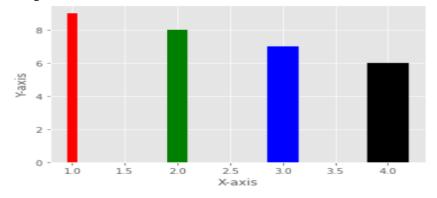


x = [1,2,3,4]
y = [9,8,7,6]
pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.bar(x,y, width=[0.1,0.2,0.3,0.4], color = 'b')
pl.show()

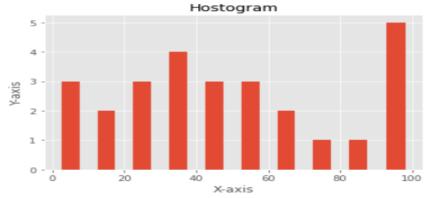
Output:



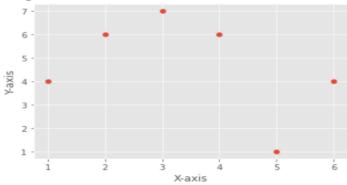
$$\begin{split} x &= [1,2,3,4] \\ y &= [9,8,7,6] \\ pl.xlabel("X-axis") \\ pl.ylabel("Y-axis") \\ pl.bar(x,y, width=[0.1,0.2,0.3,0.4], color = ['r','g','b','k']) \\ pl.show() \end{split}$$



```
population_age = [50,30,60,76,34,29,90,100,9,8,5,23,65,34,21,54,87,98,43,56,
45,99,12,10,44,35,101]
bins = [0,10,20,30,40,50,60,70,80,90,101]
pl.hist(population_age,bins,histtype = 'bar', rwidth=0.5)
pl.xlabel("X-axis")
pl.ylabel("Y-axis")
pl.title("Hostogram")
pl.show()
```



x = [1,2,3,4,5,6] y = [4,6,7,6,1,4] pl.scatter(x,y) pl.xlabel("X-axis") pl.ylabel("Y-axis") pl.show()



Practical No: 3

Aim: Study of Supervised Learning

- a) Linear Regression
- b) Logistic Regression
- c) K Nearest Neighbour Algorithm

a) Linear Regression

import numpy as np
import pandas as pd
import matplotlib.pyplot as pl
from sklearn import linear_model
df = pd.read_csv("dataset1.csv")
df

Output:

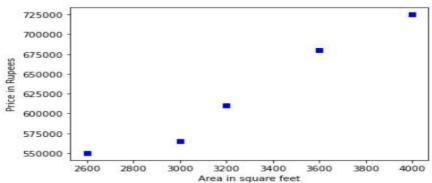
area price 0 2600 550000 1 3000 565000 2 3200 610000 3 3600 680000 4 4000 725000

df.shape Output: (5, 2)

%matplotlib inline pl.xlabel('Area in square feet') pl.ylabel('Price in Rupees') pl.scatter(df.area,df.price, color='b', marker='s')

Output:

[24]: <matplotlib.collections.PathCollection at 0x13360421a90>



reg = linear_model.LinearRegression()
reg.fit(df[['area']],df.price)

Output: LinearRegression()

reg.coef_ #coef is m (slope)

Output: array([135.78767123])

reg.predict([[3300]])

Output: array([628715.75342466]) reg.intercept_ #intercept is b

Output: 180616.43835616432

y = m * x + b

135.78767123*3300+180616.43835616432

Output: 628715.7534151643

b) Logistic Regression

import numpy as np import pandas as pd from matplotlib import pyplot as pl

df1 = pd.read_csv("loan.csv")
df1.head()

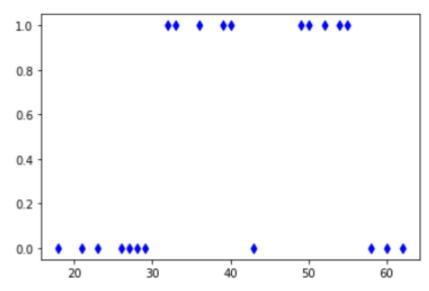
Output:

	age	bought_loan
0	23	0
1	18	0
2	55	1
3	43	0
4	36	1

pl.scatter(df1.age,df1.bought_loan, marker='d', color='b')

Output:

<matplotlib.collections.PathCollection at 0x25b5e99d160>



from sklearn.model_selection import train_test_split x_train, x_test,y_train,y_test = train_test_split(df1[['age']], df1.bought_loan, train_size=0.9, shuffle=False)

x_test

Output:

age 18 33 19 60 20 52

from sklearn.linear_model import LogisticRegression logreg = LogisticRegression()

logreg.fit(x_train,y_train)

Output: LogisticRegression()

logreg.predict(x_test)

Output: array([0, 1, 1], dtype=int64)

c) K Nearest Neighbour Algorithm

import numpy as np
import pandas as pd
ds = pd.read_csv("iris.csv")
ds

Output:

	ld	${\sf SepalLengthCm}$	${\sf SepalWidthCm}$	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
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

ds.iloc[: ,1:5]

Output:

	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	5.1	3.5	1.4
1	4.9	3.0	1.4
2	4.7	3.2	1.3
3	4.6	3.1	1.5
4	5.0	3.6	1.4
145	6.7	3.0	5.2
146	6.3	2.5	5.0
147	6.5	3.0	5.2
148	6.2	3.4	5.4
149	5.9	3.0	5.1

```
150 rows × 3 columns

x = ds.iloc[:, 1:5].values

y = ds.iloc[:, 5].values

from sklearn.preprocessing import LabelEncoder #0 and nclass-1

lblenc_y = LabelEncoder()

y = lblenc_y.fit_transform(y)

y
```

from sklearn.model_selection import train_test_split x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2) from sklearn.neighbors import KNeighborsClassifier knn_model = KNeighborsClassifier(n_neighbors=5) knn_model.fit(x_train, y_train)

Output: KNeighborsClassifier()

y_predict = knn_model.predict(x_test)
from sklearn.metrics import confusion_matrix, classification_report
print(confusion_matrix(y_test,y_predict))

Output:

[[13 0 0] [0 7 2] [0 0 8]]

#accuracy print(28/30)

Output: 0.9333333333333333

print(classification_report(y_test, y_predict))

	precision	recall	f1-score	support
0	1.00	1.00 0.91	1.00	9 11
2	0.90	0.90	0.90	10
accuracy			0.93	30
macro avg weighted avg	0.94 0.93	0.94 0.93	0.94 0.93	30 30

Practical No: 4.

Aim: Implementation of Bagging Algorithm: Decision Tree.

a) Decision Tree classifier

import pandas as pd
ds = pd.read_csv("IrisNew.csv")
ds.head()

	ld	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 = ds.iloc[:,1:5]

y = ds.iloc[:,5]

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test

from sklearn.tree import DecisionTreeClassifier

dt_classifier = DecisionTreeClassifier()

dt_classifier.fit(X_train, y_train)

DecisionTreeClassifier()

y_pred = dt_classifier.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score

cf = confusion_matrix(y_test, y_pred)

cf

ac = accuracy_score(y_test, y_pred)

ac

0.91111111111111111

b) Bagging classifier

import pandas as pd
ds = pd.read_csv("IrisNew.csv")
ds.head()

	ld	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 = ds.iloc[:,1:5]

y = ds.iloc[:,5]

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test

from sklearn.ensemble import BaggingClassifier

bagclassifier = BaggingClassifier()

bagclassifier.fit(X train, y train)

BaggingClassifier()

```
y_pred = bagclassifier.predict(X_test)
from sklearn.metrics import confusion matrix, accuracy score
bag_cs = confusion_matrix(y_test, y_pred)
bag_cs
 array([[15, 0, 0],
             [ 0, 13, 1],
             [ 0, 2, 14]], dtype=int64)
bag_ac = accuracy_score(y_test, y_pred)
bag ac
0.9333333333333333
import pickle
with open('bagModelIris.pkl','wb') as file:
  pickle.dump(bagclassifier, file)
import flask
from flask import Flask, request
model bagging = pickle.load(open('bagModelIris.pkl','rb'))
app = Flask( name )
# Two parts (base address + route address)
@app.route('/', methods = ['GET','POST'])
def main():
  return "Begging Flask API Development"
@app.route('/classify', methods = ['GET'])
def classify():
  if flask.request.method == 'GET':
     SepalLengthCm = request.args.get('sl')
     SepalWidthCm = request.args.get('sw')
     PetalLengthCm = request.args.get('pl')
     PetalWidthCm = request.args.get('pw')
     prediction = model_bagging.predict([[SepalLengthCm, SepalWidthCm, PetalLengthCm,
PetalWidthCm]])
     return 'Class of species is '+str(prediction)
if name == ' main ':
  app.run()
    Serving Flask app "__main__" (lazy loading)
Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Aug/2021 22:55:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2021 22:56:36] "GET /classify?sl=1.2&
Testing using Postman
 DET http://127.0.0.1:50...
                                                                                 No Environment
  http://127.0.0.1:5000/classify?sl=1.2&sw=3.1&pl=2.4&pw=5.6
                                                                                                http://127.0.0.1:5000/classify?sl=1.2&sw=3.1&pl=2.4&pw=5.6
 Params Authorization Headers (6) Body Pre-request Script Tests Settings
      KEY
                                       VALUE
                                                                        DESCRIPTION
  ✓ sl
  ✓ sw
                                       3.1
  pw pw
                                       5.6
 Body Cookies Headers (4) Test Results
                                                                     ② 200 OK 53 ms 191 B Save Response
 Pretty Raw Preview Visualize HTML ~ 🚍
                                                                                                    ■ Q
    1 Class of species is ['Iris-virginica']
```

Practical No: 5

Aim: Implementation of K-Means Clustering

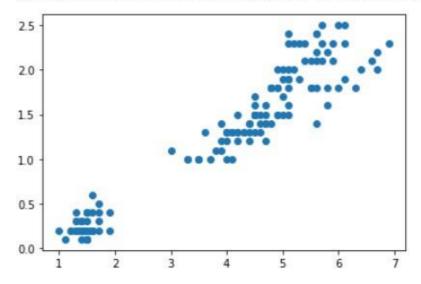
Find the exact / proper K # Elbow method

import pandas as pd
import numpy as np
ds = pd.read_csv("Iris.csv")
ds.head()

	PetalLength	PetalWidth	Species
0	1.4	0.2	Iris-setosa
1	1.4	0.2	Iris-setosa
2	1.3	0.2	Iris-setosa
3	1.5	0.2	Iris-setosa
4	1.4	0.2	Iris-setosa

from matplotlib import pyplot as pl pl.scatter(ds['PetalLength'],ds['PetalWidth'])

<matplotlib.collections.PathCollection at 0x25124fdc3d0>



from sklearn.cluster import KMeans kmean = KMeans(n_clusters = 3) kmean

KMeans(n_clusters=3)

```
y_predict = kmean.fit_predict(ds[['PetalLength','PetalWidth']])
y_predict
```

	PetalLength	PetalWidth	Species	cluster
0	1.4	0.2	Iris-setosa	1
1	1.4	0.2	Iris-setosa	1
2	1.3	0.2	Iris-setosa	1
3	1.5	0.2	Iris-setosa	1
4	1.4	0.2	Iris-setosa	1
145	5.2	2.3	Iris-virginica	2
146	5.0	1.9	Iris-virginica	2
147	5.2	2.0	Iris-virginica	2
148	5.4	2.3	Iris-virginica	2
149	5.1	1.8	Iris-virginica	2

150 rows × 4 columns

kmean.cluster_centers_

```
array([[4.26923077, 1.34230769],
[1.464 , 0.244 ],
[5.59583333, 2.0375 ]])
```

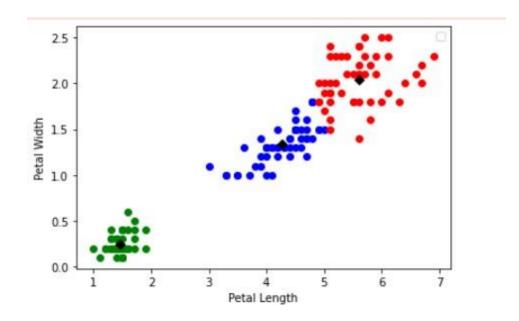
ds1 = ds[ds.cluster == 0]

ds2 = ds[ds.cluster == 1]

ds3 = ds[ds.cluster == 2]

pl.show()

pl.scatter(ds1.PetalLength,ds1.PetalWidth, color = 'blue')
pl.scatter(ds2.PetalLength,ds2.PetalWidth, color = 'red')
pl.scatter(ds3.PetalLength,ds3.PetalWidth, color = 'green')
pl.scatter(kmean.cluster_centers_[:,0], kmean.cluster_centers_[:,1], color = 'black', marker='D')
pl.xlabel('Petal Length')
pl.ylabel('Petal Width')
pl.legend()



since values of x and y are mismatch # so we need to scale the values

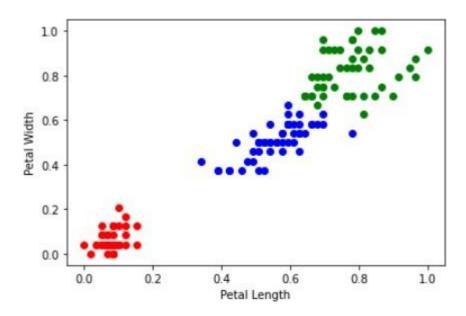
```
from sklearn.preprocessing import MinMaxScaler
scl = MinMaxScaler()
scl.fit(ds[['PetalLength']])
ds['PetalLength'] = scl.transform(ds[['PetalLength']])
scl.fit(ds[['PetalWidth']])
ds['PetalWidth'] = scl.transform(ds[['PetalWidth']])
# Applying KMean once again
kmean = KMeans(n\_clusters = 3)
y predict = kmean.fit predict(ds[['PetalLength', 'PetalWidth']])
y_predict
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2,
    ds['cluster'] = y predict
```

	PetalLength	PetalWidth	Species	cluster
0	0.067797	0.041667	Iris-setosa	1
1	0.067797	0.041667	Iris-setosa	1
2	0.050847	0.041667	Iris-setosa	1
3	0.084746	0.041667	Iris-setosa	1
4	0.067797	0.041667	Iris-setosa	1
	***	***		
145	0.711864	0.916667	Iris-virginica	2
146	0.677966	0.750000	Iris-virginica	2
147	0.711864	0.791667	Iris-virginica	2
148	0.745763	0.916667	Iris-virginica	2
149	0.694915	0.708333	Iris-virginica	2

150 rows × 4 columns

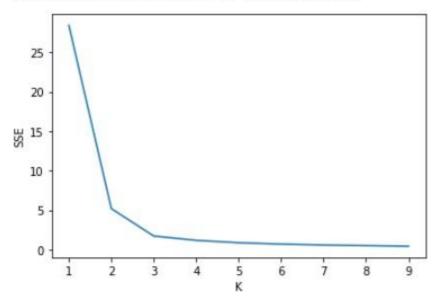
ds

```
ds1 = ds[ds.cluster == 0]
ds2 = ds[ds.cluster == 1]
ds3 = ds[ds.cluster == 2]
pl.scatter(ds1.PetalLength,ds1.PetalWidth, color = 'blue')
pl.scatter(ds2.PetalLength,ds2.PetalWidth, color = 'red')
pl.scatter(ds3.PetalLength,ds3.PetalWidth, color = 'green')
pl.xlabel('Petal Length')
pl.ylabel('Petal Width')
pl.show()
```



```
k_range = range(1,10)
sse = []
for k in k_range:
  kmean = KMeans(n\_clusters = k)
  kmean.fit(ds[['PetalLength','PetalWidth']])
  sse.append(kmean.inertia_)
[28.391514358368717,
5.179687509974783,
1.7050986081225123,
1.1621031930971286,
0.8570856553216398,
0.6833274904190353,
0.5683512655008139,
0.48911635449076774,
0.4155388630360096]
pl.xlabel('K')
pl.ylabel('SSE')
pl.plot(k_range, sse)
```

[<matplotlib.lines.Line2D at 0x1c46ba07d90>]



Practical No: 6

Aim: Implementation of dimensionality reduction techniques:

- 1)Normalization
- 2)Principal Components Analysis.

1) Normalization

import numpy as np
import pandas as pd
ds = pd.read_csv("FeatureSelection.csv")
ds.head()

	age	weight	height	cholestrol	sugar	Target
0	35	70	150	233	250	1
1	56	75	100	250	300	0
2	67	68	180	204	260	0
3	72	60	170	236	450	1
4	39	77	190	354	220	1

Simple feature scaling

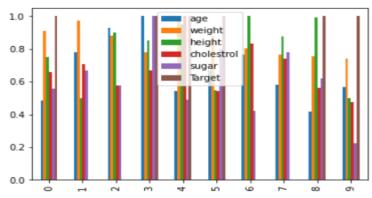
for column in ds.columns:

ds[column]=ds[column]/ds[column].abs().max()
ds.head()

	age	weight	height	cholestrol	sugar	Target
0	0.486111	0.909091	0.75	0.658192	0.555556	1.0
1	0.777778	0.974026	0.50	0.706215	0.666667	0.0
2	0.930556	0.883117	0.90	0.576271	0.577778	0.0
3	1.000000	0.779221	0.85	0.666667	1.000000	1.0
4	0.541667	1.000000	0.95	1.000000	0.488889	1.0

import matplotlib.pyplot as pl
ds.plot(kind='bar')

<AxesSubplot:>



Min Max method xold=(xold-xmin)/(xmax-xmin)

ds1 = ds.copy()

for column in ds1.columns:

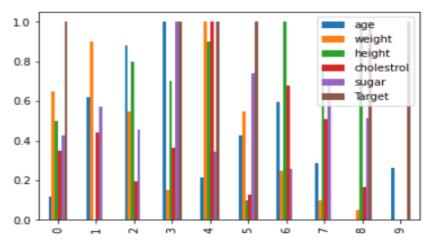
 $ds1[column] = (ds1[column] - ds1[column].min()) \, / \, (ds1[column].max() - ds1[column].min())$

ds1.head()

	age	weight	height	cholestrol	sugar	Target
0	0.119048	0.65	0.5	0.349462	0.428571	1.0
1	0.619048	0.90	0.0	0.440860	0.571429	0.0
2	0.880952	0.55	0.8	0.193548	0.457143	0.0
3	1.000000	0.15	0.7	0.365591	1.000000	1.0
4	0.214286	1.00	0.9	1.000000	0.342857	1.0

ds1.plot(kind='bar')

<AxesSubplot:>



Standardization (Z score method or 0 mean)

ds2=ds.copy()

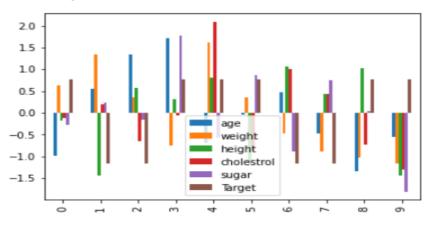
for column in ds2.columns:

 $ds2[column] = (ds2[column] - ds2[column].mean()) / ds2[column].std() \\ ds2.head()$

	age	weight	height	cholestrol	sugar	Target
0	-0.980397	0.637633	-0.182525	-0.114881	-0.266226	0.774597
1	0.544665	1.330712	-1.432694	0.195116	0.245747	-1.161895
2	1.343507	0.360401	0.567577	-0.643699	-0.163831	-1.161895
3	1.706617	-0.748525	0.317543	-0.060176	1.781663	0.774597
4	-0.689909	1.607944	0.817611	2.091567	-0.573409	0.774597

ds2.plot(kind='bar')

<AxesSubplot:>



2) Principal Components Analysis

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pl
ds = pd.read_csv("wine.csv")
ds.head()
```

	Alcohol	Malic.acid	Ash	AcI	Mg	Phenols	Flavanoids	Nonflavanoid.phenols	Proanth	Color.int	Hue	OD	Proline	Wine
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065	1
1	13.20	1.78	2.14	11.2	100	2,65	2.76	0.26	1.28	4.38	1.05	3,40	1050	1
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185	1
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480	1
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735	1

X = ds.iloc[:,0:13]

X.head()

	Alcohol	Malic.acid	Ash	AcI	Mg	Phenols	Flavanoids	Nonflavanoid.phenols	Proanth	Color.int	Hue	OD	Proline
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3,92	1065
1	13.20	1.78	2.14	11.2	100	2,65	2.76	0.26	1.28	4.38	1.05	3,40	1050
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

```
y = ds.iloc[:,-1]
```

y.head()

- 0 1
- 2 1
- 3 1

Name: Wine, dtype: int64

from sklearn.model_selection import train_test_split

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

from sklearn.preprocessing import StandardScaler

st = StandardScaler()

fit and transform

```
X_train = st.fit_transform(X_train)
```

```
X_train
```

```
array([[ 0.86085012, -0.83342199, 0.47848374, ..., -0.21754034, 0.38175836, 1.84584136], [-0.73835419, 1.70964634, 1.29881334, ..., -0.08954008, 0.73014966, -1.20811907], [-0.22168818, 0.46354285, 0.87081528, ..., -0.21754034, -0.62160859, -0.46122657], ..., [-1.47644849, -0.58759206, -1.73283953, ..., -0.00420658, -0.21747468, -1.04214296], [-1.15660763, -1.08772883, 0.51415024, ..., 1.57446322, 0.17272357, -0.32844568], [ 1.27910356, -0.66388411, -0.30617936, ..., 0.63579469, 1.55235313, 0.16948265]])
```

 $X_{test} = st.transform(X_{test})$

Applying PCA

from sklearn.decomposition import PCA

Check Eigenvalue of components

```
pca = PCA(n components = 2)
```

X_train = pca.fit_transform(X_train)

 $X_{test} = pca.transform(X_{test})$

pca.explained_variance_ratio_

array([0.36138769, 0.1937306])

Sorted Descending order

Now classification

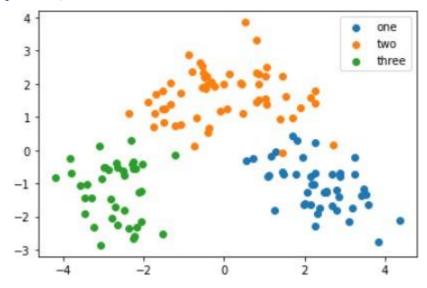
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
LogisticRegression()
y_test
29
149
63
158
22
Name: Wine, dtype: int64
y_predict = logreg.predict(X_test)
y_predict
array([1, 3, 2, 3, 1, 1, 2, 2, 2, 1, 1, 3, 2, 3, 2, 1, 3, 3, 1, 1, 3, 2,
    1, 1, 2, 1, 1, 2, 2, 3, 3, 1, 2, 3, 1, 2], dtype=int64)
# check with Confusion Metrix for Actual vs Predict
from sklearn.metrics import confusion_matrix
c = confusion_matrix(y_test, y_predict)
```

c array([[13, 1, 0], [1, 11, 1], [0, 0, 9]], dtype=int64)

#3 category

Visualize 2 components (0th and 1st)

```
 \begin{array}{l} X\_disp, y\_disp = X\_train, y\_train \\ pl.scatter(X\_disp[ y\_disp == 1,0], X\_disp[ y\_disp == 1,1], label = 'one') \\ pl.scatter(X\_disp[ y\_disp == 2,0], X\_disp[ y\_disp == 2,1], label = 'two') \\ pl.scatter(X\_disp[ y\_disp == 3,0], X\_disp[ y\_disp == 3,1], label = 'three') \\ pl.legend() \\ pl.show() \end{array}
```



Practical No: 7

Aim: Study of Support Vector Machines (SVMs).

import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()

iris.feature_names

['sepal length (cm)',
'sepal width (cm)',
'petal length (cm)',
'petal width (cm)']

ds = pd.DataFrame(iris.data, columns=iris.feature_names)
ds.head()

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Appending one column as target column

ds['target'] = iris.target
ds.head()

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

0 ----> Setosa 1--> Versicolor 2---> Virginica

iris.target_names

array(['setosa', 'versicolor', 'virginica'], dtype='<U10')

#Want to see the number of rows for each flower type

ds[ds.target==2]

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2
101	5.8	2.7	5.1	1.9	2
102	7.1	3.0	5.9	2.1	2
103	6.3	2.9	5.6	1.8	2
104	6.5	3.0	5.8	2.2	2
105	7.6	3.0	6.6	2.1	2
106	4.9	2.5	4.5	1.7	2
14	16 6.3	2.5	5.0	1.9	2
14	7 6.5	3.0	5.2	2.0	2
14	6.2	3.4	5.4	2.3	2
14	19 5.9	3.0	5.1	1.8	2

Want to append flower name column ---> based on index (0, 1, 2) i.e Setosa and all

ds['fname'] = ds.target.apply(lambda x: iris.target_names[x])
ds.head()

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	fname
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1,4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

from matplotlib import pyplot as pl

Create 3 DF for three flowers

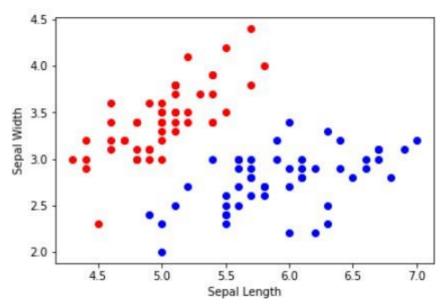
ds1 = ds[ds.target==0]

ds2 = ds[ds.target==1]

ds3 = ds[ds.target==2]

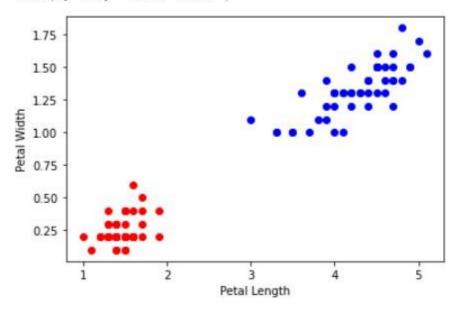
pl.scatter(ds1['sepal length (cm)'], ds1['sepal width (cm)'], color = 'red', marker = 'o') pl.scatter(ds2['sepal length (cm)'], ds2['sepal width (cm)'], color = 'blue', marker = 'o') pl.xlabel('Sepal Length') pl.ylabel('Sepal Width')

Text(0, 0.5, 'Sepal Width')



pl.scatter(ds1['petal length (cm)'], ds1['petal width (cm)'], color = 'red', marker = 'o') pl.scatter(ds2['petal length (cm)'], ds2['petal width (cm)'], color = 'blue', marker = 'o') pl.xlabel('Petal Length') pl.ylabel('Petal Width')

Text(0, 0.5, 'Petal Width')



from sklearn.model_selection import train_test_split # Remove fname column X = ds.drop(['target','fname'], axis = 'columns')

X.head()

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

y = ds['target']

y.head()

0 0

1 0

2 0

3 0

4 0

Name: target, dtype: int32

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False) from sklearn.svm import SVC

svmodel = SVC(kernel='linear')

svmodel.fit(X_train, y_train)

SVC(kernel='linear')

Accuracy of model

svmodel.score(X_train, y_train)

0.99166666666666667