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

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
```

```
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
       [6.6, 2.9, 4.6, 1.3],
       [5.2, 2.7, 3.9, 1.4],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3. , 4.2, 1.5],
       [6. , 2.2, 4. , 1. ],
       [6.1, 2.9, 4.7, 1.4],
       [5.6, 2.9, 3.6, 1.3],
       [6.7, 3.1, 4.4, 1.4],
       [5.6, 3. , 4.5, 1.5],
       [5.8, 2.7, 4.1, 1. ],
       [6.2, 2.2, 4.5, 1.5],
       [5.6, 2.5, 3.9, 1.1],
       [5.9, 3.2, 4.8, 1.8],
       [6.1, 2.8, 4. , 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.9, 4.3, 1.3],
       [6.6, 3. , 4.4, 1.4],
       [6.8, 2.8, 4.8, 1.4],
       [6.7, 3. , 5. , 1.7],
       [6. , 2.9, 4.5, 1.5],
       [5.7, 2.6, 3.5, 1. ],
       [5.5, 2.4, 3.8, 1.1],
       [5.5, 2.4, 3.7, 1. ],
       [5.8, 2.7, 3.9, 1.2],
       [6. , 2.7, 5.1, 1.6],
       [5.4, 3. , 4.5, 1.5],
       [6. , 3.4, 4.5, 1.6],
       [6.7, 3.1, 4.7, 1.5],
       [6.3, 2.3, 4.4, 1.3],
       [5.6, 3. , 4.1, 1.3],
       [5.5, 2.5, 4. , 1.3],
       [5.5, 2.6, 4.4, 1.2],
       [6.1, 3. , 4.6, 1.4],
       [5.8, 2.6, 4. , 1.2],
       [5. , 2.3, 3.3, 1. ],
```

```
       [5.6, 2.7, 4.2, 1.3],
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
```

```
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
'frame': None, 'target_names': array(['setosa', 'versicolor',
'virginica'], dtype='<U10'), 'DESCR': '.. _iris_dataset:\n\nIris
plants dataset\n--------------------\n\n**Data Set Characteristics:**\
n\n    :Number of Instances: 150 (50 in each of three classes)\
n    :Number of Attributes: 4 numeric, predictive attributes and the
class\n    :Attribute Information:\n        - sepal length in cm\n
- sepal width in cm\n        - petal length in cm\n        - petal
width in cm\n        - class:\n                - Iris-Setosa\n
- Iris-Versicolour\n                - Iris-Virginica\
n                \n    :Summary Statistics:\n\n    ============== ====
==== ======= ===== ====================\n                    Min  Max
Mean    SD   Class Correlation\n    ============== ==== ==== =======
===== ====================\n    sepal length:   4.3  7.9   5.84   0.83
0.7826\n    sepal width:    2.0  4.4   3.05   0.43   -0.4194\n
petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\n    petal
width:    0.1  2.5   1.20   0.76    0.9565  (high!)\n
============== ==== ==== ======= ===== ====================\n\
n    :Missing Attribute Values: None\n    :Class Distribution: 33.3%
for each of 3 classes.\n    :Creator: R.A. Fisher\n    :Donor: Michael
Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe
famous Iris database, first used by Sir R.A. Fisher. The dataset is
taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not
as in the UCI\nMachine Learning Repository, which has two wrong data
points.\n\nThis is perhaps the best known database to be found in the\
npattern recognition literature.  Fisher\'s paper is a classic in the
field and\nis referenced frequently to this day.  (See Duda & Hart,
for example.)  The\ndata set contains 3 classes of 50 instances each,
where each class refers to a\ntype of iris plant.  One class is
linearly separable from the other 2; the\nlatter are NOT linearly
separable from each other.\n\n|details-start|\n**References**\n|
```

```
details-split|\n\n- Fisher, R.A. "The use of multiple measurements in taxonomic problems"\n  Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to\n  Mathematical Statistics" (John Wiley, NY, 1950).\n- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n  (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n  Structure and Classification Rule for Recognition in Partially Exposed\n  Environments".  IEEE Transactions on Pattern Analysis and Machine\n  Intelligence, Vol. PAMI-2, No. 1, 67-71.\n- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n  on Information Theory, May 1972, 431-433.\n- See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n conceptual clustering system finds 3 classes in the data.\n- Many, many more ...\n\n|details-end|', 'feature_names': ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'], 'filename': 'iris.csv', 'data_module': 'sklearn.datasets.data'}

iris.keys()

dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])

print(type(iris.data))

<class 'numpy.ndarray'>

print(iris.target)

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

print(iris.data.shape)

(150, 4)

print(iris.target_names)

['setosa' 'versicolor' 'virginica']

x=iris.data
y=iris.target
print(x)
print(y)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.2]
 [5.  3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.6 1.4 0.1]
 [4.4 3.  1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.  3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.  3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.  1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.  3.3 1.4 0.2]
```

```
[7.  3.2 4.7 1.4]
[6.4 3.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.  1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
[6.3 3.3 4.7 1.6]
[4.9 2.4 3.3 1. ]
[6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4]
[5.  2.  3.5 1. ]
[5.9 3.  4.2 1.5]
[6.  2.2 4.  1. ]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
[5.8 2.7 4.1 1. ]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.  5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
[5.  2.3 3.3 1. ]
[5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
[5.7 2.8 4.1 1.3]
```

```
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
[6.5 3.2 5.1 2. ]
[6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
[5.7 2.5 5.  2. ]
[5.8 2.8 5.1 2.4]
[6.4 3.2 5.3 2.3]
[6.5 3.  5.5 1.8]
[7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3]
[6.  2.2 5.  1.5]
[6.9 3.2 5.7 2.3]
[5.6 2.8 4.9 2. ]
[7.7 2.8 6.7 2. ]
[6.3 2.7 4.9 1.8]
[6.7 3.3 5.7 2.1]
[7.2 3.2 6.  1.8]
[6.2 2.8 4.8 1.8]
[6.1 3.  4.9 1.8]
[6.4 2.8 5.6 2.1]
[7.2 3.  5.8 1.6]
[7.4 2.8 6.1 1.9]
[7.9 3.8 6.4 2. ]
[6.4 2.8 5.6 2.2]
[6.3 2.8 5.1 1.5]
[6.1 2.6 5.6 1.4]
[7.7 3.  6.1 2.3]
[6.3 3.4 5.6 2.4]
[6.4 3.1 5.5 1.8]
[6.  3.  4.8 1.8]
[6.9 3.1 5.4 2.1]
[6.7 3.1 5.6 2.4]
[6.9 3.1 5.1 2.3]
[5.8 2.7 5.1 1.9]
[6.8 3.2 5.9 2.3]
[6.7 3.3 5.7 2.5]
[6.7 3.  5.2 2.3]
[6.3 2.5 5.  1.9]
[6.5 3.  5.2 2. ]
[6.2 3.4 5.4 2.3]
[5.9 3.  5.1 1.8]]
```

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

Convert Data set to data frame

```
a=pd.DataFrame(x,columns=iris.feature_names)
print(a)
```

```
     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
..                 ...               ...                ...
...
145                6.7               3.0                5.2
2.3
146                6.3               2.5                5.0
1.9
147                6.5               3.0                5.2
2.0
148                6.2               3.4                5.4
2.3
149                5.9               3.0                5.1
1.8

[150 rows x 4 columns]
```

Display first and last 5 rows

```
print(a.head())
print(a.tail())
print(a.describe())
```

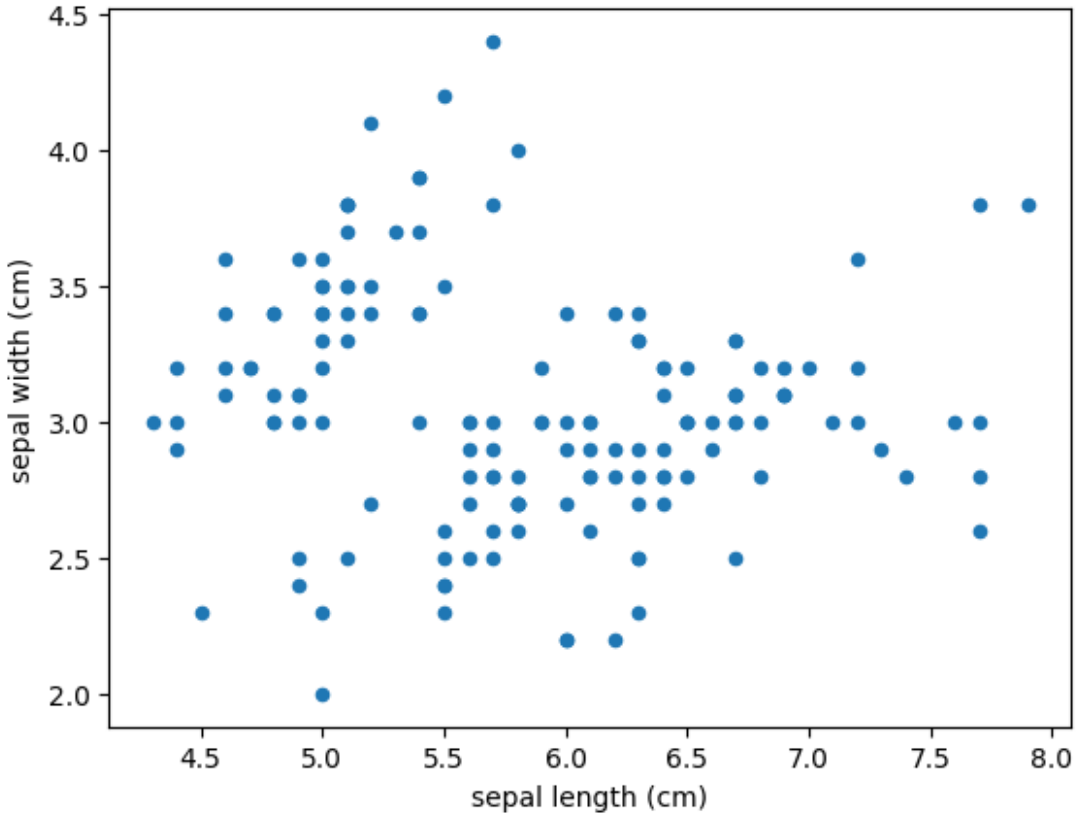
```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal
width (cm)
145                6.7               3.0                5.2
2.3
146                6.3               2.5                5.0
1.9
147                6.5               3.0                5.2
2.0
148                6.2               3.4                5.4
2.3
149                5.9               3.0                5.1
1.8
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
```

```
a.min()
a.max()
a.mean()
a.median()
a.std()
```

```
sepal length (cm)    0.828066
sepal width (cm)     0.435866
petal length (cm)    1.765298
petal width (cm)     0.762238
dtype: float64
```

```
import matplotlib.pyplot as plt
#a.plot(x='sepal length (cm)',y='sepal width (cm)',kind='scatter')
```

```
<Axes: xlabel='sepal length (cm)', ylabel='sepal width (cm)'>
```




```
d=datasets.load_diabetes()
print(d)
```

```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -
0.00259226,
         0.01990749, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06833155, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286131, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04688253,  0.01549073],
```

```
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452873, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00422151,  0.00306441]]), 'target': array([151.,  75.,
141., 206., 135.,  97., 138.,  63., 110., 310., 101.,
        69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,
49.,
        68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59.,
341.,
        87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,
92.,
       259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104.,
182.,
       128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71.,
163.,
       150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42.,
170.,
       200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55.,
134.,
        42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,
92.,
        83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,
81.,
       104.,  59., 246., 297., 258., 229., 275., 281., 179., 200.,
200.,
       173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274.,
158.,
       107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317.,
235.,
        60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,
71.,
       197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131.,
214.,
        59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151.,
127.,
       237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101.,
137.,
       143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72.,
129.,
       142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202.,
155.,
        77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214.,
185.,
        78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53.,
220.,
       154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275.,
177.,
        71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,
91.,
```

```
       150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308.,
116.,
       145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,
66.,
        94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156.,
233.,
        60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,
89.,
        31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44.,
172.,
       114., 142., 109., 180., 144., 163., 147.,  97., 220., 190.,
109.,
       191., 122., 230., 242., 248., 249., 192., 131., 237.,  78.,
135.,
       244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95.,
216.,
       263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,
71.,
        77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182.,
321.,
        58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71.,
168.,
       140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,
69.,
       219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273.,
258.,
        43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,
72.,
       140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,
55.,
        84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67.,
310.,
        94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104.,
132.,
       220.,  57.]), 'frame': None, 'DESCR': '.. _diabetes_dataset:\n\
nDiabetes dataset\n----------------\n\nTen baseline variables, age,
sex, body mass index, average blood\npressure, and six blood serum
measurements were obtained for each of n =\n442 diabetes patients, as
well as the response of interest, a\nquantitative measure of disease
progression one year after baseline.\n\n**Data Set Characteristics:**\
n\n  :Number of Instances: 442\n\n  :Number of Attributes: First 10
columns are numeric predictive values\n\n  :Target: Column 11 is a
quantitative measure of disease progression one year after baseline\n\
n  :Attribute Information:\n      - age     age in years\n      - sex\
n      - bmi     body mass index\n      - bp      average blood
pressure\n      - s1      tc, total serum cholesterol\n      - s2
ldl, low-density lipoproteins\n      - s3      hdl, high-density
lipoproteins\n      - s4      tch, total cholesterol / HDL\n      - s5
ltg, possibly log of serum triglycerides level\n      - s6      glu,
```

```
blood sugar level\n\nNote: Each of these 10 feature variables have been mean centered and scaled by the standard deviation times the square root of `n_samples` (i.e. the sum of squares of each column totals 1).\n\nSource URL:\nhttps://www4.stat.ncsu.edu/~boos/var.select/diabetes.html\n\nFor more information see:\nBradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," Annals of Statistics (with discussion), 407-499.\n(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)\n', 'feature_names': ['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6'], 'data_filename': 'diabetes_data_raw.csv.gz', 'target_filename': 'diabetes_target.csv.gz', 'data_module': 'sklearn.datasets.data'}
```

MATRIX

```
import numpy as np
a=np.array([[12,45],[23,98]])
print(a)
b=np.array([[35,78],[34,65]])
print(b)
print("ADDITION")
print(np.add(a,b))
print("SUBTRACTION")
print(np.subtract(a,b))
print("MULTIPLICATION")
print(np.multiply(a,b))
print("DIVISION")
print(np.divide(a,b))
```

```
[[12 45]
 [23 98]]
[[35 78]
 [34 65]]
ADDITION
[[ 47 123]
 [ 57 163]]
SUBTRACTION
[[-23 -33]
 [-11  33]]
MULTIPLICATION
[[ 420 3510]
 [ 782 6370]]
DIVISION
[[0.34285714 0.57692308]
 [0.67647059 1.50769231]]
```

```
import pandas as pd
df=pd.DataFrame([[101,'Aliya',59,90,89],[102,'Berna',90,87,67],
```

```
[103,'Celina',90,87,45],[104,'Dalini',90,56,78],
[105,'Elisa',89,78,68]],columns=['Roll
No','Name','Mark1','Mark2','Mark3'])
print(df)
```

```
   Roll No    Name  Mark1  Mark2  Mark3
0      101   Aliya     59     90     89
1      102   Berna     90     87     67
2      103  Celina     90     87     45
3      104  Dalini     90     56     78
4      105   Elisa     89     78     68
```

```
print(df.loc[0:3])
print(df.iloc[0:3,1:3])
print(df.iloc[0:3])
```

```
   Roll No    Name  Mark1  Mark2  Mark3
0      101   Aliya     59     90     89
1      102   Berna     90     87     67
2      103  Celina     90     87     45
3      104  Dalini     90     56     78
     Name  Mark1
0   Aliya     59
1   Berna     90
2  Celina     90
   Roll No    Name  Mark1  Mark2  Mark3
0      101   Aliya     59     90     89
1      102   Berna     90     87     67
2      103  Celina     90     87     45
```

LINEAR EQUATION

```
import pandas as pd
import numpy as np
a=np.array([[1,1],[1,-1]])
print(a)
b=np.array([2,0])
print(b)
r=np.linalg.solve(a,b)
print(r)
```

```
[[ 1  1]
 [ 1 -1]]
[2 0]
[1. 1.]
```

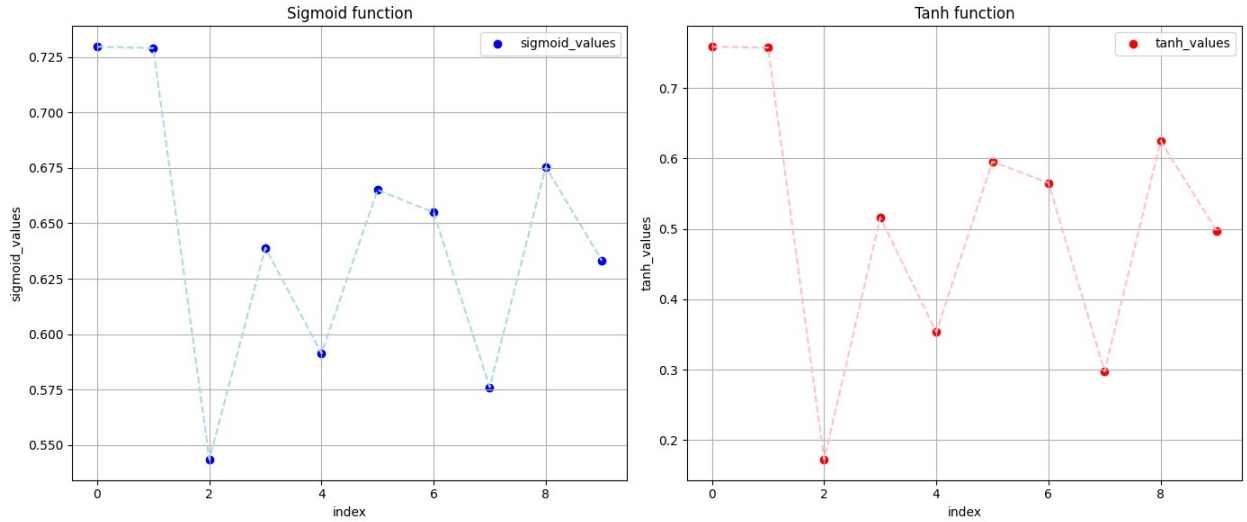
```
import pandas as pd
import numpy as np
a=np.array([[1,1,1],[1,-1,1],[-1,-1,1]])
print(a)
```

```
b=np.array([6,2,3])
print(b)
r=np.linalg.solve(a,b)
print(r)
```

```
[[ 1  1  1]
 [ 1 -1  1]
 [-1 -1  1]]
[6 2 3]
[-0.5  2.   4.5]
```

Sigmoid and tanh

```
import numpy as np
import matplotlib.pyplot as plt
def sigmoid(x):
    return 1/(1+np.exp(-x))
def tanh(x):
  return np.tanh(x)
random_values=np.random.rand(10)
sigmoid_values=sigmoid(random_values)
tanh_values=tanh(random_values)
indices=np.arange(len(random_values))
plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
plt.scatter(indices,sigmoid_values,color='blue',label='sigmoid_values'
)
plt.plot(indices,sigmoid_values,color='lightblue',linestyle='--')
plt.title('Sigmoid function')
plt.xlabel('index')
plt.ylabel('sigmoid_values')
plt.grid(True)
plt.legend()
plt.subplot(1,2,2)
plt.scatter(indices,tanh_values,color='red',label='tanh_values')
plt.plot(indices,tanh_values,color='pink',linestyle='--')
plt.title('Tanh function')
plt.xlabel('index')
plt.ylabel('tanh_values')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```




Build in data set

```
from sklearn import datasets
import pandas as pd
iris=datasets.load_iris()
print(iris)
print("\ntype:\n",type(iris))
print("\nkeys:\n",iris.keys())
print("\ntype of data and target:\n",type(iris.data), type
(iris.target))
print("\ndata shape:\n",iris.data.shape)
print("\ntarget names:\n",iris.target_names)
X = iris.data
Y = iris.target
df = pd.DataFrame(X, columns=iris.feature_names)
print("\nIris dataframe:\n", df.head())
print("----------\n")
diabetes = datasets.load_diabetes()
print("\ndiabetes dataset:\n", diabetes);
X = diabetes.data
Y = diabetes.target
df = pd.DataFrame(X, columns=diabetes.feature_names)
print("\nDiabetes dataframe:\n",df.head())
print("---------\n")
data = datasets.load_breast_cancer()
label_names = data['target_names']
labels = data['target']
feature_names = data['feature_names']
features= data['data']
print("Breast Cancer data:\n", data);
print("\nLabel names:\n", label_names)
print("\nLabels:\n", labels)
```

```
print("\nFeature names:\n", feature_names)
print("\nFeatures:\n", features)
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
```

```
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
       [6.6, 2.9, 4.6, 1.3],
       [5.2, 2.7, 3.9, 1.4],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3. , 4.2, 1.5],
       [6. , 2.2, 4. , 1. ],
       [6.1, 2.9, 4.7, 1.4],
       [5.6, 2.9, 3.6, 1.3],
       [6.7, 3.1, 4.4, 1.4],
       [5.6, 3. , 4.5, 1.5],
       [5.8, 2.7, 4.1, 1. ],
       [6.2, 2.2, 4.5, 1.5],
       [5.6, 2.5, 3.9, 1.1],
       [5.9, 3.2, 4.8, 1.8],
       [6.1, 2.8, 4. , 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.9, 4.3, 1.3],
       [6.6, 3. , 4.4, 1.4],
       [6.8, 2.8, 4.8, 1.4],
       [6.7, 3. , 5. , 1.7],
       [6. , 2.9, 4.5, 1.5],
       [5.7, 2.6, 3.5, 1. ],
       [5.5, 2.4, 3.8, 1.1],
       [5.5, 2.4, 3.7, 1. ],
       [5.8, 2.7, 3.9, 1.2],
       [6. , 2.7, 5.1, 1.6],
       [5.4, 3. , 4.5, 1.5],
       [6. , 3.4, 4.5, 1.6],
       [6.7, 3.1, 4.7, 1.5],
       [6.3, 2.3, 4.4, 1.3],
       [5.6, 3. , 4.1, 1.3],
       [5.5, 2.5, 4. , 1.3],
       [5.5, 2.6, 4.4, 1.2],
       [6.1, 3. , 4.6, 1.4],
       [5.8, 2.6, 4. , 1.2],
       [5. , 2.3, 3.3, 1. ],
       [5.6, 2.7, 4.2, 1.3],
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
```

```
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
```

```
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
'frame': None, 'target_names': array(['setosa', 'versicolor',
'virginica'], dtype='<U10'), 'DESCR': '.. _iris_dataset:\n\nIris
plants dataset\n--------------------\n\n**Data Set Characteristics:**\
n\n:Number of Instances: 150 (50 in each of three classes)\n:Number of
Attributes: 4 numeric, predictive attributes and the class\n:Attribute
Information:\n    - sepal length in cm\n    - sepal width in cm\n    -
petal length in cm\n    - petal width in cm\n    - class:\n
- Iris-Setosa\n            - Iris-Versicolour\n            - Iris-
Virginica\n\n:Summary Statistics:\n\n============== ==== ==== =======
===== ====================\n                Min  Max   Mean    SD
Class Correlation\n============== ==== ==== ======= =====
====================\nsepal length:   4.3  7.9   5.84   0.83
0.7826\nsepal width:    2.0  4.4   3.05   0.43   -0.4194\npetal
length:   1.0  6.9   3.76   1.76    0.9490  (high!)\npetal width:
0.1  2.5   1.20   0.76    0.9565  (high!)\n============== ==== ====
======= ===== ====================\n\n:Missing Attribute Values: None\
n:Class Distribution: 33.3% for each of 3 classes.\n:Creator: R.A.
Fisher\n:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\
n:Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A.
Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s
the same as in R, but not as in the UCI\nMachine Learning Repository,
which has two wrong data points.\n\nThis is perhaps the best known
database to be found in the\npattern recognition literature.
Fisher\'s paper is a classic in the field and\nis referenced
frequently to this day.  (See Duda & Hart, for example.)  The\ndata
set contains 3 classes of 50 instances each, where each class refers
to a\ntype of iris plant.  One class is linearly separable from the
other 2; the\nlatter are NOT linearly separable from each other.\n\n..
dropdown:: References\n\n  - Fisher, R.A. "The use of multiple
measurements in taxonomic problems"\n    Annual Eugenics, 7, Part II,
179-188 (1936); also in "Contributions to\n    Mathematical
Statistics" (John Wiley, NY, 1950).\n  - Duda, R.O., & Hart, P.E.
(1973) Pattern Classification and Scene Analysis.\n    (Q327.D83) John
Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n  - Dasarathy, B.V.
(1980) "Nosing Around the Neighborhood: A New System\n    Structure
```

```
and Classification Rule for Recognition in Partially Exposed\n
Environments".  IEEE Transactions on Pattern Analysis and Machine\n
Intelligence, Vol. PAMI-2, No. 1, 67-71.\n  - Gates, G.W. (1972) "The
Reduced Nearest Neighbor Rule".  IEEE Transactions\n    on Information
Theory, May 1972, 431-433.\n  - See also: 1988 MLC Proceedings, 54-64.
Cheeseman et al"s AUTOCLASS II\n    conceptual clustering system finds
3 classes in the data.\n  - Many, many more ...\n', 'feature_names':
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal
width (cm)'], 'filename': 'iris.csv', 'data_module':
'sklearn.datasets.data'}

type:
 <class 'sklearn.utils._bunch.Bunch'>

keys:
 dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR',
'feature_names', 'filename', 'data_module'])

type of data and target:
 <class 'numpy.ndarray'> <class 'numpy.ndarray'>

data shape:
 (150, 4)

target names:
 ['setosa' 'versicolor' 'virginica']

Iris dataframe:
    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
----------


diabetes dataset:
 {'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -
0.00259226,
         0.01990749, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06833155, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
```

```
         0.00286131, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04688253,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452873, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00422151,  0.00306441]]), 'target': array([151.,  75.,
141., 206., 135.,  97., 138.,  63., 110., 310., 101.,
        69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,
49.,
        68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59.,
341.,
        87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,
92.,
       259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104.,
182.,
       128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71.,
163.,
       150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42.,
170.,
       200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55.,
134.,
        42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,
92.,
        83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,
81.,
       104.,  59., 246., 297., 258., 229., 275., 281., 179., 200.,
200.,
       173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274.,
158.,
       107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317.,
235.,
        60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,
71.,
       197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131.,
214.,
        59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151.,
127.,
       237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101.,
137.,
       143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72.,
129.,
       142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202.,
155.,
        77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214.,
185.,
        78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53.,
220.,
       154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275.,
```

```
177.,
        71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,
91.,
       150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308.,
116.,
       145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,
66.,
        94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156.,
233.,
        60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,
89.,
        31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44.,
172.,
       114., 142., 109., 180., 144., 163., 147.,  97., 220., 190.,
109.,
       191., 122., 230., 242., 248., 249., 192., 131., 237.,  78.,
135.,
       244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95.,
216.,
       263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,
71.,
        77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182.,
321.,
        58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71.,
168.,
       140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,
69.,
       219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273.,
258.,
        43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,
72.,
       140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,
55.,
        84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67.,
310.,
        94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104.,
132.,
       220.,  57.]), 'frame': None, 'DESCR': '.. _diabetes_dataset:\n\
nDiabetes dataset\n----------------\n\nTen baseline variables, age,
sex, body mass index, average blood\npressure, and six blood serum
measurements were obtained for each of n =\n442 diabetes patients, as
well as the response of interest, a\nquantitative measure of disease
progression one year after baseline.\n\n**Data Set Characteristics:**\
n\n:Number of Instances: 442\n\n:Number of Attributes: First 10
columns are numeric predictive values\n\n:Target: Column 11 is a
quantitative measure of disease progression one year after baseline\n\
n:Attribute Information:\n    - age     age in years\n    - sex\n    -
bmi     body mass index\n    - bp      average blood pressure\n    -
s1      tc, total serum cholesterol\n    - s2      ldl, low-density
lipoproteins\n    - s3      hdl, high-density lipoproteins\n    - s4
```

```
tch, total cholesterol / HDL\n    - s5      ltg, possibly log of serum
triglycerides level\n    - s6      glu, blood sugar level\n\nNote:
Each of these 10 feature variables have been mean centered and scaled
by the standard deviation times the square root of `n_samples` (i.e.
the sum of squares of each column totals 1).\n\nSource
URL:\nhttps://www4.stat.ncsu.edu/~boos/var.select/diabetes.html\n\nFor
more information see:\nBradley Efron, Trevor Hastie, Iain Johnstone
and Robert Tibshirani (2004) "Least Angle Regression," Annals of
Statistics (with discussion),
407-499.\n(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_200
2.pdf)\n', 'feature_names': ['age', 'sex', 'bmi', 'bp', 's1', 's2',
's3', 's4', 's5', 's6'], 'data_filename': 'diabetes_data_raw.csv.gz',
'target_filename': 'diabetes_target.csv.gz', 'data_module':
'sklearn.datasets.data'}

Diabetes dataframe:
        age       sex       bmi        bp        s1        s2
s3  \
0  0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -
0.043401
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163
0.074412
2  0.085299  0.050680  0.044451 -0.005670 -0.045599 -0.034194 -
0.032356
3 -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -
0.036038
4  0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596
0.008142

         s4        s5        s6
0 -0.002592  0.019907 -0.017646
1 -0.039493 -0.068332 -0.092204
2 -0.002592  0.002861 -0.025930
3  0.034309  0.022688 -0.009362
4 -0.002592 -0.031988 -0.046641
---------

Breast Cancer data:
 {'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01,
4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
```

```
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1,
1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
'frame': None, 'target_names': array(['malignant', 'benign'],
dtype='<U9'), 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer
wisconsin (diagnostic) dataset\
n--------------------------------------------\n\n**Data Set
Characteristics:**\n\n:Number of Instances: 569\n\n:Number of
Attributes: 30 numeric, predictive attributes and the class\n\
n:Attribute Information:\n    - radius (mean of distances from center
to points on the perimeter)\n    - texture (standard deviation of
gray-scale values)\n    - perimeter\n    - area\n    - smoothness
(local variation in radius lengths)\n    - compactness (perimeter^2 /
area - 1.0)\n    - concavity (severity of concave portions of the
contour)\n    - concave points (number of concave portions of the
contour)\n    - symmetry\n    - fractal dimension ("coastline
approximation" - 1)\n\n    The mean, standard error, and "worst" or
largest (mean of the three\n    worst/largest values) of these
features were computed for each image,\n    resulting in 30 features.
For instance, field 0 is Mean Radius, field\n    10 is Radius SE,
field 20 is Worst Radius.\n\n    - class:\n            - WDBC-
Malignant\n            - WDBC-Benign\n\n:Summary Statistics:\n\
n===================================== ====== ======\n
Min    Max\n===================================== ====== ======\
nradius (mean):                          6.981  28.11\ntexture (mean):
9.71   39.28\nperimeter (mean):                      43.79  188.5\narea
(mean):                          143.5  2501.0\nsmoothness (mean):
0.053  0.163\ncompactness (mean):                   0.019  0.345\
nconcavity (mean):                       0.0    0.427\nconcave points
(mean):                0.0    0.201\nsymmetry (mean):
0.106  0.304\nfractal dimension (mean):             0.05   0.097\
nradius (standard error):              0.112  2.873\ntexture (standard
error):             0.36   4.885\nperimeter (standard error):
0.757  21.98\narea (standard error):                6.802  542.2\
nsmoothness (standard error):          0.002  0.031\ncompactness
(standard error):         0.002  0.135\nconcavity (standard error):
0.0    0.396\nconcave points (standard error):      0.0    0.053\
nsymmetry (standard error):            0.008  0.079\nfractal dimension
(standard error):   0.001  0.03\nradius (worst):
7.93   36.04\ntexture (worst):                      12.02  49.54\
nperimeter (worst):                    50.41  251.2\narea (worst):
185.2  4254.0\nsmoothness (worst):                   0.071  0.223\
ncompactness (worst):                  0.027  1.058\nconcavity
(worst):                    0.0    1.252\nconcave points (worst):
0.0    0.291\nsymmetry (worst):                     0.156  0.664\
nfractal dimension (worst):            0.055  0.208\
n===================================== ====== ======\n\n:Missing
Attribute Values: None\n\n:Class Distribution: 212 - Malignant, 357 -
Benign\n\n:Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L.
Mangasarian\n\n:Donor: Nick Street\n\n:Date: November, 1995\n\nThis is
a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.\

nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of a fine needle\naspirate (FNA) of a breast mass.  They describe\ncharacteristics of the cell nuclei present in the image.\n\nSeparating plane described above was obtained using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via Linear Programming." Proceedings of the 4th\nMidwest Artificial Intelligence and Cognitive Science Society,\npp. 97-101, 1992], a classification method which uses linear\nprogramming to construct a decision tree.  Relevant features\nwere selected using an exhaustive search in the space of 1-4\nfeatures and 1-3 separating planes.\n\nThe actual linear program used to obtain the separating plane\nin the 3-dimensional space is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Linearly Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis database is also available through the UW CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. dropdown:: References\n\n  - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction\n    for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on\n Electronic Imaging: Science and Technology, volume 1905, pages 861-870,\n    San Jose, CA, 1993.\n  - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and\n    prognosis via linear programming. Operations Research, 43(4), pages 570-577,\n    July-August 1995.\n  - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques\n    to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)\n    163-171.\n',
'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
'filename': 'breast_cancer.csv', 'data_module': 'sklearn.datasets.data'}

Label names:
 ['malignant' 'benign']

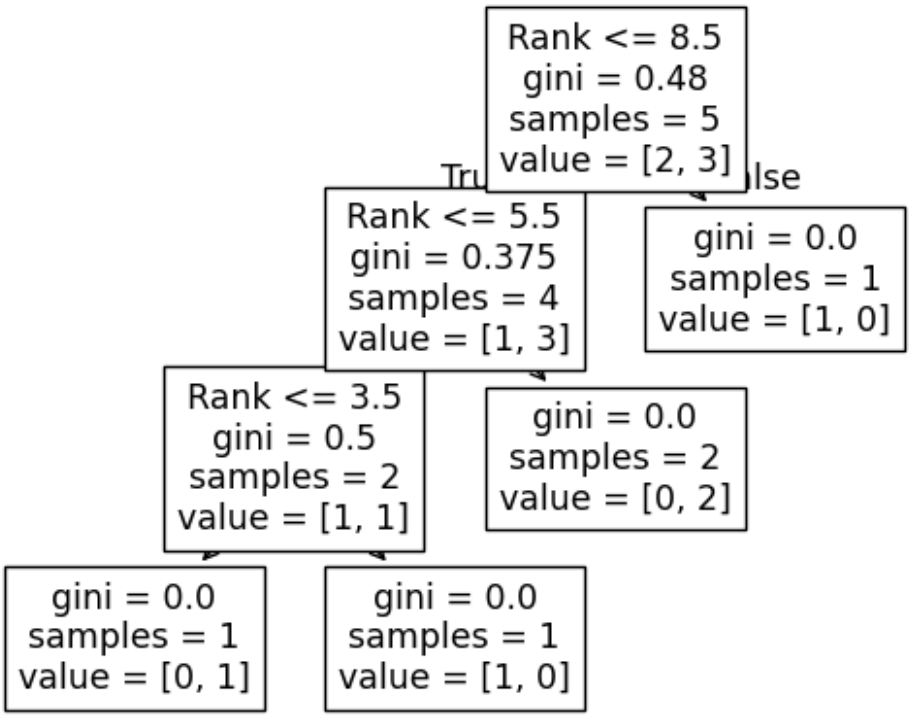
Labels:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0

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

Feature names:
 ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']

Features:
 [[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
```

Decision tree

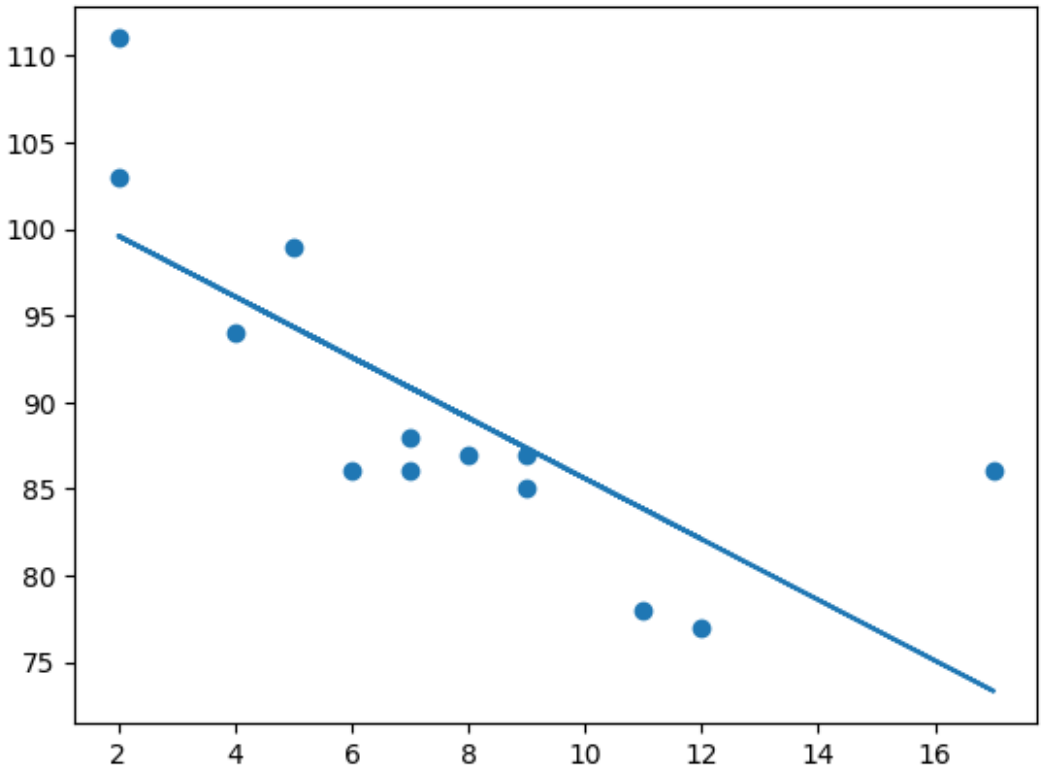
```
import pandas as pd
data = {
    'Age': [25, 30, 35, 40, 45],
    'Experience': [1, 3, 5, 10, 15],
    'Rank': [3, 4, 7, 9, 8],
    'Nationality': ['UK', 'USA', 'N', 'UK', 'USA'],
    'Go': ['YES', 'NO', 'YES', 'NO', 'YES']
}
df = pd.DataFrame(data)
df.to_csv('decision_tree.csv', index=False)

import matplotlib.pyplot as plt
import pandas
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
df = pandas.read_csv("decision_tree.csv")
print(df)
d = {'UK': 0, 'USA': 1, 'N':2}
df['Nationality'] = df['Nationality'].map(d)
d = {'YES': 1, 'NO': 0}
df['Go']=df['Go'].map(d)
features = ['Age', 'Experience', 'Rank', 'Nationality']
x= df[features]
y=df['Go']
dtree = DecisionTreeClassifier()
dtree= dtree.fit(x, y)
tree.plot_tree(dtree, feature_names=features)
plt.show()
```

```
   Age  Experience  Rank Nationality   Go
0   25           1     3          UK  YES
1   30           3     4         USA   NO
2   35           5     7           N  YES
3   40          10     9          UK   NO
4   45          15     8         USA  YES
```



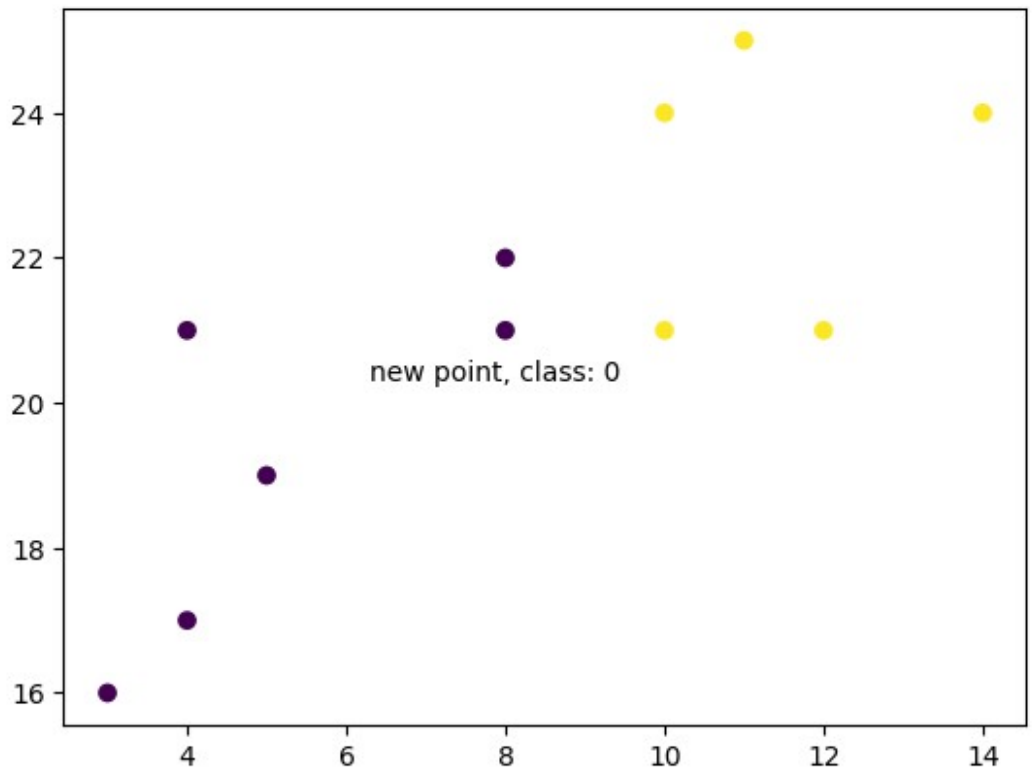

Linear Regression

```
import matplotlib.pyplot as plt
from scipy import stats
x= [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]
slope, intercept, r, p, std_err = stats.linregress(x, y)
def myfunc(x):
  return slope *x + intercept
mymodel = list(map(myfunc, x))
plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```




K Nearest Neighbour

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
x= [4, 5, 10, 4, 3, 11, 14, 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]
data = list(zip(x, y))
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(data, classes)
new_x = 8
new_y=21
new_point = [(new_x, new_y)]
prediction = knn.predict(new_point)
plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class:
{prediction[0]}")
plt.show()
```

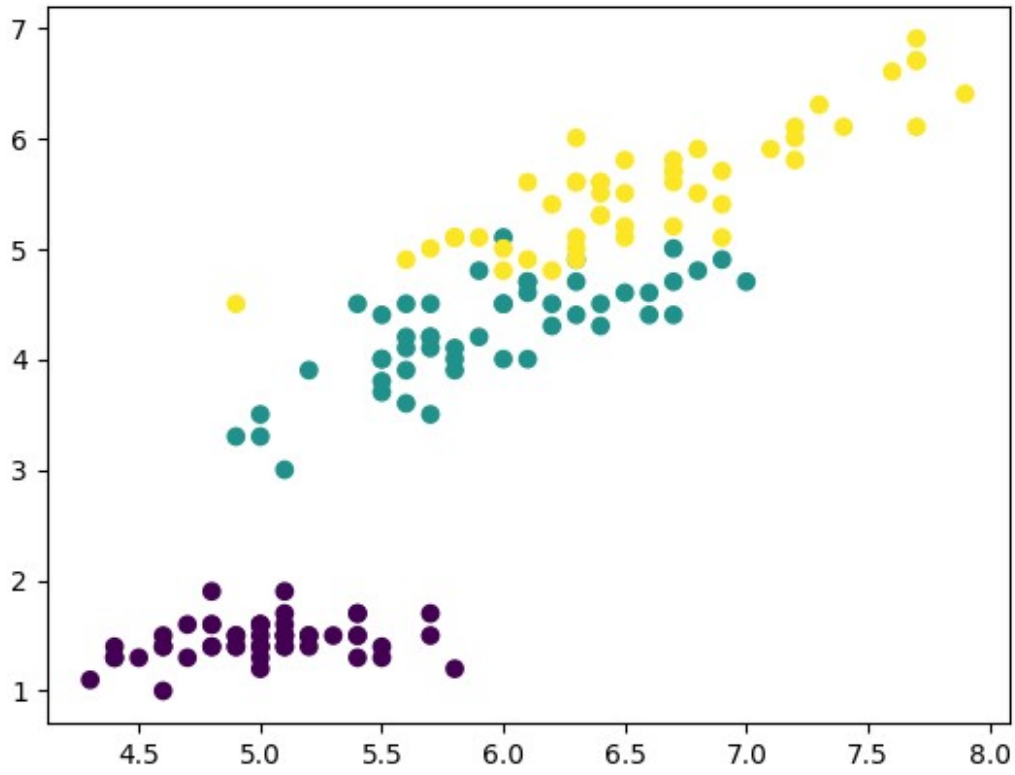



Unsupervised Learining

Clustering

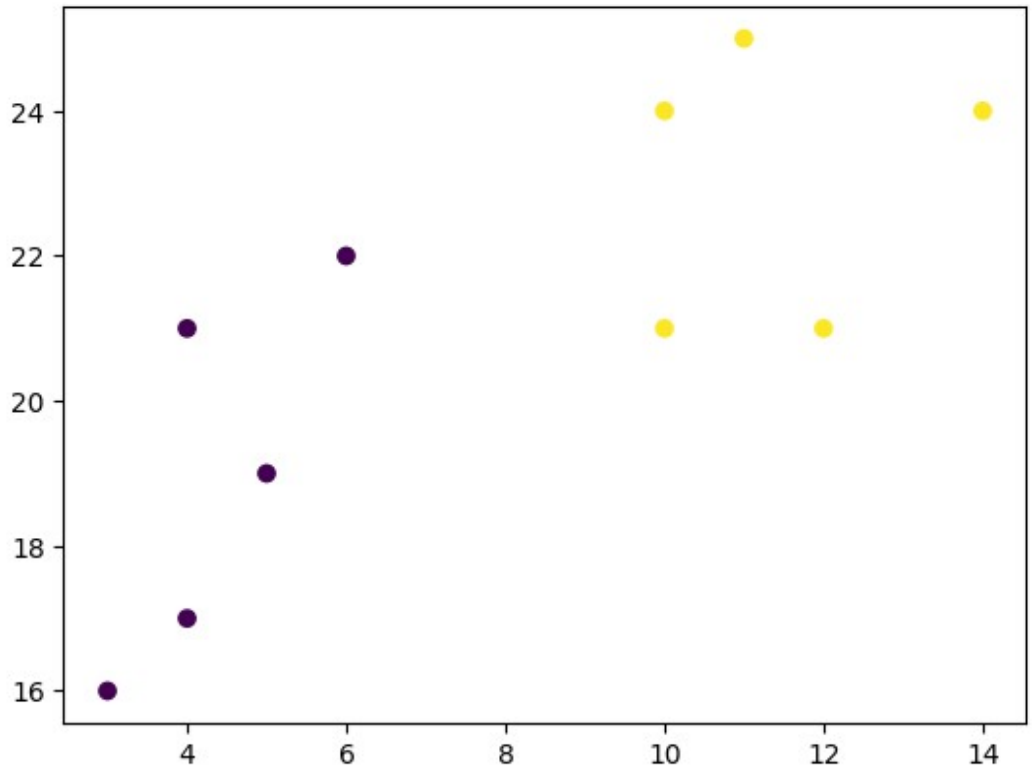
```
#Importing Modules
from sklearn import datasets
import matplotlib.pyplot as plt
# Loading dataset
iris_df = datasets.load_iris()
# Available methods on dataset
print("Methods:\n",dir(iris_df))
# Features
print("\nFeatures:\n",iris_df.feature_names)
# Targets
print("\nTargets: \n", iris_df.target)
#Target Names
print("\nTarget names:\n",iris_df.target_names)
label = {0: 'red', 1: 'blue', 2: 'green'}
# Dataset Slicing
x_axis = iris_df.data[:, 0] # Sepal Length
y_axis = iris_df.data[:, 2] # Sepal Width
# Plotting
plt.scatter(x_axis, y_axis, c=iris_df.target)
plt.show()
```

```
Methods:
 ['DESCR', 'data', 'data_module', 'feature_names', 'filename',
'frame', 'target', 'target_names']

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

Targets:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]

Target names:
 ['setosa' 'versicolor' 'virginica']
```




K-Means

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
x = [4, 5, 10, 4, 3, 11, 14, 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
data = list(zip(x, y))
kmeans = KMeans(n_clusters=2)
kmeans.fit(data)
plt.scatter(x, y, c=kmeans.labels_)
plt.show()
```




Model Evaluation

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs
X, y = make_blobs(random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)
logreg = LogisticRegression().fit(X_train, y_train)
print("Test set score: {:.2f}".format(logreg.score(X_test, y_test)))

Test set score: 0.88

from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
X, y = make_blobs(random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)
clf_model=LogisticRegression()
clf_model.fit(X_train, y_train)
y_pred = clf_model.predict(X_test)
print("Accuracy: ", accuracy_score(y_test, y_pred))
```

```
Accuracy:  0.9393939393939394
```

Engineering Features

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'Age': [42, 15, 67, 55, 1, 29, 75, 89, 4,
10, 15, 38, 22, 77]})
print("Before: \n")
print(df)
df['Label'] = pd.cut(x=df['Age'], bins=[0, 3, 17, 63, 99],
labels=['Baby/Toddler', 'Child', 'Adult',
'Elderly'])
print("After: \n")
print(df)
print("Categories: \n")
print(df['Label'].value_counts())
```

```
Before:

    Age
0    42
1    15
2    67
3    55
4     1
5    29
6    75
7    89
8     4
9    10
10   15
11   38
12   22
13   77
After:

    Age         Label
0    42         Adult
1    15         Child
```

```
2    67       Elderly
3    55         Adult
4     1  Baby/Toddler
5    29         Adult
6    75       Elderly
7    89       Elderly
8     4         Child
9    10         Child
10   15         Child
11   38         Adult
12   22         Adult
13   77       Elderly
Categories:

Label
Adult           5
Child           4
Elderly         4
Baby/Toddler    1
Name: count, dtype: int64
```

```
# Importing pandas and numpy libraries
import pandas as pd
import numpy as np
# Creating a dummy DataFrame of 12 numbers randomly
# ranging from 150-180 for height
df = pd.DataFrame({'Height': [150.4, 157.6, 170, 176, 164.2, 155,
159.2, 175, 162.4, 176, 153, 170.9]})
# Printing DataFrame Before Sorting Continuous to Categories
print("Before: ")
print(df)
# A column of name 'Label' is created in DataFrame
# Categorizing Height into 3 Categories
# Short: (150,157], 150 is excluded & 157 is included
# Average: (157,169], 157 is excluded & 169 is included
# Tall: (169,180], 169 is excluded & 180 is included.
df['Label'] = pd.cut(x=df['Height'],
bins=[150, 157, 169, 180],
labels=['Short', 'Average', 'Tall'])
# Printing Data Frame After Sorting Continuous to Categories
print("After: ")
print(df)
# Check the number of values in each bin
print("Categories: ")
print(df['Label'].value_counts())
```

```
Before:
    Height
0    150.4
1    157.6
```

```
2    170.0
3    176.0
4    164.2
5    155.0
6    159.2
7    175.0
8    162.4
9    176.0
10   153.0
11   170.9
After:
    Height    Label
0    150.4    Short
1    157.6  Average
2    170.0     Tall
3    176.0     Tall
4    164.2  Average
5    155.0    Short
6    159.2  Average
7    175.0     Tall
8    162.4  Average
9    176.0     Tall
10   153.0    Short
11   170.9     Tall
Categories:
Label
Tall       5
Average    4
Short      3
Name: count, dtype: int64
```