2. SCIKIT LEARN

Importing Datasets from sklearn package

```
1 from google.colab import drive
2  drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
1 import sklearn
2 from sklearn import datasets
3
3 dir(datasets)  #---displays all the datasets in the 'dataset' package of sklearn
```

```
'_california_housing',
'_covtype',
'_kddcup99',
'_lfw',
'_olivetti_faces',
'_openml',
'_rcv1',
'_samples_generator',
'_species_distributions',
'_svmlight_format_fast',
'_svmlight_format_io',
'_twenty_newsgroups',
'clear_data_home',
'dump_svmlight_file',
'fetch_20newsgroups',
'fetch_20newsgroups_vectorized',
'fetch_california_housing',
'fetch_covtype',
'fetch_kddcup99',
'fetch_lfw_pairs',
'fetch_lfw_people',
'fetch_olivetti_faces',
'fetch_openml',
'fetch_rcv1',
'fetch_species_distributions',
'get_data_home',
'load_boston',
'load_breast_cancer',
'load_diabetes',
'load_digits',
'load_files',
'load_iris',
'load_linnerud',
'load_sample_image',
'load_sample_images',
'load_svmlight_file',
'load_svmlight_files',
'load_wine',
'make_biclusters',
'make_blobs',
'make_checkerboard',
'make_circles',
'make_classification',
'make_friedman1',
'make_friedman2',
'make_friedman3',
'make_gaussian_quantiles',
'make_hastie_10_2',
'make_low_rank_matrix',
'make_moons',
'make_multilabel_classification',
'make_regression',
'make_s_curve',
'make_sparse_coded_signal',
'make_sparse_spd_matrix',
'make_sparse_uncorrelated',
'make_spd_matrix',
'make_swiss_roll']
```

Load Dataset

```
1 iterate = datasets.load_wine()
2 print(type(iterate))
```

```
3 print(iterate)
```

```
<class 'sklearn.utils.Bunch'>
{'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]]), '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, 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, 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]), 'frame': None, 'target_names': array(['class_0', 'class_1', 'class_2'], dtype='<U7'), 'DESCR': '.. _wine_dataset:\n\
```

```
1 i=datasets.load_iris()  #---Load the dataset
2 print(type(i))
4 print(i)
```

```
       [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, 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, 2, 2, 2, 2,
       2, 2]), 'frame': None, 'target_names': array(['setosa', 'versicolor', 'vi
```

Feature names/column names of the dataset

```
features=iterate.feature_names  #---fetch the feature names or the column names
print(features)
```

```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proantho
```

Print the loaded dataset

```
print(i.data) #---print the loaded dataset feature matrix
```

```
[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=i.target  #---gets the labels associated with the data points
print(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 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 2 2
 2 2]
```

```
1 print(i.target_names)  #--displays the target names associated with values 0 and 1 and 2
```

```
['setosa' 'versicolor' 'virginica']
```

```
1 print(i.DESCR)  #--gives all the detailed description about the dataset
```

```
**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature. Fisher's paper is a classic in the field and
is referenced frequently to this day. (See Duda & Hart, for example.) The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant. One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

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

Download any dataset from openml repository

```
1 from sklearn.datasets import fetch_openml
2 mice=fetch_openml(name='miceprotein',version=4)
3 mice
```

```
Gardiner KJ, Cios KJ (2015) Self-Organizing Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down
Syndrome. PLoS ONE 10(6): e0129126.\n\nExpression levels of 77 proteins measured in the cerebral cortex of 8 classes of control
and Down syndrome mice exposed to context fear conditioning, a task used to assess associative learning.\n\nThe data set
consists of the expression levels of 77 proteins/protein modifications that produced detectable signals in the nuclear fraction
of cortex. There are 38 control mice and 34 trisomic mice (Down syndrome), for a total of 72 mice. In the experiments, 15
measurements were registered of each protein per sample/mouse. Therefore, for control mice, there are 38x15, or 570
measurements, and for trisomic mice, there are 34x15, or 510 measurements. The dataset contains a total of 1080 measurements per
protein. Each measurement can be considered as an independent sample/mouse. \n\nThe eight classes of mice are described based on
features such as genotype, behavior and treatment. According to genotype, mice can be control or trisomic. According to
behavior, some mice have been stimulated to learn (context-shock) and others have not (shock-context) and in order to assess the
effect of the drug memantine in recovering the ability to learn in trisomic mice, some mice have been injected with the drug and
others have not. \n\nClasses: \n```\n* c-CS-s: control mice, stimulated to learn, injected with saline (9 mice) \n* c-CS-m:
```

control mice, stimulated to learn, injected with memantine (10 mice) \n* c-SC-s: control mice, not stimulated to learn, injected with saline (9 mice) \n* c-SC-m: control mice, not stimulated to learn, injected with memantine (10 mice) \n* t-CS-s: trisomy mice, stimulated to learn, injected with saline (7 mice) \n* t-CS-m: trisomy mice, stimulated to learn, injected with memantine (9 mice) \n* t-SC-s: trisomy mice, not stimulated to learn, injected with saline (9 mice) \n* t-SC-m: trisomy mice, not stimulated to learn, injected with memantine (9 mice) \n```\n\nThe aim is to identify subsets of proteins that are discriminant between the classes. \n\n### Attribute Information:\n\n```\n1 Mouse ID \n2..78 Values of expression levels of 77 proteins; the names of proteins are followed by â€œ_nâ€\x9d indicating that they were measured in the nuclear fraction. For example: DYRK1A_n \n79 Genotype: control (c) or trisomy (t) \n80 Treatment type: memantine (m) or saline (s) \n81 Behavior: context-shock (CS) or shock-context (SC) \n82 Class: c-CS-s, c-CS-m, c-SC-s, c-SC-m, t-CS-s, t-CS-m, t-SC-s, t-SC-m \n```\n\n### Relevant Papers:\n\nHiguera C, Gardiner KJ, Cios KJ (2015) Self-Organizing Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down Syndrome. PLoS ONE 10(6): e0129126. [Web Link] journal.pone.0129126 \n\nAhmed MM, Dhanasekaran AR, Block A, Tong S, Costa ACS, Stasko M, et al. (2015) Protein Dynamics Associated with Failed and Rescued Learning in the Ts65Dn Mouse Model of Down Syndrome. PLoS ONE 10(3): e0119491.\n\nDownloaded from openml.org.', 'details': {'id': '40966',
  'name': 'MiceProtein',
  'version': '4',
  'description_version': '1',
  'format': 'ARFF',
  'upload_date': '2017-11-08T16:00:15',
  'licence': 'Public',
  'url': 'https://api.openml.org/data/v1/download/17928620/MiceProtein.arff',
  'parquet_url': 'http://openml1.win.tue.nl/dataset40966/dataset_40966.pq',
  'file_id': '17928620',
  'default_target_attribute': 'class',
  'row_id_attribute': 'MouseID',
  'ignore_attribute': ['Genotype', 'Treatment', 'Behavior'],
  'tag': ['OpenML-CC18', 'study_135', 'study_98', 'study_99'],
  'visibility': 'public',
  'minio_url': 'http://openml1.win.tue.nl/dataset40966/dataset_40966.pq',
  'status': 'active',
  'processing_date': '2018-10-04 00:49:58',
  'md5_checksum': '3c479a6885bfa0438971388283a1ce32'}, 'url': 'https://www.openml.org/d/40966'}

Suppose data is of Excel, Jason, SQL, CSV file-then its best to use PANDAS libraryIf

the file is binary like .mat then its recomended to use scipy library

Polynomial data-numpy array

Image and video-load it in numpy array using skimage library