Introduction

The objective of this colab is to demonstrate `sklearn` dataset API.

Recall that it has three APIs:

1. Loaders (`load_*`) load small standard datasets bundled with `sklearn`.
2. Fetchers (`fetch_*`) fetch large datasets from the internet and loads them in memory.
3. Generators (`generate_*`) generate controlled synthetic datasets.

Loaders and fetchers return a `bunch` object and generators return a tuple of feature matrix and label vector (or matrix).

Loaders

Loading iris dataset

```
from sklearn.datasets import load_iris
data = load_iris()
```

This returns a `Bunch` object `data` which is a dictionary like object with the following attributes:

- `data`, which has the feature matrix.
- `target`, which is the label vector
- `feature_names` contain the names of the features.
- `target_names` contain the names of the classes.
- `DESCR` has the full description of dataset.
- `filename` has the path to the location of data.

```
type(data)
```

```
sklearn.utils.Bunch
```

We can access them one by one and examine their contents. For example, we can access `feature_names` as follows:

```
data.feature_names
```

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

We can see the names of the features in this dataset.

Let's examine the names of the labels.

```
data.target_names
```

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

There are three classes: `setosa`, `versicolor`, `virginica`.

The feature matrix can be accessed as follows: `data.data`. Let's look at the first five examples in feature matrix.

```
data.data[:5]
```

```
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]])
```

We can observe 4 features per example.

Let's examine the shape of the feature matrix.

```
data.data.shape
```

```
(150, 4)
```

There are 150 examples and each example has 4 features.

Finally, we will examine the label vector and its shape.

```
data.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, 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])
```

There are 50 examples each from three classes: 0, 1 and 2.

We can read additional documentation about `load_iris` in the following manner:

```
?load_iris
```

```
Object `load_iris` not found.
```

In this way, we can load and examine different datasets.

We can obtain feature matrix and label or target from `load_iris` and other loaders in general by setting `return_X_y` argument to `True`.

```
feature_matrix, label_vector = load_iris(return_X_y=True)
print ('Shape of feature matrix:', feature_matrix.shape)
print ('Shape of label vector:', label_vector.shape)
```

```
Shape of feature matrix: (150, 4)
Shape of label vector: (150,)
```

Loading diabetes dataset

```
from sklearn.datasets import load_diabetes
data = load_diabetes()
```

Additional details about this loader can be accessed from the documentation.

```
?load_diabetes
```

`load_diabetes`

Step 2. Load the dataset and obtain a `Bunch` object.

```
# Call the loader and obtain the `Bunch` object.
```

Step 3. Examine the bunch object.

Look at the description of the dataset.

```
data.feature_names
```

```
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

Find out the shape of the feature matrix.

```
# Write code for finding the shape of the feature matrix.
```

Look at the first five examples from the feature matrix.

```
# Look at the first five examples from the feature matrix.
```

Find out the shape of the label matrix.

```
# Write code to find shape of label matrix.
```

Look at the labels of the first five examples.

```
# Look at the labels of the first five examples.
```

Find out the names of the features.

```
# Get the names of the features.
```

Find names of class labels.

```
# Find names of class labels.
```

Loading digits dataset

```
from sklearn.datasets import load_digits
```

```
?load_digits
```

```
data = load_digits()
```

load_digits

Step 2. Load the dataset and obtain a `Bunch` object.

```
# Call the loader and obtain the `Bunch` object.
```

Step 3. Examine the bunch object.

Look at the description of the dataset.

Find out the shape of the feature matrix.

```
# Write code for finding the shape of the feature matrix.
```

Look at the first five examples from the feature matrix.

```
# Look at the first five examples from the feature matrix.
```

Find out the shape of the label matrix.

```
# Write code to find shape of label matrix.
```

Look at the labels of the first five examples.

```
# Look at the labels of the first five examples.
```

Find out the names of the features.

```
# Get the names of the features.
```

Find names of class labels.

```
# Find names of class labels.
```

Exercise

Experiment with other dataset loaders e.g. `load_wine`, `load_breast_cancer` and `load_linnerud`.

`load_wine`

Step 1. Import the loader.

```
# Write your code here.
from sklearn.datasets import load_wine
```

Step 1a. In case, you want to know more about the loader, access its documentation by using `?' command.

```
# Access the documentation.
?load_wine
```

Step 2. Load the dataset and obtain a `Bunch` object.

```
# Call the loader and obtain the `Bunch` object.
data = load_wine()
```

Step 3. Examine the bunch object.

Look at the description of the dataset.

```
data.DESCR
```

```
'.. _wine_dataset:\n\nWine recognition dataset\n------------------------\n\n**Data Set
Characteristics:**\n\n    :Number of Instances: 178 (50 in each of three classes)\n
:Number of Attributes: 13 numeric, predictive attributes and the class\n    :Attribute
Information:\n \t\t- Alcohol\n \t\t- Malic acid\n \t\t- Ash\n\t\t- Alcalinity of ash
\n \t\t- Magnesium\n\t\t- Total phenols\n \t\t- Flavanoids\n \t\t- Nonflavanoid phenols
\n \t\t- Proanthocyanins\n\t\t- Color intensity\n \t\t- Hue\n \t\t- OD280/OD315 of dilu
ted wines\n \t\t- Proline\n\n    - class:\n            - class_0\n            - class_1
\n            - class_2\n\t\t\n    :Summary Statistics:\n    \n    --------------------
```

Find out the shape of the feature matrix.

```
# Write code for finding the shape of the feature matrix.
data.data.shape
```

```
(178, 13)
```

Look at the first five examples from the feature matrix.

```
# Look at the first five examples from the feature matrix.
data.data[:5]
```

```
array([[1.423e+01, 1.710e+00, 2.430e+00, 1.560e+01, 1.270e+02, 2.800e+00,
        3.060e+00, 2.800e-01, 2.290e+00, 5.640e+00, 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, 1.120e+01, 1.000e+02, 2.650e+00,
        2.760e+00, 2.600e-01, 1.280e+00, 4.380e+00, 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, 1.860e+01, 1.010e+02, 2.800e+00,
        3.240e+00, 3.000e-01, 2.810e+00, 5.680e+00, 1.030e+00, 3.170e+00,
        1.185e+03],
       [1.437e+01, 1.950e+00, 2.500e+00, 1.680e+01, 1.130e+02, 3.850e+00,
        3.490e+00, 2.400e-01, 2.180e+00, 7.800e+00, 8.600e-01, 3.450e+00,
        1.480e+03],
       [1.324e+01, 2.590e+00, 2.870e+00, 2.100e+01, 1.180e+02, 2.800e+00,
        2.690e+00, 3.900e-01, 1.820e+00, 4.320e+00, 1.040e+00, 2.930e+00,
        7.350e+02]])
```

Find out the shape of the label matrix.

```
# Write code to find shape of label matrix.
data.target.shape
```

```
(178,)
```

Look at the labels of the first five examples.

```
# Look at the labels of the first five examples.
data.target[:5]
```

```
array([0, 0, 0, 0, 0])
```

Find out the names of the features.

```
# Get the names of the features.
data.feature_names
```

```
['alcohol',
 'malic_acid',
 'ash',
 'alcalinity_of_ash',
 'magnesium',
 'total_phenols',
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
 'od280/od315_of_diluted_wines',
 'proline']
```

Find names of class labels.

```
# Find names of class labels.
data.target_names
```

```
array(['class_0', 'class_1', 'class_2'], dtype='<U7')
```

`load_breast_cancer`

Step 1. Import the loader.

```
# Write your code here.
```

Step 1a. In case, you want to know more about the loader, access its documentation by using `?' command.

```
# Access the documentation.
```

Step 2. Load the dataset and obtain a `Bunch` object.

```
# Call the loader and obtain the `Bunch` object.
```

Step 3. Examine the bunch object.

Look at the description of the dataset.

Find out the shape of the feature matrix.

```
# Write code for finding the shape of the feature matrix.
```

Look at the first five examples from the feature matrix.

```
# Look at the first five examples from the feature matrix.
```

Find out the shape of the label matrix.

```
# Write code to find shape of label matrix.
```

Look at the labels of the first five examples.

```
# Look at the labels of the first five examples.
```

Find out the names of the features.

```
# Get the names of the features.
```

Find names of class labels.

```
# Find names of class labels.
```

`load_linnerud`

Step 1. Import the loader.

```
# Write your code here.
```

Step 1a. In case, you want to know more about the loader, access its documentation by using `?' command.

```
# Access the documentation.
```

Step 2. Load the dataset and obtain a `Bunch` object.

```
# Call the loader and obtain the `Bunch` object.
```

Step 3. Examine the bunch object.

Look at the description of the dataset.

```

```

Find out the shape of the feature matrix.

```
# Write code for finding the shape of the feature matrix.
```

Look at the first five examples from the feature matrix.

```
# Look at the first five examples from the feature matrix.
```

Find out the shape of the label matrix.

```
# Write code to find shape of label matrix.
```

Look at the labels of the first five examples.

```
# Look at the labels of the first five examples.
```

Find out the names of the features.

```
# Get the names of the features.
```

Find names of class labels.

```
# Find names of class labels.
```

Fetchers

fetch_california_housing

Step 1: Import the library and access the documentation.

```
from sklearn.datasets import fetch_california_housing
?fetch_california_housing
```

Note that the `fetch_*` also returns a `Bunch` object just like loaders.

We can examine various attributes of this dataset on the lines of datasets in loaders.

Step 2. Load the dataset and obtain a `Bunch` object.

```
# Call the loader and obtain the `Bunch` object.
housing_data = fetch_california_housing()
```

Step 3. Examine the bunch object.

Look at the description of the dataset.

```
housing_data.DESCR
```

```
'.. _california_housing_dataset:\n\nCalifornia Housing dataset\n-----------------------
---\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 20640\n\n    :Number
of Attributes: 8 numeric, predictive attributes and the target\n\n    :Attribute Inform
ation:\n        - MedInc        median income in block group\n        - HouseAge      m
edian house age in block group\n        - AveRooms      average number of rooms per hou
sehold\n        - AveBedrms     average number of bedrooms per household\n        - Pop
ulation    block group population\n        - AveOccup      average number of household
members\n        - Latitude      block group latitude\n        - Longitude     block gr
```

Find out the shape of the feature matrix.

```
# Write code for finding the shape of the feature matrix.
housing_data.data.shape
```

```
(20640, 8)
```

Look at the first five examples from the feature matrix.

```
# Look at the first five examples from the feature matrix.
housing_data.data[:5]
```

```
array([[ 8.32520000e+00,  4.10000000e+01,  6.98412698e+00,
         1.02380952e+00,  3.22000000e+02,  2.55555556e+00,
         3.78800000e+01, -1.22230000e+02],
       [ 8.30140000e+00,  2.10000000e+01,  6.23813708e+00,
         9.71880492e-01,  2.40100000e+03,  2.10984183e+00,
         3.78600000e+01, -1.22220000e+02],
       [ 7.25740000e+00,  5.20000000e+01,  8.28813559e+00,
         1.07344633e+00,  4.96000000e+02,  2.80225989e+00,
         3.78500000e+01, -1.22240000e+02],
       [ 5.64310000e+00,  5.20000000e+01,  5.81735160e+00,
         1.07305936e+00,  5.58000000e+02,  2.54794521e+00,
         3.78500000e+01, -1.22250000e+02],
       [ 3.84620000e+00,  5.20000000e+01,  6.28185328e+00,
         1.08108108e+00,  5.65000000e+02,  2.18146718e+00,
         3.78500000e+01, -1.22250000e+02]])
```

Find out the shape of the label matrix.

```
# Write code to find shape of label matrix.
housing_data.target.shape
```

```
(20640,)
```

Look at the labels of the first five examples.

```
# Look at the labels of the first five examples.
housing_data.target[:5]
```

```
array([4.526, 3.585, 3.521, 3.413, 3.422])
```

Note that the labels seem to be real numbers.

Find out the names of the features.

```
# Get the names of the features.
housing_data.feature_names
```

```
['MedInc',
 'HouseAge',
 'AveRooms',
 'AveBedrms',
 'Population',
 'AveOccup',
     'Latitude',
     'Longitude']
```

Find names of class labels.

```
# Find names of class labels.
housing_data.target_names
```

```
['MedHouseVal']
```

`fetch_openml`

openml.org is a public repository for machine learning data and experiments, that allows everybody to upload open datasets.

Import the library and access the documentation.

```
from sklearn.datasets import fetch_openml
?fetch_openml
```

Note that this is an experimental API and is likely to change in the future releases.

> We use this API for loading MNIST dataset.

```
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
print ("Feature matrix shape:", X.shape)
print ("Label shape:", y.shape)
```

```
Feature matrix shape: (70000, 784)
Label shape: (70000,)
```

Exercise

`fetch_20newsgroups`

Step 1. Import the loader.

```
# Write your code here.
```

Step 1a. In case, you want to know more about the loader, access its documentation by using `?' command.

```
# Access the documentation.
```

Step 2. Load the dataset and obtain a `Bunch` object.

```
# Call the loader and obtain the `Bunch` object.
```

Step 3. Examine the bunch object.

Look at the description of the dataset.

```

```

Find out the shape of the feature matrix.

```
# Write code for finding the shape of the feature matrix.
```

Look at the first five examples from the feature matrix.

```
# Look at the first five examples from the feature matrix.
```

Find out the shape of the label matrix.

```
# Write code to find shape of label matrix.
```

Look at the labels of the first five examples.

```
# Look at the labels of the first five examples.
```

Find out the names of the features.

```
# Get the names of the features.
```

Find names of class labels.

```
# Find names of class labels.
```

`fetch_kddcup99`

Step 1. Import the loader.

```
# Write your code here.
```

Step 1a. In case, you want to know more about the loader, access its documentation by using `?' command.

```
# Access the documentation.
```

Step 2. Load the dataset and obtain a `Bunch` object.

```
# Call the loader and obtain the `Bunch` object.
```

Step 3. Examine the bunch object.

Look at the description of the dataset.

```

```

Find out the shape of the feature matrix.

```
# Write code for finding the shape of the feature matrix.
```

Look at the first five examples from the feature matrix.

```
# Look at the first five examples from the feature matrix.
```

Find out the shape of the label matrix.

```
# Write code to find shape of label matrix.
```

Look at the labels of the first five examples.

```
# Look at the labels of the first five examples.
```

Find out the names of the features.

```
# Get the names of the features.
```

Find names of class labels.

```
# Find names of class labels.
```

Generators

`make_regression`

```
from sklearn.datasets import make_regression
?make_regression
```

Example 1

Let's generate 100 samples with 5 features for a single label regression problem.

```
X, y = make_regression(n_samples=100, n_features=5, n_targets=1, shuffle=True, random_stat
```

It's a good practice to set seed so that we get to see repeatability in the experimentation.

Let's look at the shapes of feature matrix and label vector.

```
X.shape
```

```
(100, 5)
```

```
y.shape
```

```
(100,)
```

Example 2

Let's generate 100 samples with 5 features for multiple regression problem with 5 outputs.

```
X, y = make_regression(n_samples=100, n_features=5, n_targets=5, shuffle=True, random_stat
```

Let's look at the shapes of feature matrix and label vector.

```
X.shape
```

```
(100, 5)
```

```
y.shape
```

```
(100, 5)
```

Since we generated multi-output target with 5 outputs, the output has shape `(100, 5)`.

`make_classification`

Generate a random n-class classification problem set up.

```
from sklearn.datasets import make_classification
?make_classification
```

Let's generate a binary classification problem with 10 features and 100 samples.

```
X, y = make_classification(n_samples=100, n_features=10, n_classes=2, n_clusters_per_class
```

Let's examine the shapes of feature matrix and label vector.

```
X.shape
```

```
(100, 10)
```

```
y.shape
```

```
(100,)
```

Look at a few examples and their labels.

```
X[:5]
```

```
array([[ 0.11422765, -1.71016839, -0.06822216, -0.14928517,  0.30780177,
         0.15030176, -0.05694562, -0.22595246, -0.36361221, -0.13818757],
       [ 0.70775194, -1.57022472, -0.23503183, -0.63604713,  0.62180996,
        -0.56246678,  0.97255445, -0.77719676,  0.63240774, -0.47809669],
       [ 0.63859246,  0.04739867,  0.33273433,  1.1046981 , -0.65183611,
        -1.66152006, -1.2110162 ,  1.09821151, -0.0660798 ,  0.68024225],
       [-0.23894805, -0.97755524,  0.0379061 ,  0.19896733,  0.50091719,
        -0.90756366,  0.75539123,  0.12437227, -0.57677133,  0.07871283],
       [-0.59239392, -0.05023811,  0.17573204, -1.43949185,  0.27045683,
        -0.86399077, -0.83095012,  0.60046915,  0.04852163,  0.32557953]])
```

```
y[:5]
```

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

Let's generate a three class classification problem with 100 samples and 10 features.

```
X, y = make_classification(n_samples=100, n_features=10, n_classes=3, n_clusters_per_class
```

Let's examine shapes of feature matrix and labels.

```
X.shape
```

```
(100, 10)
```

```
y.shape
```

```
(100,)
```

Let's look at a few examples - features and labels.

```
X[:5]
```

```
array([[-0.58351628, -1.73833907, -1.37298251, -1.77311485,  0.45918008,
         0.83392215, -1.66096093,  0.20768769, -0.07016571,  0.42961822],
       [-1.0044394 , -1.43862044,  0.47335819, -0.21188291,  0.0125924 ,
         0.22409248, -0.77300978,  0.49799829,  0.0976761 ,  0.02451017],
       [ 0.07740833,  0.19896733,  0.12437227,  0.17738132, -0.97755524,
         0.50091719,  0.75138712,  0.54336019,  0.09933231, -1.66940528],
       [-0.91759569, -0.9609536 ,  1.07746664,  0.4522739 , -0.32138584,
        -0.8254972 , -0.56372455,  0.24368721,  0.41293145, -0.8222204 ],
       [-0.96222828, -0.96090774,  1.21530116,  0.55980482, -1.24778318,
        -0.25256815, -1.43014138,  0.13074058,  1.6324113 , -0.44004449]])
```

```
y[:5]
```

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

make_multilabel_classification

This function helps us generating a random multi-label classification problem.

```
from sklearn.datasets import make_multilabel_classification
?make_multilabel_classification
```

Let's generate a multilabel classification problem with 100 samples, 10 features, 5 labels and on an average 2 labels per example.

```
X, y = make_multilabel_classification(n_samples=100, n_features=20, n_classes=5, n_labels=
```

First of all, let's examine shapes of feature matrix and label vector.

```
X.shape
```

```
(100, 20)
```

```
y.shape
```

```
(100, 5)
```

Let's examine a few rows of feature matrix and label matrix.

```
X[:5]
```

```
array([[ 1.,  4.,  2.,  0.,  0.,  2.,  2.,  3.,  4.,  3.,  5.,  0.,  2.,
         5.,  3.,  1.,  1.,  0.,  2.,  7.],
       [ 4.,  1.,  2.,  0.,  3.,  1.,  2.,  2.,  2.,  2.,  1.,  1.,  1.,
         3.,  1.,  2.,  4.,  2.,  2.,  2.],
       [ 0.,  1.,  4.,  0.,  2.,  1.,  4.,  0.,  6.,  2.,  4.,  2.,  1.,
         0.,  5.,  0.,  5.,  5.,  1.,  7.],
       [ 5.,  3.,  3.,  0.,  0.,  2.,  6.,  2., 10.,  0.,  2.,  2.,  2.,
         0.,  4.,  0.,  5.,  5.,  5.,  3.],
       [ 4.,  3.,  5.,  0.,  4.,  2.,  6.,  1.,  2.,  2.,  3.,  1.,  4.,
         1.,  5.,  4.,  3.,  4.,  2.,  1.]])
```

```
y[:5]
```

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

make_blobs

`make_blobs` enables us to generate random data for clustering.

```
from sklearn.datasets import make_blobs
?make_blobs
```

Let's generate a random dataset of 10 samples with 2 features each for clustering.

```
X, y = make_blobs(n_samples=10, centers=3, n_features=2, random_state=42)
print ("Feature matrix shape:", X.shape)
print ("Label shape:", y.shape)
```

```
Feature matrix shape: (10, 2)
Label shape: (10,)
```

We can find the cluster membership of each point in `y`.

```
y
```

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