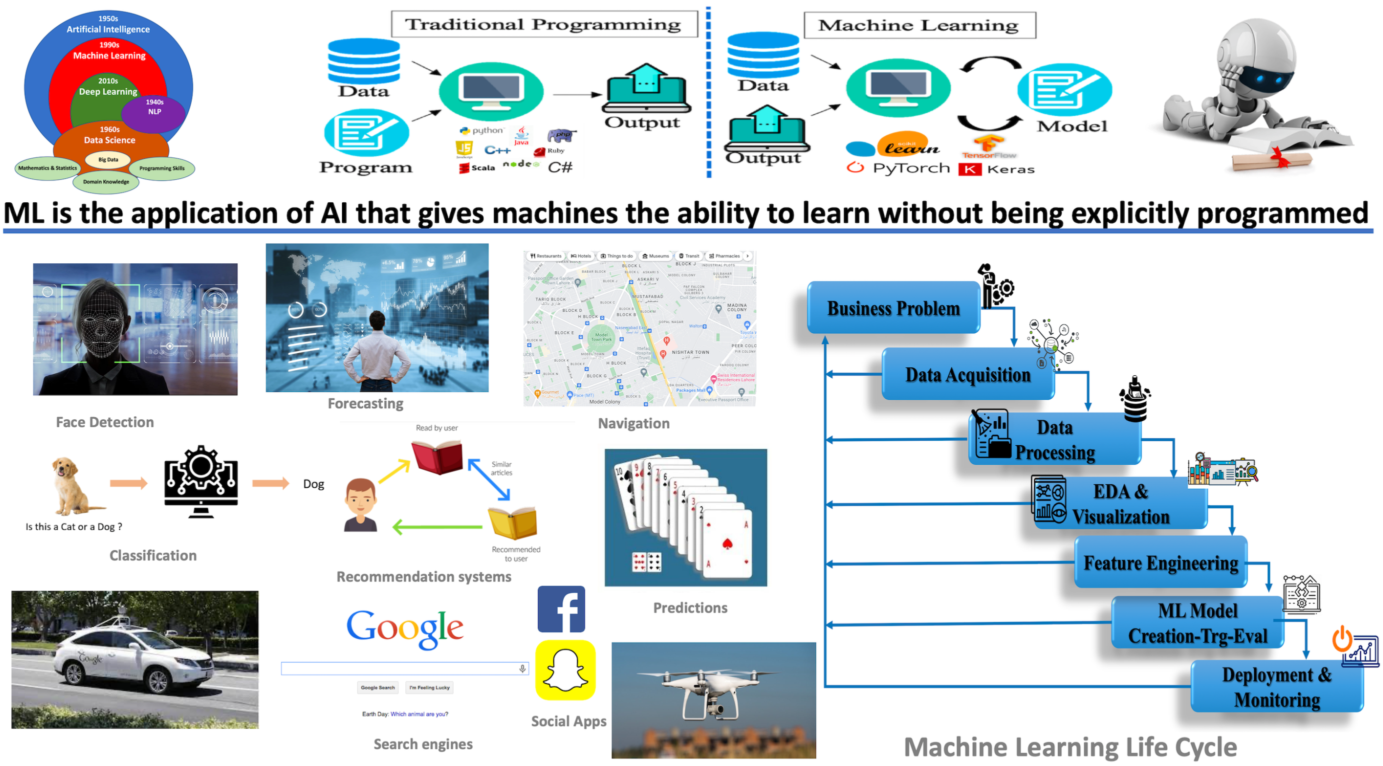
Department of Data Science

Course: Tools and Techniques for Data Science

Instructor: Muhammad Arif Butt, Ph.D.

Lecture 6.5 (Overview of Scikit-Learn Library)

Open in Colab
[(https://colab.research.google.com/github/arifpucit/data-science/blob/master/Section-4-Mathematics-for-Data-Science/Lec-4.1(Descriptive-Statistics).ipynb)](https://colab.research.google.com/github/arifpucit/data-science/blob/master/Section-4-Mathematics-for-Data-Science/Lec-4.1(Descriptive-Statistics).ipynb)



Learning agenda of this notebook

1. Overview of Scikit-Learn
    - What is Scikit-Learn (sklearn)?
    - Download and Install Scikit-Learn
    - Sklearn Built-in Datasets
        - Toy Datasets
        - Real World Datasets

- Downloading Datasets from Public ML Repositories
- Random Sample Generators

1. Overview of Scikit-Learn

a. What is Scikit-Learn (sklearn) ?

- Scikit-Learn is a Python Machine Learning library that provides a uniform framework for a wide variety of Machine Learning models/algorithms/estimators to perform Classification, Regression and Clustering tasks.
- Scikit-Learn also comes with many convenience tools to perform feature scaling, Train-Test-Split, Cross Validation and a variety of reporting metric functions to analyze model performance.
- The name `Scikit` comes from SciPy and Toolkit, and it is built on top of NumPy, SciPy and Matplotlib.
- In Scikit-Learn all the machine learning algorithms are imported, fitted and used in a uniform fashion, thus allowing the users to apply almost any algorithm effectively without truly understanding what the algorithm is doing.
- So you don't really have to learn the math before you start to use machine learning models. It provides you an abstraction that lets us apply the benefits of machine learning to our problem straight away without going to understand the math behind the models.
- It is a more pragmatic industry style approach rather than an academic approach of describing the model and its parameters.

Visit: [https://scikit-learn.org/ (https://scikit-learn.org/)](https://scikit-learn.org/)

b. Download and Install Scikit-Learn

Source: [https://scikit-learn.org/stable/install.html (https://scikit-learn.org/stable/install.html)](https://scikit-learn.org/stable/install.html)

- For Scikit version1.1, you should have:
  - Python 3.8+
  - NumPY 1.17.3+
  - SciPY 1.3.2+
  - Matplotlib 3.1.2+

In [1]:

```
!python --version
```

```
Python 3.9.13
```

In [2]:

```
import numpy, scipy, matplotlib
numpy.__version__, scipy.__version__, matplotlib.__version__
```

Out[2]:

```
('1.23.1', '1.9.1', '3.5.2')
```

In [3]:

```
import sys
!{sys.executable} -m pip install scikit-learn -q
```

In [4]:

```
import sklearn
sklearn.__version__
```

Out[4]:

```
'1.0.2'
```

c. Scikit-Learn Built-in Data Sets

(i) Toy Datasets (`datasets.load_XXX()` methods)

[https://scikit-learn.org/stable/datasets/toy_dataset.html#toy-datasets (https://scikit-learn.org/stable/datasets/toy_dataset.html#toy-datasets)](https://scikit-learn.org/stable/datasets/toy_dataset.html#toy-datasets)

- Scikit-learn comes with a few small standard datasets that do not require to download any file from some external website.
- These datasets are useful to quickly illustrate the behavior of the various algorithms implemented in scikit-learn. They are however often too small to be representative of real world machine learning tasks.
- Don't be fooled by the word "toy". These datasets are powerful and serve as a strong starting point for learning ML
- These toy datasets are broadly categorised into two types:
- Regression
  - Boston house prices dataset
  - Diabetes dataset
  - Linnerud dataset (Multi-output Regression)
  -
- Classification
  - Iris plants dataset
  - Optical recognition of handwritten digits dataset
  - Wine recognition dataset
  - Breast cancer wisconsin (diagnostic) dataset¶

Diabetes Dataset:

In [5]:

```python
from sklearn.datasets import load_diabetes
```

In [6]:

```python
diabetes = load_diabetes(as_frame=True)
type(diabetes)
```

Out[6]:

```
sklearn.utils.Bunch
```

In [7]:

```python
diabetes.keys()
```

Out[7]:

```
dict_keys(['data', 'target', 'frame', 'DESCR', 'feature_names', 'data_
filename', 'target_filename', 'data_module'])
```

In [8]:

```python
# Access values associated with 'data' key of the bunch object (diabetes)
diabetes.get('data')
diabetes['data']
diabetes.data
```

Out[8]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.01
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.06
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.00
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.02
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.03
...	...	...	...	...	...	...	...	...	
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.03
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.01
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.04
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.04
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.00

442 rows × 10 columns

In [9]:

```
# Access values associated with 'target' key of the bunch object (diabetes)
diabetes.target
```

Out[9]:

```
0      151.0
1       75.0
2      141.0
3      206.0
4      135.0
       ...
437    178.0
438    104.0
439    132.0
440    220.0
441     57.0
Name: target, Length: 442, dtype: float64
```

In [10]:

```
# Access values associated with 'feature_names' key of the bunch object (diabete
diabetes.feature_names
```

Out[10]:

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

In [11]:

```
# Access values associated with 'data_filename' key of the bunch object (diabete
diabetes.data_filename
```

Out[11]:

```
'diabetes_data.csv.gz'
```

In [12]:

```
# Access values associated with 'target_filename' key of the bunch object (diabe
diabetes.target_filename
```

Out[12]:

```
'diabetes_target.csv.gz'
```

In [13]:

```python
# Access values associated with 'DESCR ' key of the bunch object (diabetes)
print(diabetes.DESCR)
```

```
.. _diabetes_dataset:

Diabetes dataset
----------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n
=
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

  :Number of Instances: 442

  :Number of Attributes: First 10 columns are numeric predictive value
s

  :Target: Column 11 is a quantitative measure of disease progression
one year after baseline

  :Attribute Information:
      - age     age in years
      - sex
      - bmi     body mass index
      - bp      average blood pressure
      - s1      tc, total serum cholesterol
      - s2      ldl, low-density lipoproteins
      - s3      hdl, high-density lipoproteins
      - s4      tch, total cholesterol / HDL
      - s5      ltg, possibly log of serum triglycerides level
      - s6      glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and s
caled by the standard deviation times `n_samples` (i.e. the sum of squ
ares of each column totals 1).

Source URL:
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html (https://www
4.stat.ncsu.edu/~boos/var.select/diabetes.html)

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (20
04) "Least Angle Regression," Annals of Statistics (with discussion),
407-499.
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
```

Do Explore all the toy datasets at your own time...

(ii) Real World Datasets (`datasets.fetch_XXX()` methods)

[https://scikit-learn.org/stable/datasets/real_world.html (https://scikit-learn.org/stable/datasets/real_world.html)](https://scikit-learn.org/stable/datasets/real_world.html)

- Scikit-learn provides tools to load larger datasets, downloading them if necessary.
- Regression
  - California Housing dataset
- Classification
  - The Olivetti faces dataset
  - The 20 newsgroups text dataset
  - The Labeled Faces in the Wild face recognition dataset
  - Forest covertypes
  - RCV1 dataset
  - Kddcup 99 dataset

California Housing Dataset:

In [14]:

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

In [15]:

```
california = fetch_california_housing(as_frame=True, download_if_missing=True)
type(california)
```

Out[15]:

```
sklearn.utils.Bunch
```

In [16]:

```
import pandas as pd
df = pd.DataFrame(california.data, columns=california.feature_names)
df['price'] = california.target
df.sample(10)
```

Out[16]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
6240	4.5417	33.0	5.708861	0.962025	1079.0	4.552743	34.06	-117.98
3403	5.2661	37.0	6.701887	1.135849	702.0	2.649057	34.26	-118.34
11028	5.3041	37.0	5.741597	0.966387	1378.0	2.894958	33.79	-117.84
17070	6.2007	4.0	5.769570	1.222712	1690.0	1.863286	37.53	-122.26
14491	10.0707	22.0	7.906818	1.018182	1252.0	2.845455	32.85	-117.24
19839	1.5714	39.0	3.830357	1.017857	1222.0	5.455357	36.52	-119.29
46	2.0260	50.0	3.700658	1.059211	616.0	2.026316	37.83	-122.26
1015	5.7907	20.0	6.408261	0.966640	3489.0	2.771247	37.67	-121.77
8550	3.2594	36.0	4.059585	0.937824	1054.0	2.730570	33.88	-118.35
3933	5.0150	37.0	5.578313	1.012048	826.0	3.317269	34.20	-118.58

In [17]:

```python
print(california.DESCR)
```

```
.. _california_housing_dataset:

California Housing dataset
--------------------------

**Data Set Characteristics:**

    :Number of Instances: 20640

    :Number of Attributes: 8 numeric, predictive attributes and the ta
rget

    :Attribute Information:
        - MedInc        median income in block group
        - HouseAge      median house age in block group
        - AveRooms      average number of rooms per household
        - AveBedrms     average number of bedrooms per household
        - Population    block group population
        - AveOccup      average number of household members
        - Latitude      block group latitude
        - Longitude     block group longitude

    :Missing Attribute Values: None

This dataset was obtained from the StatLib repository.
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html (https://
www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html)

The target variable is the median house value for California district
s,
expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per
census
block group. A block group is the smallest geographical unit for which
the U.S.
Census Bureau publishes sample data (a block group typically has a pop
ulation
of 600 to 3,000 people).

An household is a group of people residing within a home. Since the av
erage
number of rooms and bedrooms in this dataset are provided per househol
d, these
columns may take surpinsingly large values for block groups with few h
ouseholds
and many empty houses, such as vacation resorts.

It can be downloaded/loaded using the
:func:`sklearn.datasets.fetch_california_housing` function.

.. topic:: References

    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregression
s,
      Statistics and Probability Letters, 33 (1997) 291-297
```

In [18]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
 8   price       20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

Do Explore all the real world datasets at your own time...

(iii) Downloading Datasets from Public ML Repositories (`fetch_XXX()` and `fetch_openml()` methods)

- Kaggle: [https://www.kaggle.com/datasets (https://www.kaggle.com/datasets)](https://www.kaggle.com/datasets)
- UCI ML Repository: [https://archive.ics.uci.edu/ml/index.php (https://archive.ics.uci.edu/ml/index.php)](https://archive.ics.uci.edu/ml/index.php)
- OpenML Repository: [https://www.openml.org/ (https://www.openml.org/)](https://www.openml.org/)

In [19]:

```
from sklearn import datasets
print(dir(datasets))
```

```
['__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__lo
ader__', '__name__', '__package__', '__path__', '__spec__', '_base',
'_california_housing', '_covtype', '_kddcup99', '_lfw', '_olivetti_fac
es', '_openml', '_rcv1', '_samples_generator', '_species_distribution
s', '_svmlight_format_fast', '_svmlight_format_io', '_twenty_newsgroup
s', 'clear_data_home', 'data', 'descr', 'dump_svmlight_file', 'fetch_2
0newsgroups', 'fetch_20newsgroups_vectorized', 'fetch_california_housi
ng', '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_c
ancer', 'load_diabetes', 'load_digits', 'load_files', 'load_iris', 'lo
ad_linnerud', 'load_sample_image', 'load_sample_images', 'load_svmligh
t_file', 'load_svmlight_files', 'load_wine', 'make_biclusters', 'make_
blobs', 'make_checkerboard', 'make_circles', 'make_classification', 'm
ake_friedman1', 'make_friedman2', 'make_friedman3', 'make_gaussian_qua
ntiles', 'make_hastie_10_2', 'make_low_rank_matrix', 'make_moons', 'ma
ke_multilabel_classification', 'make_regression', 'make_s_curve', 'mak
e_sparse_coded_signal', 'make_sparse_spd_matrix', 'make_sparse_uncorre
lated', 'make_spd_matrix', 'make_swiss_roll']
```

In [20]:

```
titanic = datasets.fetch_openml(name='titanic', version=1)
type(titanic)
```

Out[20]:

```
sklearn.utils.Bunch
```

In [21]:

```
titanic.keys()
```

Out[21]:

```
dict_keys(['data', 'target', 'frame', 'categories', 'feature_names',
'target_names', 'DESCR', 'details', 'url'])
```

In [22]:

```
titanic.url
```

Out[22]:

```
'https://www.openml.org/d/40945'
```

In [23]:

```
titanic.details
```

Out[23]:


```
{'id': '40945',
 'name': 'Titanic',
 'version': '1',
 'description_version': '7',
 'format': 'ARFF',
 'upload_date': '2017-10-16T01:17:36',
 'licence': 'Public',
 'url': 'https://api.openml.org/data/v1/download/16826755/Titanic.arf
f',
 'parquet_url': 'http://openml1.win.tue.nl/dataset40945/dataset_40945.
pq',
 'file_id': '16826755',
 'default_target_attribute': 'survived',
 'tag': 'text_data',
 'visibility': 'public',
 'minio_url': 'http://openml1.win.tue.nl/dataset40945/dataset_40945.p
q',
 'status': 'active',
 'processing_date': '2018-10-04 07:19:36',
 'md5_checksum': '60ac7205eee0ba5045c90b3bba95b1c4'}
```

In [24]:

```python
print(titanic.DESCR)
```

```
**Author**: Frank E. Harrell Jr., Thomas Cason
**Source**: [Vanderbilt Biostatistics](http://biostat.mc.vanderbilt.ed
u/wiki/pub/Main/DataSets/titanic.html)
**Please cite**:

The original Titanic dataset, describing the survival status of indivi
dual passengers on the Titanic. The titanic data does not contain info
rmation from the crew, but it does contain actual ages of half of the
passengers. The principal source for data about Titanic passengers is
the Encyclopedia Titanica. The datasets used here were begun by a vari
ety of researchers. One of the original sources is Eaton & Haas (1994)
Titanic: Triumph and Tragedy, Patrick Stephens Ltd, which includes a p
assenger list created by many researchers and edited by Michael A. Fin
dlay.

Thomas Cason of UVa has greatly updated and improved the titanic data
frame using the Encyclopedia Titanica and created the dataset here. So
me duplicate passengers have been dropped, many errors corrected, many
missing ages filled in, and new variables created.

For more information about how this dataset was constructed:
http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic3info.t
xt (http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic3in
fo.txt)


### Attribute information

The variables on our extracted dataset are pclass, survived, name, ag
e, embarked, home.dest, room, ticket, boat, and sex. pclass refers to
passenger class (1st, 2nd, 3rd), and is a proxy for socio-economic cla
ss. Age is in years, and some infants had fractional values. The titan
ic2 data frame has no missing data and includes records for the crew,
but age is dichotomized at adult vs. child. These data were obtained f
rom Robert Dawson, Saint Mary's University, E-mail. The variables are
pclass, age, sex, survived. These data frames are useful for demonstra
ting many of the functions in Hmisc as well as demonstrating binary lo
gistic regression analysis using the Design library. For more details
and references see Simonoff, Jeffrey S (1997): The "unusual episode" a
nd a second statistics course. J Statistics Education, Vol. 5 No. 1.

Downloaded from openml.org.
```

In [25]:

```
df = pd.DataFrame(titanic.data, columns=titanic.feature_names)
df['target'] = titanic.target
df
```

Out[25]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	b
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	
2	1.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	N
3	1.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	N
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	N
...	...	...	...	...	...	...	...	...	...	...	
1304	3.0	Zabour, Miss. Hileni	female	14.5000	1.0	0.0	2665	14.4542	None	C	N
1305	3.0	Zabour, Miss. Thamine	female	NaN	1.0	0.0	2665	14.4542	None	C	N
1306	3.0	Zakarian, Mr. Mapriededer	male	26.5000	0.0	0.0	2656	7.2250	None	C	N
1307	3.0	Zakarian, Mr. Ortin	male	27.0000	0.0	0.0	2670	7.2250	None	C	N
1308	3.0	Zimmerman, Mr. Leo	male	29.0000	0.0	0.0	315082	7.8750	None	S	N

1309 rows × 14 columns

In [26]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype   
---  ------     --------------  -----   
 0   pclass     1309 non-null   float64 
 1   name       1309 non-null   object  
 2   sex        1309 non-null   category
 3   age        1046 non-null   float64 
 4   sibsp      1309 non-null   float64 
 5   parch      1309 non-null   float64 
 6   ticket     1309 non-null   object  
 7   fare       1308 non-null   float64 
 8   cabin      295 non-null    object  
 9   embarked   1307 non-null   category
 10  boat       486 non-null    object  
 11  body       121 non-null    float64 
 12  home.dest  745 non-null    object  
 13  target     1309 non-null   category
dtypes: category(3), float64(6), object(5)
memory usage: 116.8+ KB
```

> Do Download and Explore other datasets from OpenML Repository at your own time...

(iv) Random Sample Generators

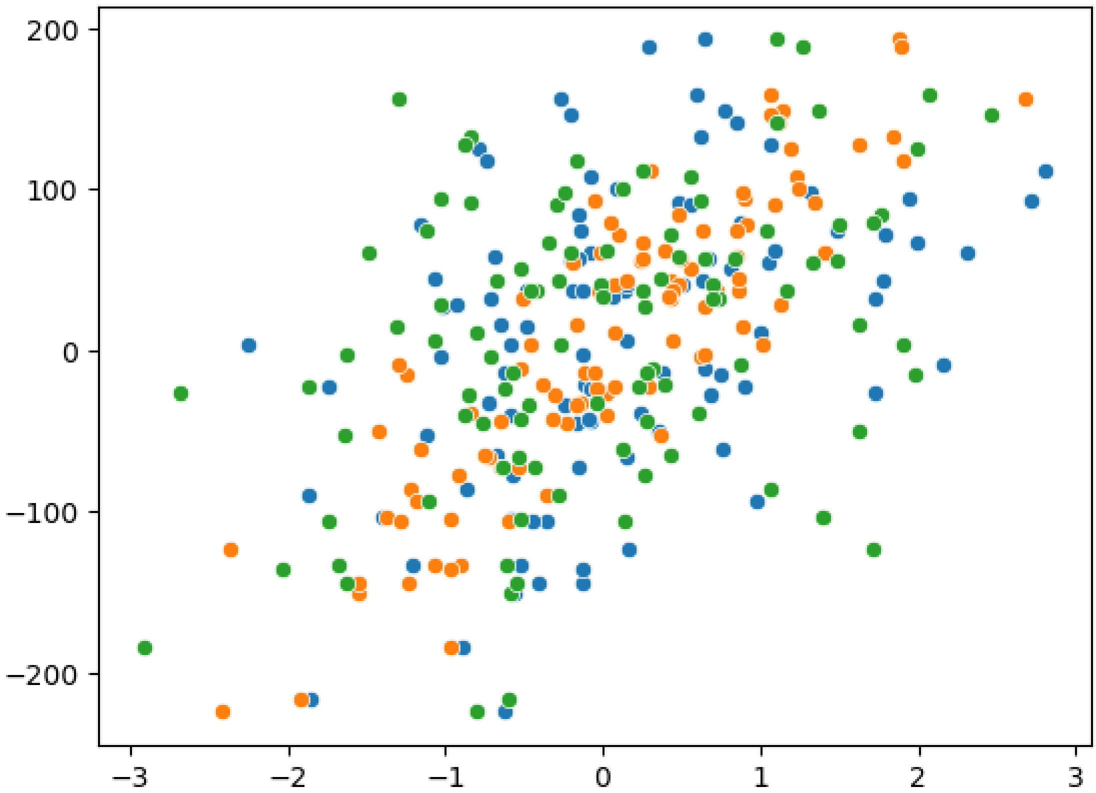
- In addition, scikit-learn includes various random sample generators that can be used to build artificial datasets of controlled size and complexity.
- These generators produce a matrix of features and corresponding discrete targets.
- Please visit [https://scikit-learn.org/stable/datasets/sample_generators.html (https://scikit-learn.org/stable/datasets/sample_generators.html)](https://scikit-learn.org/stable/datasets/sample_generators.html) for details.

Random Dataset Generation for Regression Task:

In [28]:

```
from sklearn import datasets
import seaborn as sns
X, y = datasets.make_regression(n_samples=100, # The number of samples.
                                n_features=3, #The number of features.
                                noise=0.0, #The standard deviation of the gaussian noise
                                n_targets=1, #The number of regression targets, i.e., th
                                random_state=3)#for reproducible output across multiple

sns.scatterplot(x=X[:, 0],y=y); #Plot first input feature with output y
sns.scatterplot(x=X[:, 1],y=y); #Plot second input feature with output y
sns.scatterplot(x=X[:, 2],y=y); #Plot third input feature with output y
```



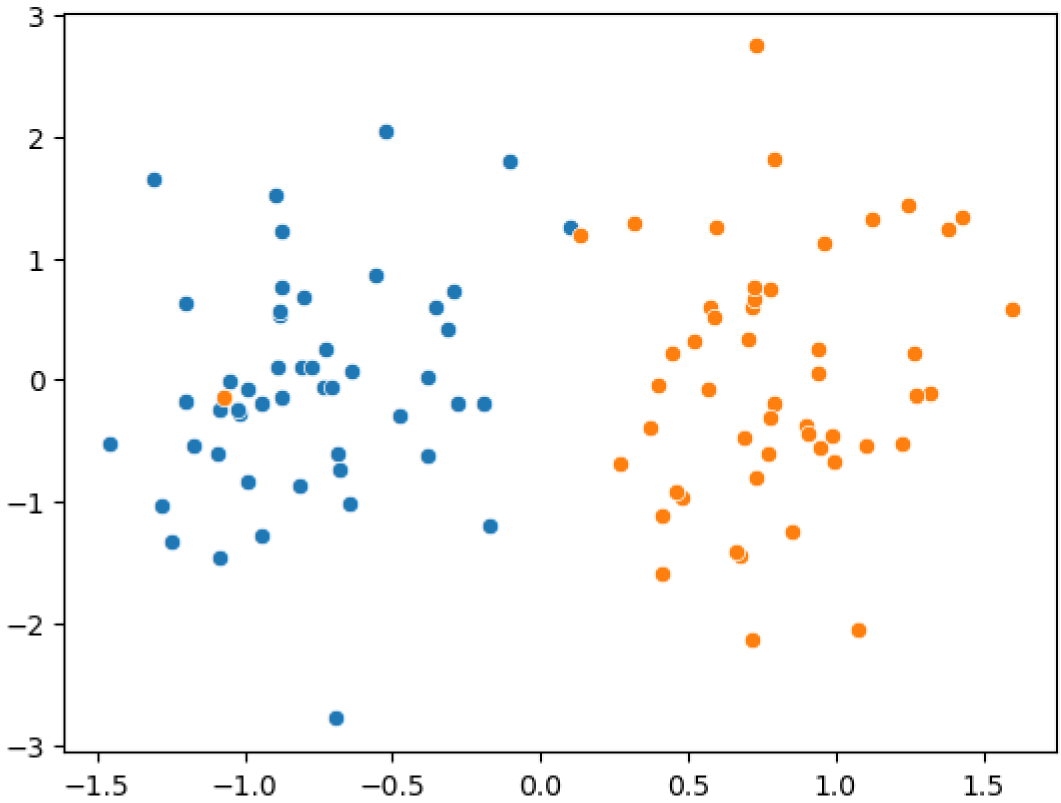
Random Dataset Generation for Classification Task:

- `make_classification()` is used to generate datasets for linear classification tasks.
- `make_moons()` and `make_circles()` are used to generate datasets for nonlinear classification tasks.

In [29]:

```
X, y = datasets.make_classification(n_samples=100,
                                    n_features=5,
                                    n_classes=2,
                                    class_sep=2,
                                    random_state=54)

sns.scatterplot(x=X[:, 0][y==1], y=X[:, 1][y==1]);#When y is 1, the class is rep
sns.scatterplot(x=X[:, 0][y==0], y=X[:, 1][y==0]);
```

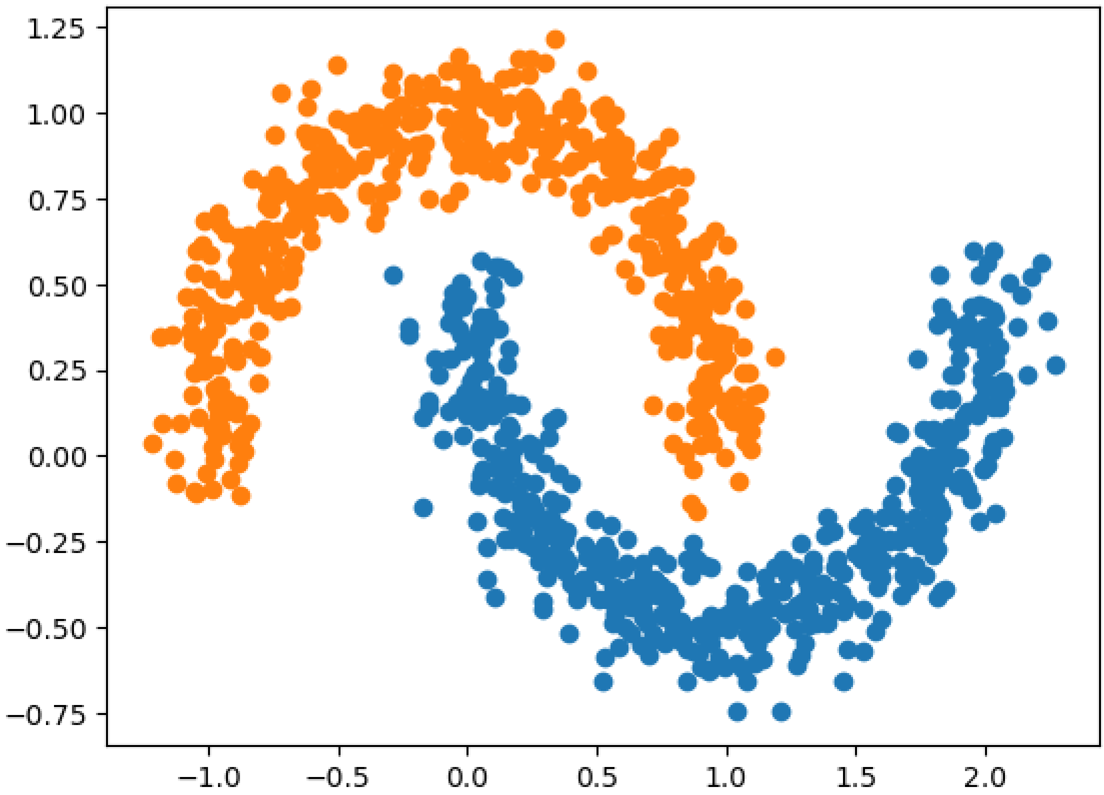


In [30]:

```python
from sklearn.datasets import make_moons
from matplotlib import pyplot as plt

X, y = make_moons(n_samples=1000, noise=0.1, random_state=42)

# When the label y is 0, the class is represented with a blue square.
# When the label y is 1, the class is represented with a green triangle.
plt.scatter(X[:, 0][y==1], X[:, 1][y==1]);
plt.scatter(X[:, 0][y==0], X[:, 1][y==0]);
```

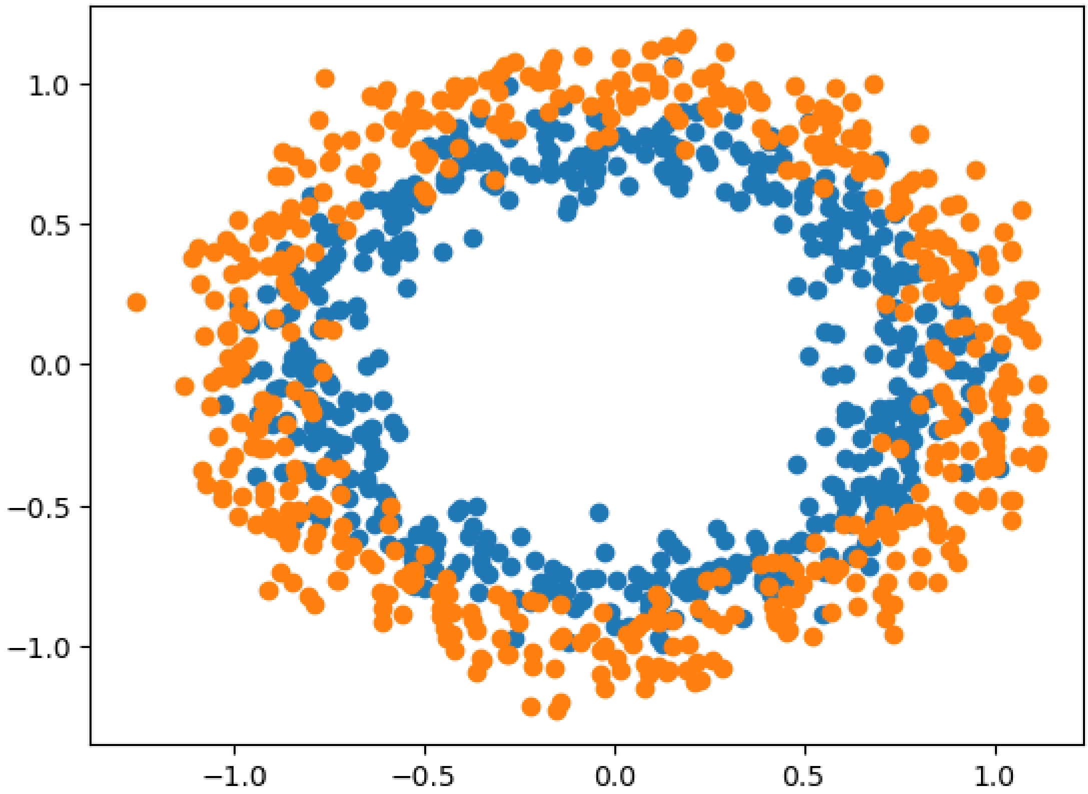


In [31]:

```python
from sklearn.datasets import make_circles
from matplotlib import pyplot as plt

X, y = make_circles(n_samples=1000, noise=0.1, random_state=42)

# When the label y is 0, the class is represented with a blue square.
# When the label y is 1, the class is represented with a green triangle.
plt.scatter(X[:, 0][y==1], X[:, 1][y==1]);
plt.scatter(X[:, 0][y==0], X[:, 1][y==0]);
```



Random Dataset Generation for Clustering Task:

- `make_blobs()` is used to generate datasets for clustering tasks

In [32]:

```
# This code generates a dataset with 500 samples and 2 features (x and y coordi
# with 3 clusters centred at random locations, and with no noise.
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
blobs, blob_labels = make_blobs(n_samples=500,
                                n_features=2,
                                centers=3,
                                random_state=42)

plt.scatter(blobs[:, 0], blobs[:, 1], c=blob_labels);
```