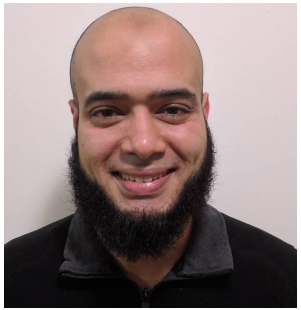
Machine Learning

Scikit-Learn Library

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Scikit-learn (Sklearn) Library

- A powerful **open source** python library for machine learning
  - BSD license
- Initially developed as a **Google summer of code** project in 2007

Installation

- Using pip
  - pip install -U scikit-learn
- Using Conda
  - conda install scikit-learn
- What is Conda?
  - Conda is an open-source, cross-platform, language-agnostic package manager and environment management system
  - We won’t start with conda, but it is great if you install and use
  - Relevant keywords: miniconda, anaconda

Features

- Supervised Learning algorithms
- Unsupervised Learning algorithms
- Clustering
- Model selection and evaluation:
- Dimensionality Reduction
- Ensemble methods
- Feature extraction
- Several datasets for demonstration

Toy datasets

scikit-learn comes with a few small standard datasets that do not require to download any file from some external website.

They can be loaded using the following functions:

`load_iris`(*[, return_X_y, as_frame])	Load and return the iris dataset (classification).
`load_diabetes`(*[, return_X_y, as_frame, scaled])	Load and return the diabetes dataset (regression).
`load_digits`(*[, n_class, return_X_y, as_frame])	Load and return the digits dataset (classification).
`load_linnerud`(*[, return_X_y, as_frame])	Load and return the physical exercise Linnerud dataset.
`load_wine`(*[, return_X_y, as_frame])	Load and return the wine dataset (classification).
`load_breast_cancer`(*[, return_X_y, as_frame])	Load and return the breast cancer wisconsin dataset (classification).

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.

Uniform API

- You will notice algorithms share the same core interface
- This allows us to change from algorithm to another easily
  - An **estimator** interface for building and fitting models
  - A **predictor** interface for making predictions
  - A **transformer** interface for converting data

Learning SKlearn

- During the course, we will meet ML topics
- After we learn the inner details, we will learn and use sklearn as black box
- Today, we will learn a simple thing!

Features Processing

- For some reason to learn later, we usually preprocess the data to make it of a small scale
- There are 2 popular ways for that:
  - MinMaxScaler
  - Standardization

Features Processing: MinMaxScaler

- Assume we have feature data: [10, 90, 22, 21, 20]
- First, compute the min and maximum: 10 and 90
- Convert each number using: (x - min) / (max-min)
- Let’s apply
  - For x = 10 = min ⇒ 0/80 = 0
  - For x = 90 = max ⇒ 80/80 = 1
- In other words, all the numbers will be between [0, 1]
  - 20 will be 0.125 21 will be 0.1375 22 will be (22 - 10) / 80 = 0.15
- Think about the intuition of this formula

Features Processing: MinMaxScaler

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler

data = np.array([ [1, 10],
                  [2, 20],
                  [3, 21],
                  [4, 22],
                  [5, 90]])
print(data)
'''
[[ 1 10]
 [ 2 20]
 [ 3 21]
 [ 4 22]
 [ 5 90]]
'''

processor = MinMaxScaler()
data_scaled = processor.fit_transform(data)
```

```
print(data_scaled)
'''
[[0.      0.     ]
 [0.25    0.125 ]
 [0.5     0.1375]
 [0.75    0.15  ]
 [1.      1.     ]]
'''

test_data = np.array([[1, 5],
                      [2.5, 20],
                      [6, 100]])

print(processor.transform(test_data))
'''
[[ 0.     -0.0625]
 [ 0.375   0.125 ]
 [ 1.25    1.125 ]]
'''
```

Features Processing: StandardScaler

- Another scaler is the standard scaler that **standardize** features by removing the mean and scaling to unit variance
  - Mean = 0 and Sigma = 1
- How to compute: compute data mean and data sigma
- For each value x transform to: (x - mean) / sigma

```
from sklearn.preprocessing import StandardScaler

processor = StandardScaler()
data_scaled = processor.fit_transform(data)

print(data_scaled)
print(processor.transform(test_data))
```

Your turn

- 1) Get some data. Compute their minmax scaler and compare with sklearn
- 2) The same for StandardScaler
- 3) Get use to read the documentation in details
  - There are always a lot of features and notes
  - Do that for the 2 SKlearn processors

Note about licences

- It is very important to educate yourself about licences
- In our field, we may download and use model, code or datasets
- We must know what kind of licenses are supported
- If your company launched a product without proper license, this can get them in legal problems
- Contact the company’s legal departments before using something in the production
    - Prototyping is less sensitive

Relevant Materials (for future)

- Tutorial (read): link, link
- Cheat sheet
- Arabic youtube playlist: Eng Hesham Asem
- Youtube should have many playlists/videos: link, link

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."