CS 103 -06

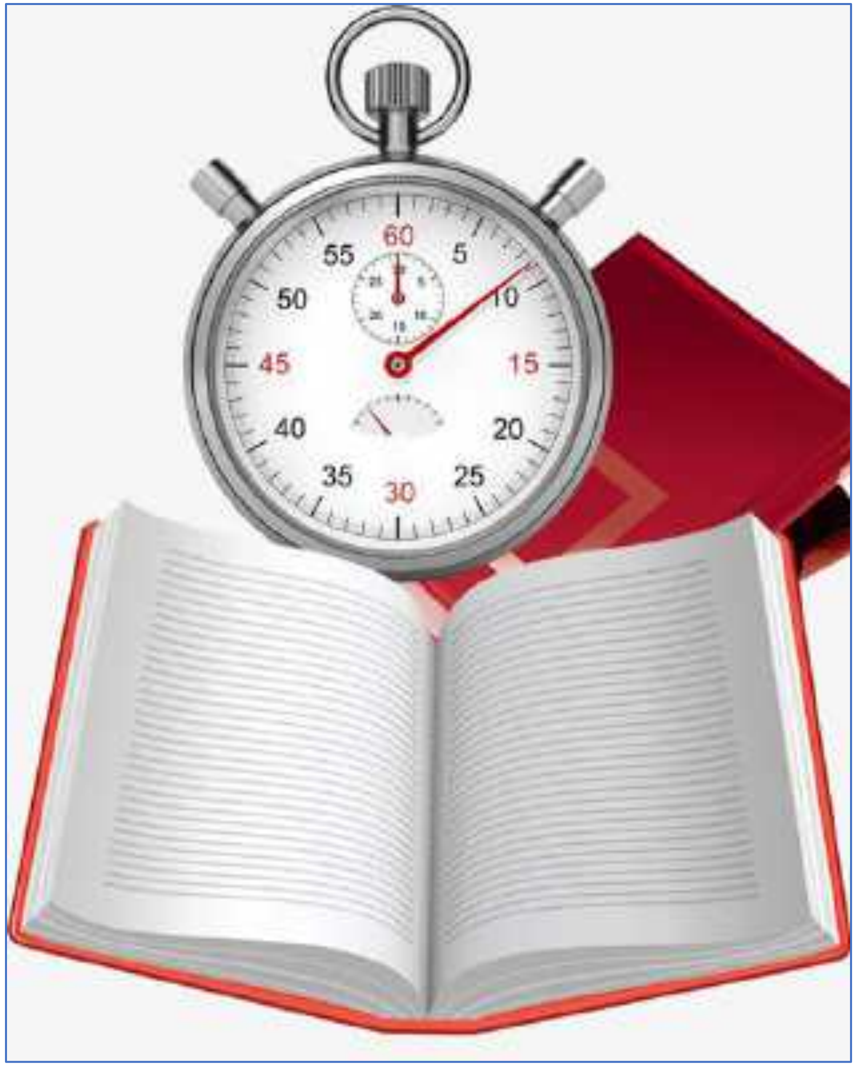
Mid-Term Review and AI Platform Introduction

Jimmy Liu 刘江

TA： 章晓庆， 张洋， 肖尊杰， 杨冰

2020-10-23

Lectures Review



Learn and Study

Active learning: It is about how much you think and learn

Collective study: Let us study together

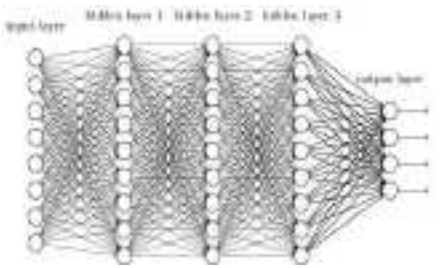
CS 103 Module Coverage

What CS 103 will cover?

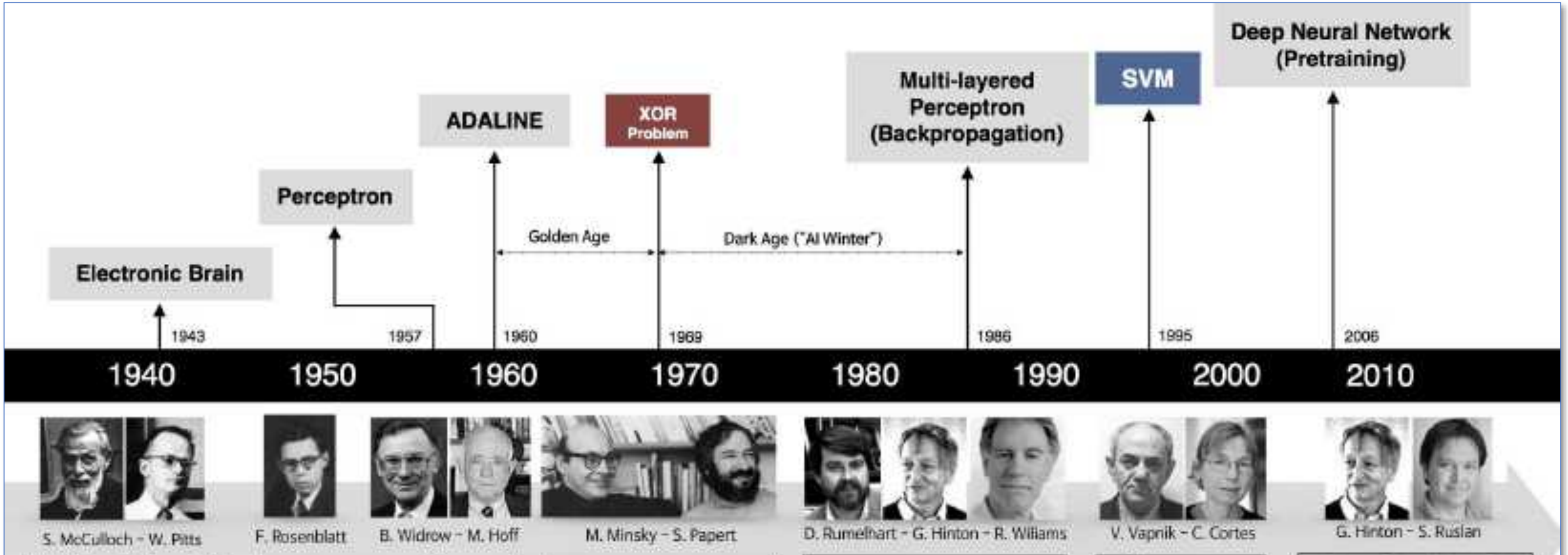
AI Concepts

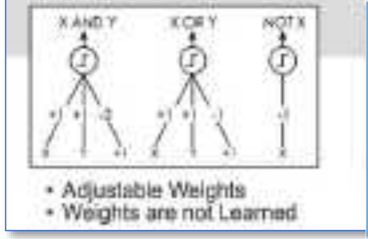
AI Algorithms

AI Application



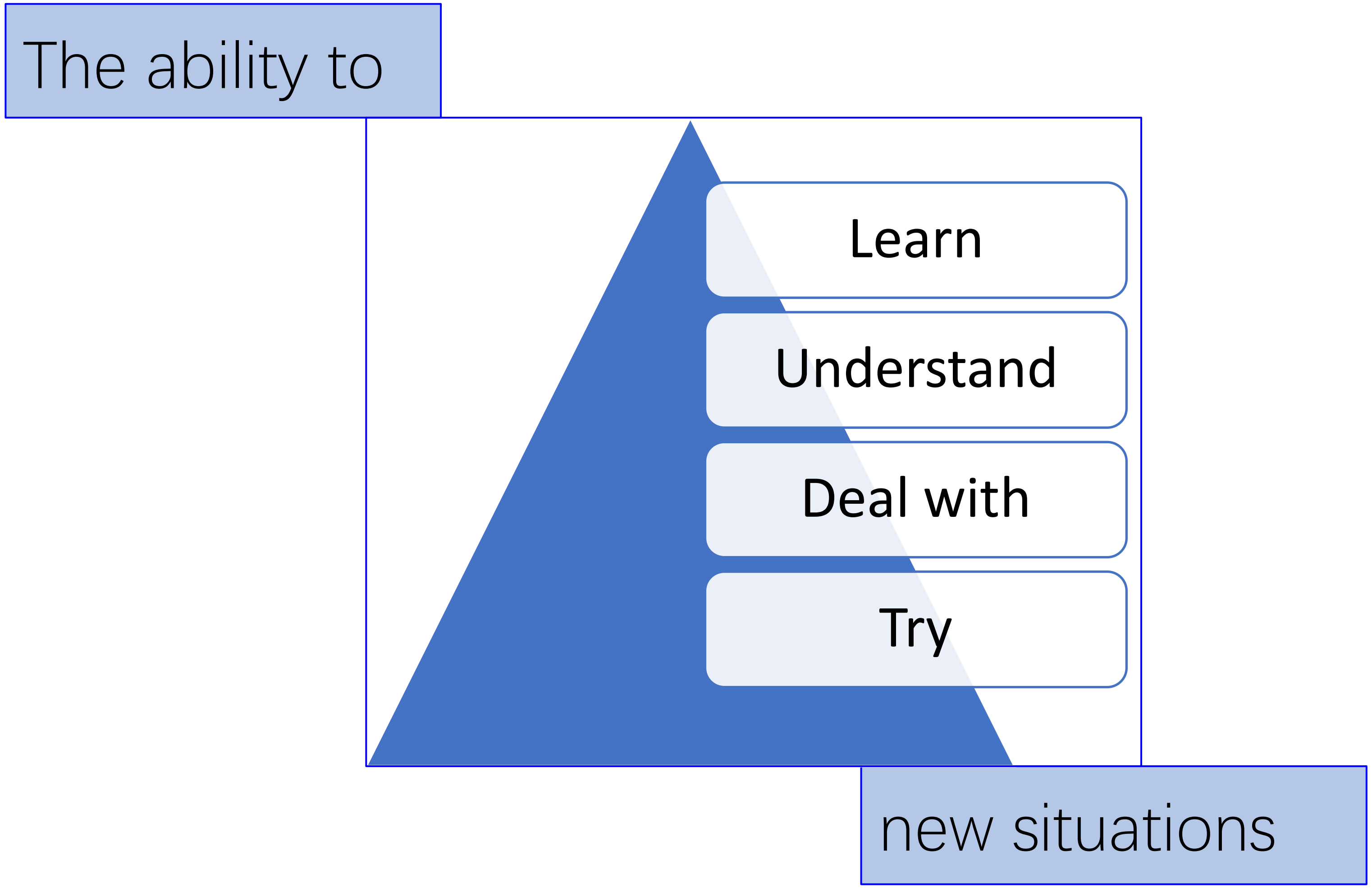
Computer Algorithm and AI algorithm Development Stages and Future Direction





ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's
Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

“Intelligence” from Dictionaries



The ability to

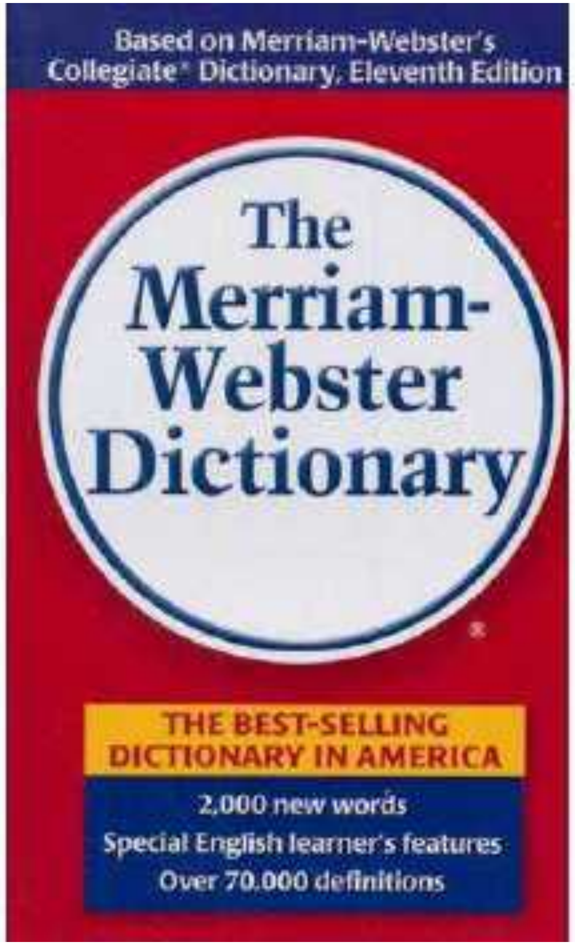
Learn

Understand

Deal with

Try

new situations



Based on Merriam-Webster's Collegiate® Dictionary, Eleventh Edition

The Merriam-Webster Dictionary

THE BEST-SELLING DICTIONARY IN AMERICA

2,000 new words

Special English learner's features

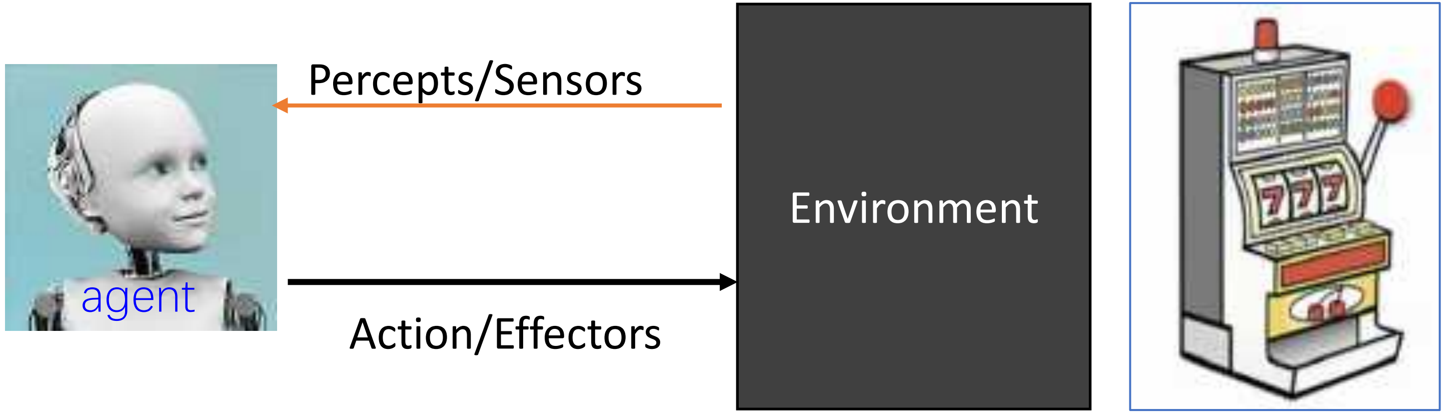
Over 70,000 definitions

AI from Computer Science - Agent

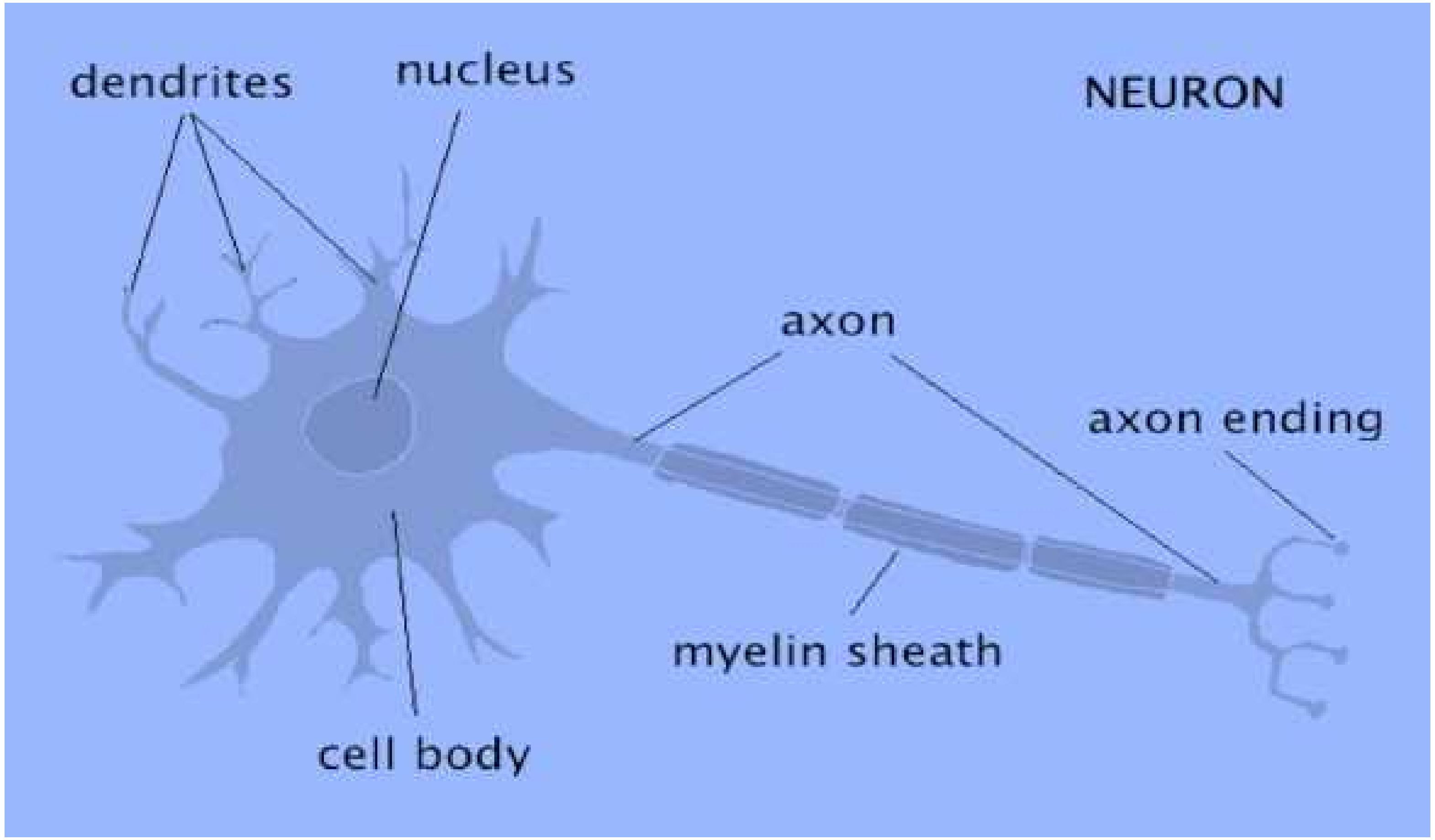
- An agent is anything that can perceive its environment through sensors and acts upon that environment through effectors. Abstractly, an agent is a function from percept histories to actions:

  $$[f: P^* \rightarrow A]$$

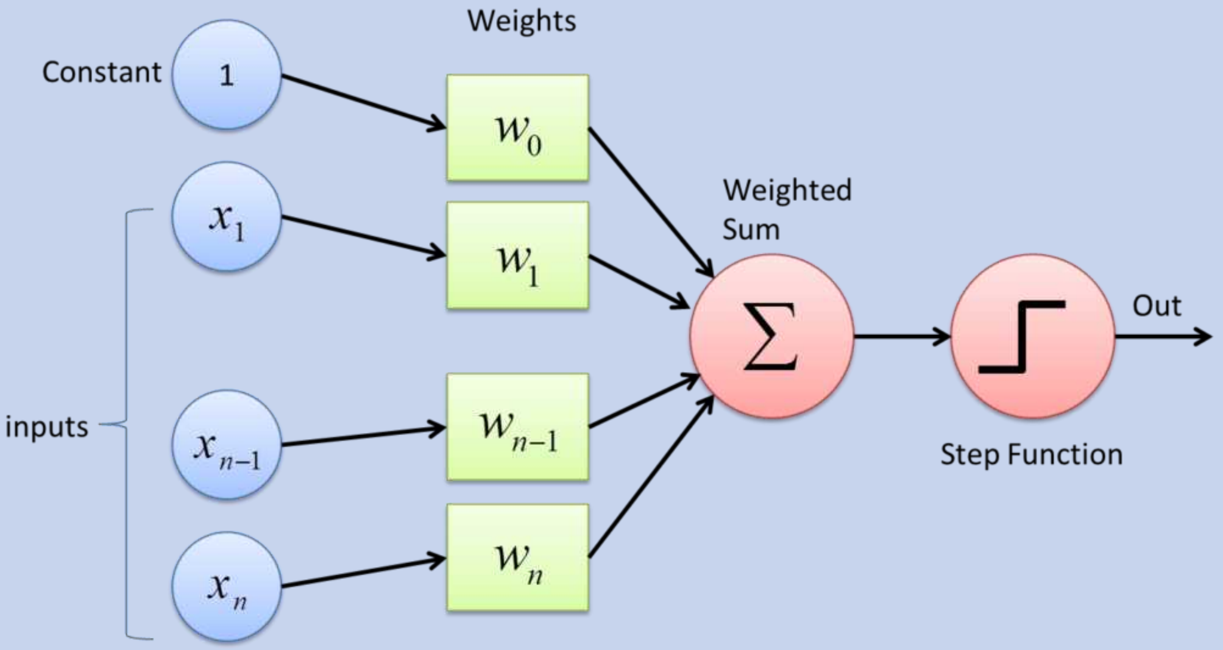
- A human agent has sensory organs such as eyes, ears, nose, tongue and skin parallel to the sensors, and other organs such as hands, legs, mouth, for effectors.
- A robotic agent replaces cameras and infrared range finders for the sensors, and various motors and actuators for effectors.
- A software agent has encoded bit strings as its programs and actions.



Single Neuron

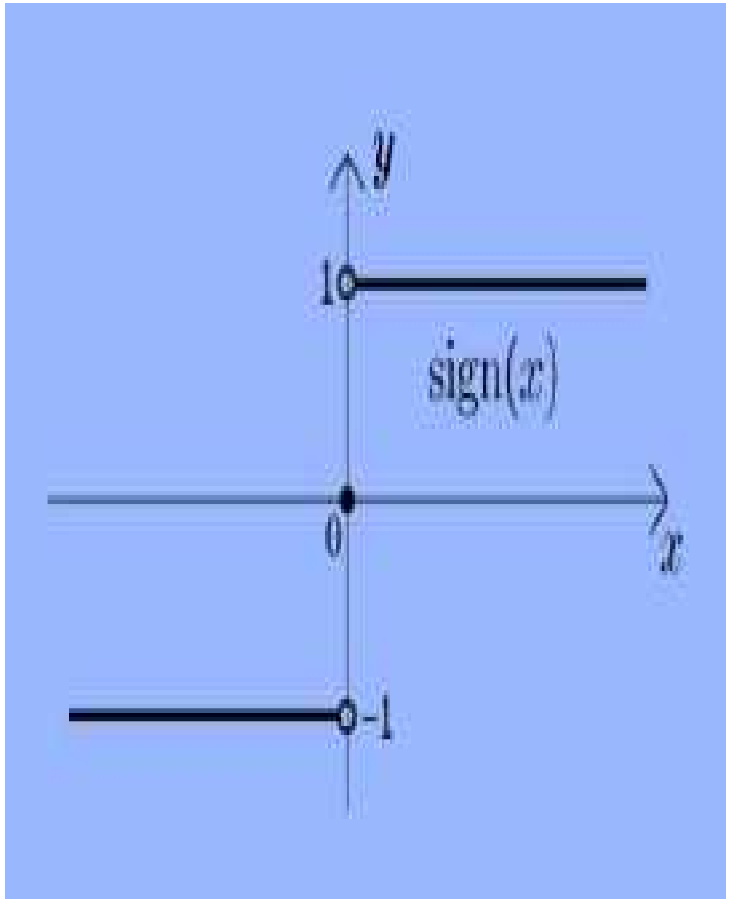


From MCP Neuron to Perceptron



$$f(x) = sign(w \cdot x + b)$$

$$sign(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$



Perceptron Learning Rule 1: PLR

1. Randomly choose the weights in the range 0 and 1.

2. Training examples are presented to perceptron one by one from the beginning, and its output is observed for each training example.

3. If the output is correct then the next training example is presented to perceptron.

4. If the output is incorrect then the weights are modified as per the following Perceptron Learning Rule (PLR).

New Wi = Wi + (η * Xi * E).

Change in Weight i = Learning Rate × Current Value of Input i
× E (Expected Output, Current Output).

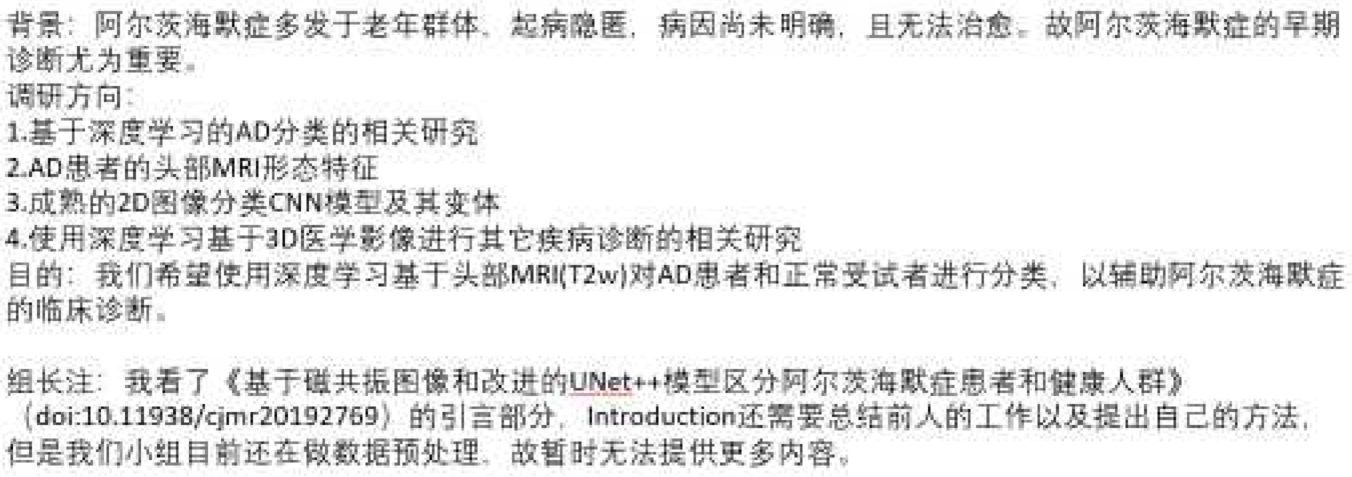
5. A simple form of E = (Expected Output - Current Output) or

SIGN (Expected Output - Current Output).

6. In PLR, output is 1/0 (or -1), and the transfer is Threshold Step Function

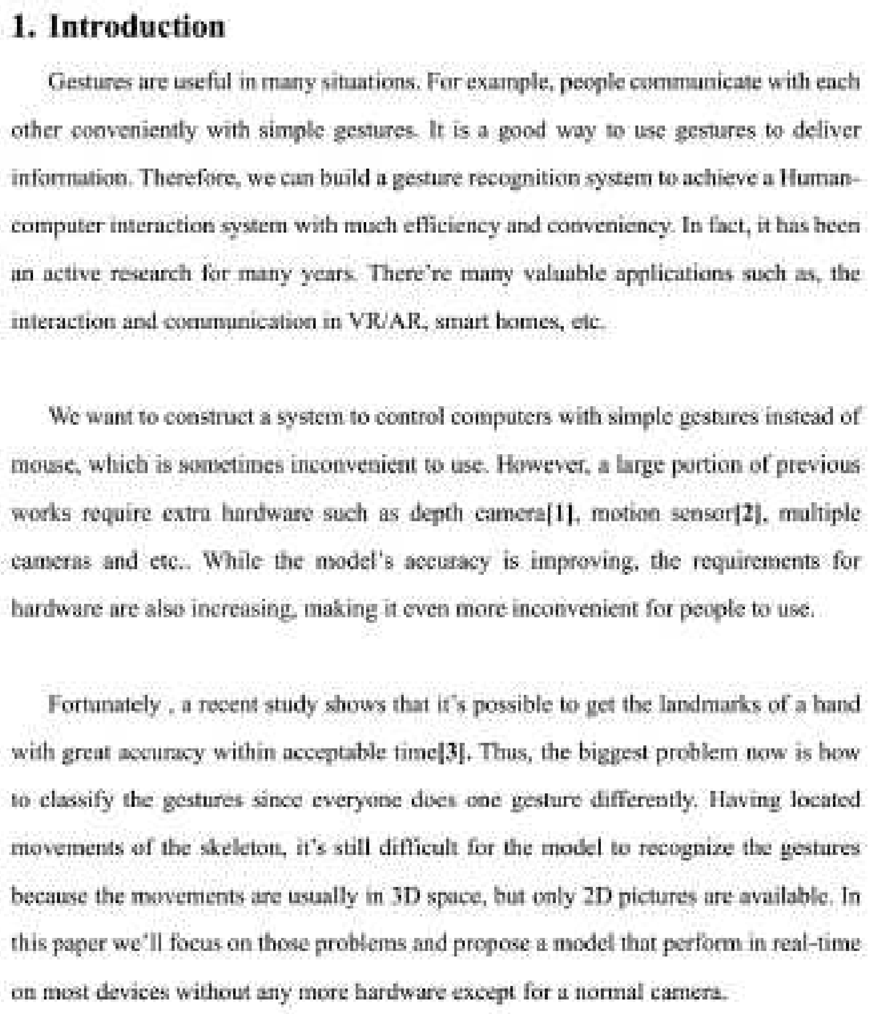
Any Question?

Group Project Introduction 1



背景：阿尔茨海默症多发于老年群体，起病隐匿，病因尚未明确，且无法治愈。故阿尔茨海默症的早期诊断尤为重要。
调研方向：
1.基于深度学习的AD分类的相关研究
2.AD患者的头部MRI形态特征
3.成熟的2D图像分类CNN模型及其变体
4.使用深度学习基于3D医学影像进行其它疾病诊断的相关研究
目的：我们希望使用深度学习基于头部MRI(T2w)对AD患者和正常受试者进行分类，以辅助阿尔茨海默症的临床诊断。

组长注：我看了《基于磁共振图像和改进的UNet++模型区分阿尔茨海默症患者和健康人群》（doi:10.11938/cjmr20192769）的引言部分，Introduction还需要总结前人的工作以及提出自己的方法，但是我们小组目前还在做数据预处理，故暂时无法提供更多内容。



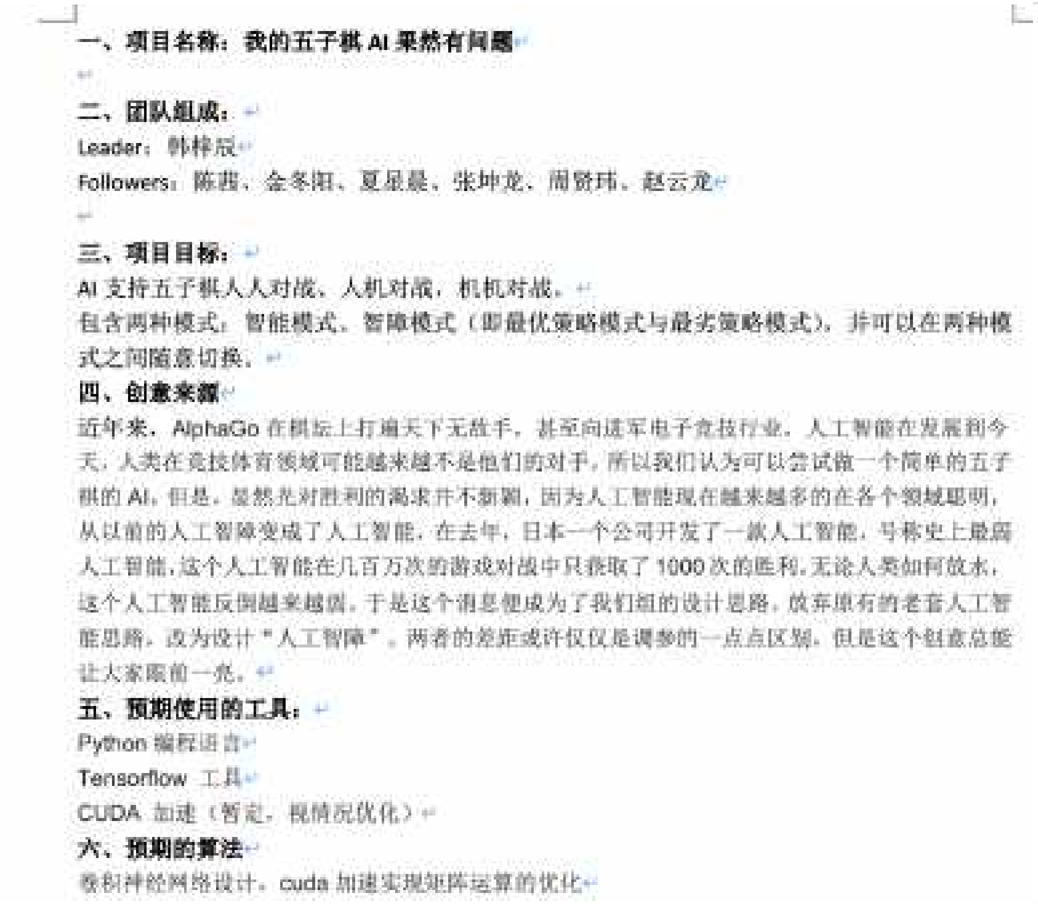
1. Introduction

Gestures are useful in many situations. For example, people communicate with each other conveniently with simple gestures. It is a good way to use gestures to deliver information. Therefore, we can build a gesture recognition system to achieve a Human-computer interaction system with much efficiency and conveniency. In fact, it has been an active research for many years. There're many valuable applications such as, the interaction and communication in VR/AR, smart homes, etc.

We want to construct a system to control computers with simple gestures instead of mouse, which is sometimes inconvenient to use. However, a large portion of previous works require extra hardware such as depth camera[1], motion sensor[2], multiple cameras and etc.. While the model's accuracy is improving, the requirements for hardware are also increasing, making it even more inconvenient for people to use.

Fortunately , a recent study shows that it's possible to get the landmarks of a hand with great accuracy within acceptable time[3]. Thus, the biggest problem now is how to classify the gestures since everyone does one gesture differently. Having located movements of the skeleton, it's still difficult for the model to recognize the gestures because the movements are usually in 3D space, but only 2D pictures are available. In this paper we'll focus on those problems and propose a model that perform in real-time on most devices without any more hardware except for a normal camera.

Group Project Introduction 2



一、项目名称：我的五子棋 AI 果然有问题

二、团队组成：

Leader：韩梓辰

Followers：陈茜、佘冬阳、夏晨晨、张坤龙、甯贤玮、赵云龙

三、项目目标：

AI 支持五子棋人人对战、人机对战，机机对战。

包含两种模式：智能模式、智障模式（即最优策略模式与最劣策略模式），并可以在两种模式之间随意切换。

四、创意来源

近年来，AlphaGo 在棋坛上打遍天下无敌手，甚至向进军电子竞技行业，人工智能在发展到今天，人类在竞技体育领域可能越来越不是他们的对手，所以我们认为可以尝试做一个简单的五子棋的 AI，但是，显然光对胜利的渴求并不新颖，因为人工智能现在越来越多的在各个领域聪明，从以前的人工智障变成了人工智能，在去年，日本一个公司开发了一款人工智能，号称史上最弱人工智能，这个人工智能在几百万次的游戏对战中只获取了 1000 次的胜利，无论人类如何放水，这个人工智能反倒越来越弱，于是这个消息便成为了我们组的设计思路，放弃原有的老套人工智能思路，改为设计"人工智障"。两者的差距或许仅仅是调参的一点点区别，但是这个创意总能让大家眼前一亮。

五、预期使用的工具：

Python 编程语言

Tensorflow 工具

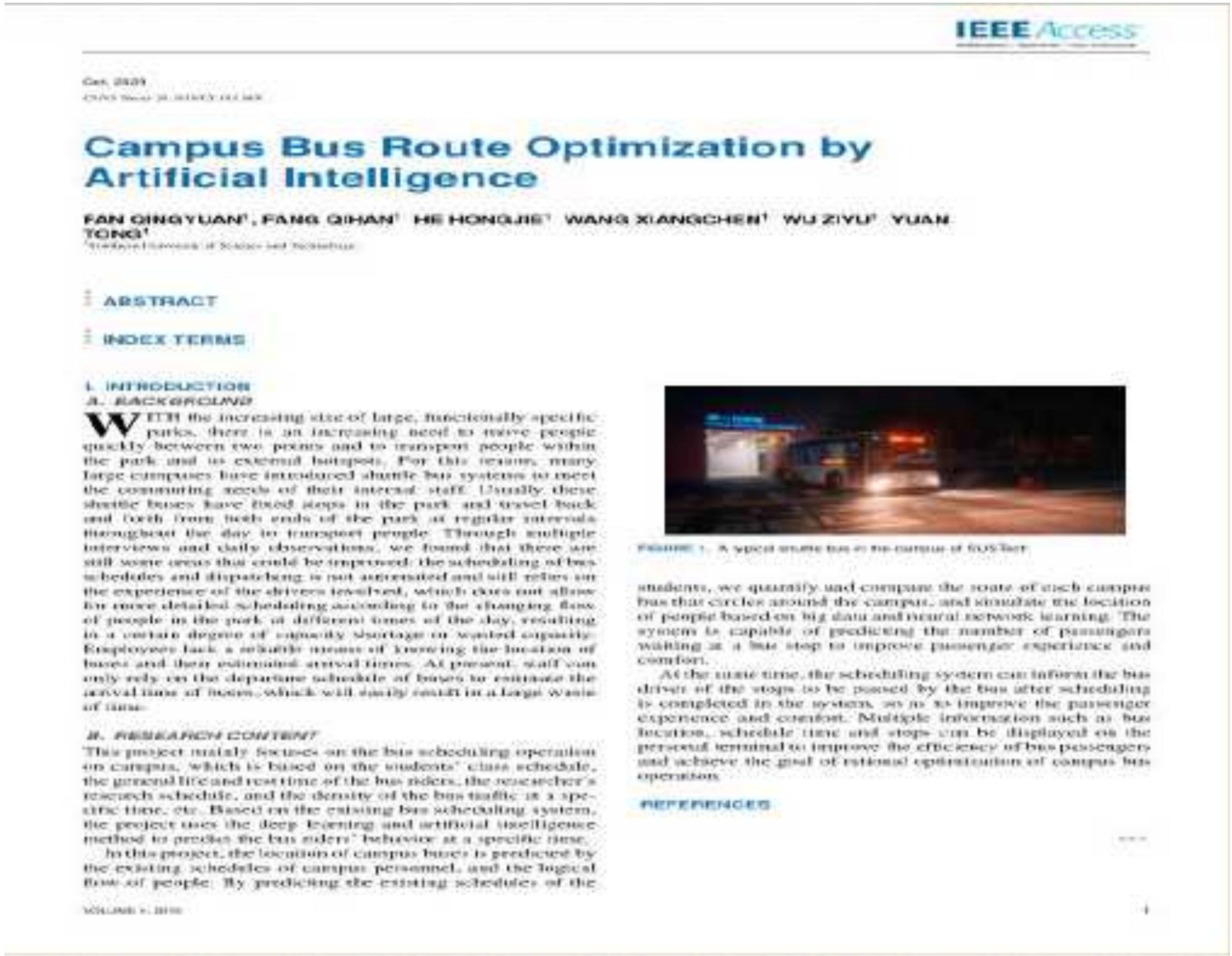
CUDA 加速（暂定，视情况优化）

六、预期的算法

卷积神经网络设计，cuda 加速实现矩阵运算的优化

图像着色

Group Project Introduction 3



IEEE Access

Oct, 2023

[illegible]

Campus Bus Route Optimization by Artificial Intelligence

FAN QINGYUAN[1], FANG QIHAN[1] HE HONGJIE[1] WANG XIANGCHEN[1] WU ZIYU[1] YUAN TONG[1]

[1]Southern University of Science and Technology

ABSTRACT

INDEX TERMS

I. INTRODUCTION

A. BACKGROUND

WITH the increasing size of large, functionally specific parks, there is an increasing need to move people quickly between two points and to transport people within the park and its external hotspots. For this reason, many large campuses have introduced shuttle bus systems to meet the commuting needs of their internal staff. Usually these shuttle buses have fixed stops in the park and travel back and forth from both ends of the park at regular intervals throughout the day to transport people. Through multiple interviews and daily observations, we found that there are still some areas that could be improved: the scheduling of bus schedules and dispatching is not automated and still relies on the experience of the drivers involved, which does not allow for more detailed scheduling according to the changing flow of people in the park at different times of the day, resulting in a certain degree of capacity shortage or wasted capacity. Employees lack a reliable means of knowing the location of buses and their estimated arrival times. At present, staff can only rely on the departure schedule of buses to estimate the arrival time of buses, which will easily result in a large waste of time.

B. RESEARCH CONTENT

This project mainly focuses on the bus scheduling operation on campus, which is based on the students' class schedule, the general life and rest time of the bus riders, the researcher's research schedule, and the density of the bus traffic at a specific time, etc. Based on the existing bus scheduling system, the project uses the deep learning and artificial intelligence method to predict the bus riders' behavior at a specific time.

In this project, the location of campus buses is predicted by the existing schedules of campus personnel, and the logical flow of people. By predicting the existing schedules of the students, we quantify and compare the route of each campus bus that circles around the campus, and simulate the location of people based on big data and neural network learning. The system is capable of predicting the number of passengers waiting at a bus stop to improve passenger experience and comfort.

FIGURE 1. A typical shuttle bus in the campus of SUSTech

At the same time, the scheduling system can inform the bus driver of the stops to be passed by the bus after scheduling is completed in the system, so as to improve the passenger experience and comfort. Multiple information such as bus location, schedule time and stops can be displayed on the personal terminal to improve the efficiency of bus passengers and achieve the goal of rational optimization of campus bus operation.

REFERENCES

Group Project Introduction 4

我们小组研究方向是AI+虚拟主播，其主要内容是识别收到的文本，将其转化为信息，再通过语音的形式输出。我们会致力于日常的对话，使之更加智能化，人性化，以致可以进行基本没有逻辑错误的日常对话和使用。

1

Start to Write the Introduction of Your Group Project

The research topic of our group is the user preferences of big data computing, and the research on big data in an era of large amount of information

1

Start to Write the Introduction of Your Group Project

The whole project is based on the MVC model, which consist four parts Front end, control, business logic and data base.

https://github.com/QAQGaeBolg/SUSTech_CS103_2020F/blob/main/Demand_ver_1.md

Outline

3 4

1 Python

2 Scikit-learn

3 Pytorch

4 Tensorflow

Background

- **Python** is an interpreted, high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for large-scale projects.
- Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.
- The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release." No more security patches or other improvements will be released for it. With Python 2’s end-of-life, only Python 3.6.x and later are supported.

Background

- **Basic Python construct:**

Focused on Python 3.0

- **Help you build a foundation**

Get ready to follow online tutorials

https://www.runoob.com/python/python-tutorial.html

Background

Python is an interpreted, high-level, object-oriented programming language.

- Flexible programming language
- Designed to be human readable

Related applications:

- Machine learning models
- Artificial intelligence projects
- Web applications

Print

- print displays output to your console

```
print('Hello world')
```

```
Hello world
```

Print

- Enclose strings in single or double quotes

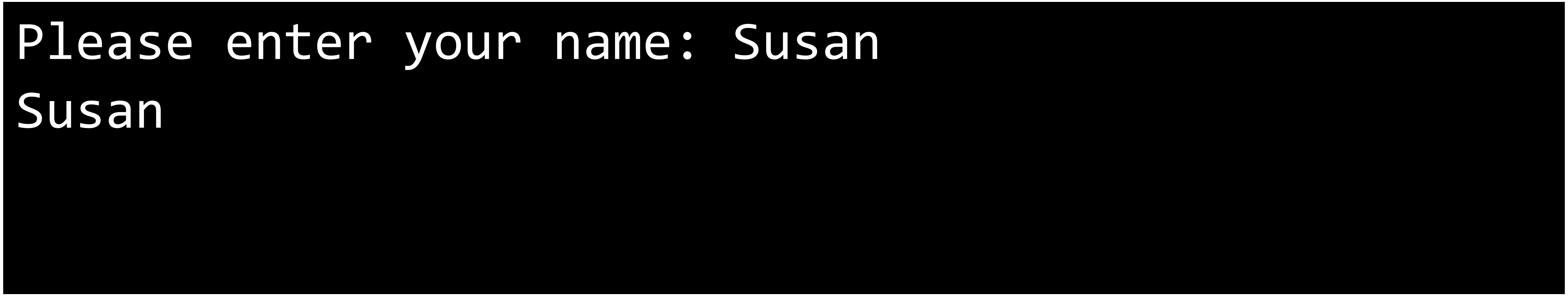
```
print('Hello world single quotes')
print("Hello world double quotes")
```

```
Hello world single quotes
Hello world double quotes
```

Print

- Getting information from the user

```
name = input('Please enter your name: ')
print(name)
```

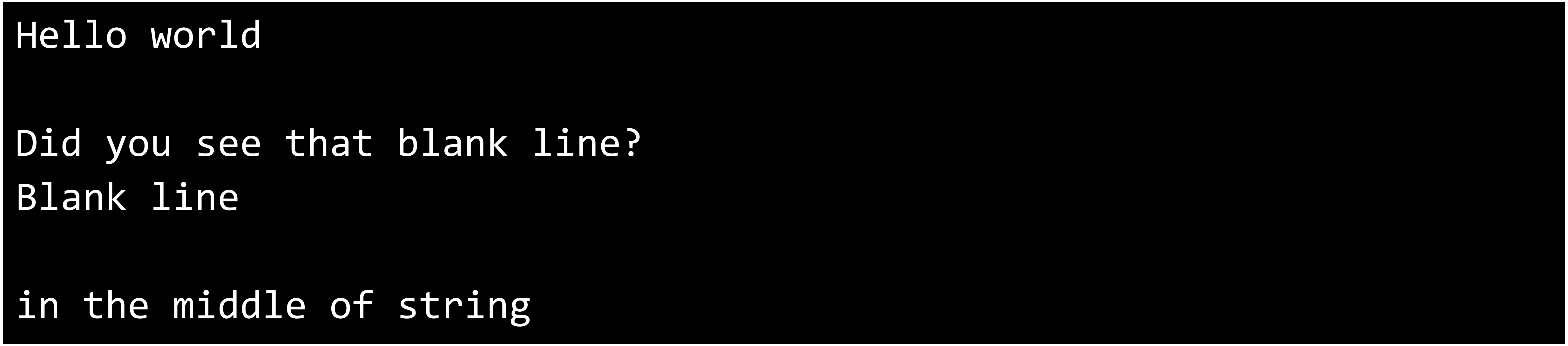


```
Please enter your name: Susan
Susan
```

Print

- Printing blank lines can improve readability

```
print('Hello world')
print()
print('Did you see that blank line?')
print('Blank line \nin the middle of string')
```



```
Hello world

Did you see that blank line?
Blank line

in the middle of string
```

Debug

- Debugging with print

```
print('Adding numbers')
x = 42 + 206
print('Performing division')
y = x / 0
print('Math complete')
```

```
Adding numbers
Performing division
Traceback (most recent call last):
  File "demo.py", line 4, in <module>
    y = x / 0
ZeroDivisionError: float division by zero
```

Comment （注释）

- Comments document your code so you and other programmers can understand the code

```python
# Using double quotes for this string because
# the string itself contains a single quote
print("It's a small world after all")
```

```
It's a small world after all
```

Variables

- Strings can be stored in variables

```python
first_name = 'Susan'
print(first_name)
```

```
Susan
```

Variables

- You can combine strings with +

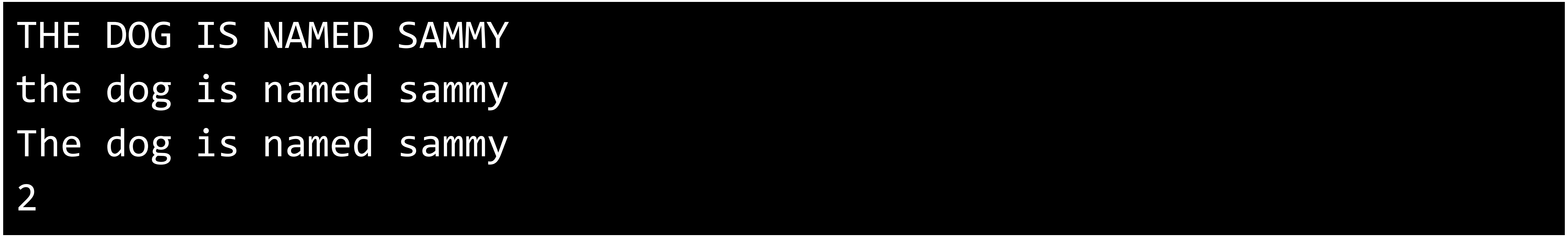
```python
first_name = 'Susan'
last_name = 'Ibach'
print(first_name + last_name)
print('Hello ' + first_name + ' ' + last_name)
```

```
SusanIbach
Hello Susan Ibach
```

Variables

- You can use functions to modify strings

```
sentence = 'The dog is named Sammy'
print(sentence.upper())
print(sentence.lower())
print(sentence.capitalize())
print(sentence.count('a'))
```



```
THE DOG IS NAMED SAMMY
the dog is named sammy
The dog is named sammy
2
```

Variables

- Numbers can be stored in variables

```
pi = 3.14159
print(pi)
```

```
3.14159
```

Variables

- You can do math with numbers

```
first_num = 6
second_num = 2
print(first_num + second_num)
print(first_num ** second_num)
```

```
8
36
```

Symbol	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponent

Variables

- If you combine strings with numbers, Python gets confused
- When displaying a string that contains numbers you must convert the numbers into strings

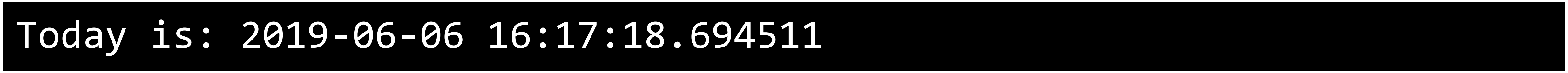
```
days_in_feb = 28
print(days_in_feb + ' days in February’)

days_in_feb = 28
print(str(days_in_feb) + ' days in February')
```

```
File "February.py", line 2, in <module>
    print(days_in_feb + ' days in February')
TypeError: unsupported operand type(s) for +: 'int' and 'str’

28 days in February
```

Date

- We often need current date and time when logging errors and saving data

```
# To get current date and time
# we need to use the datetime library
from datetime import datetime

current_date = datetime.now()
# the now function returns a datetime object
print('Today is: ' + str(current_date))
```

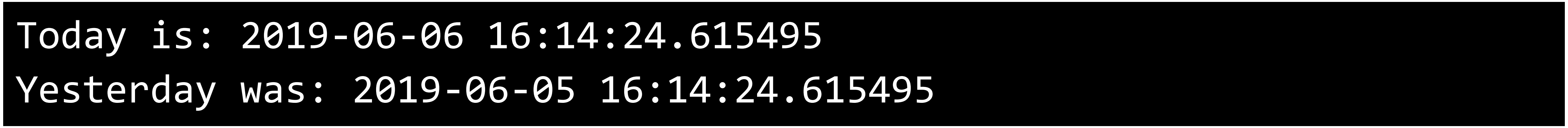


```
Today is: 2019-06-06 16:17:18.694511
```

Date

- There are functions you can use with datetime objects to manipulate dates

```
from datetime import datetime, timedelta
today = datetime.now()
print('Today is: ' + str(today))

# timedelta is used to define a period of time
one_day = timedelta(days=1)
yesterday = today - one_day
print('Yesterday was: ' + str(yesterday))
```

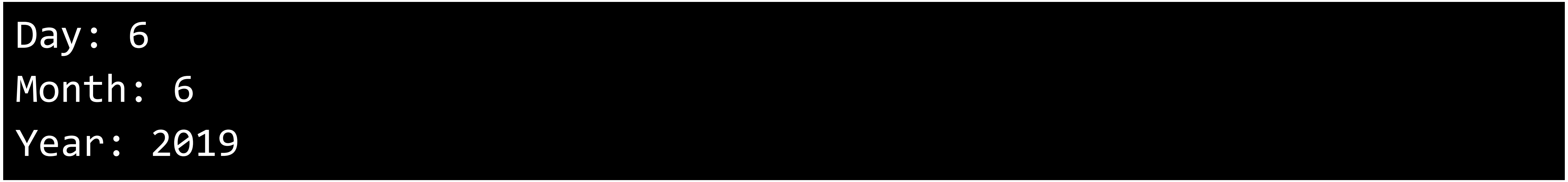


```
Today is: 2019-06-06 16:14:24.615495
Yesterday was: 2019-06-05 16:14:24.615495
```

Date

- Use date functions to control date formatting

```
from datetime import datetime
current_date = datetime.now()

print('Day: ' + str(current_date.day))
print('Month: ' + str(current_date.month))
print('Year: ' + str(current_date.year))
```



```
Day: 6
Month: 6
Year: 2019
```

Error

- Syntax errors

```
# This code won't run at all
x = 42
y = 206
if x == y
    print('Success!!')
```

```
 File "syntax.py", line 3
   if x == y
           ^


SyntaxError: invalid syntax
```

Error

- Runtime errors

```
# This code will fail when run
x = 42
y = 0
print(x / y)
```

```
Traceback (most recent call last):
  File "runtime.py", line 3, in <module>
    print(x / y)
ZeroDivisionError: division by zero
```

Error

- Catching runtime errors

```python
try:
    print(x / y)
except ZeroDivisionError as e:
    # Optionally, log e somewhere
    print('Sorry, something went wrong')
except:
    print('Something really went wrong')
finally:
    print('This always runs on success or failure')
```

```
Sorry, something went wrong
```

Judegement

- Your code needs the ability to take different actions based on different conditions

```
if price >= 1.00:
    tax = .07
    print(tax)
```

Symbol	Operation
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	is equal to
!=	is not equal to

Judgement

- You can add a default action using else

```
if price >= 1.00:
    tax = .07
    print(tax)
else:
    tax = 0
    print(tax)
```

Judgement

- Be careful when comparing strings

```
country = 'CANADA'
if country == 'canada':
    print('Oh look a Canadian')
else:
    print('You are not from Canada')
```

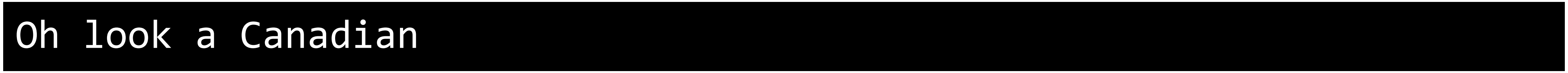
String comparisons are case sensitive

```
You are not from Canada
```

Judgement

- Use string functions to make case insensitive comparisons

```
country = 'CANADA'
if country.lower() == 'canada':
    print('Oh look a Canadian')
else:
    print('You are not from Canada')
```

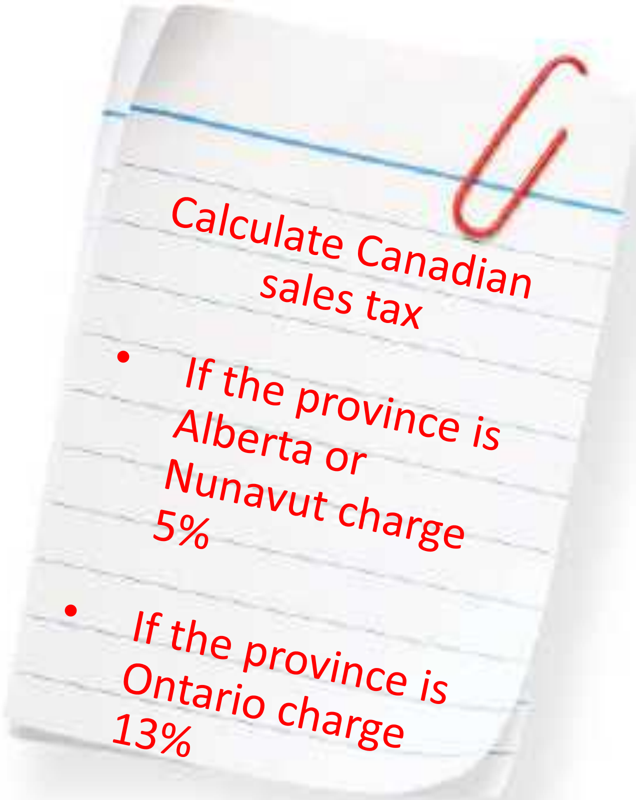


```
Oh look a Canadian
```

Judgement

- You may need to check multiple conditions to determine the correct action

```
if province == 'Alberta':
    tax = 0.05
if province == 'Nunavut':
    tax = 0.05
if province == 'Ontario':
    tax = 0.13
```



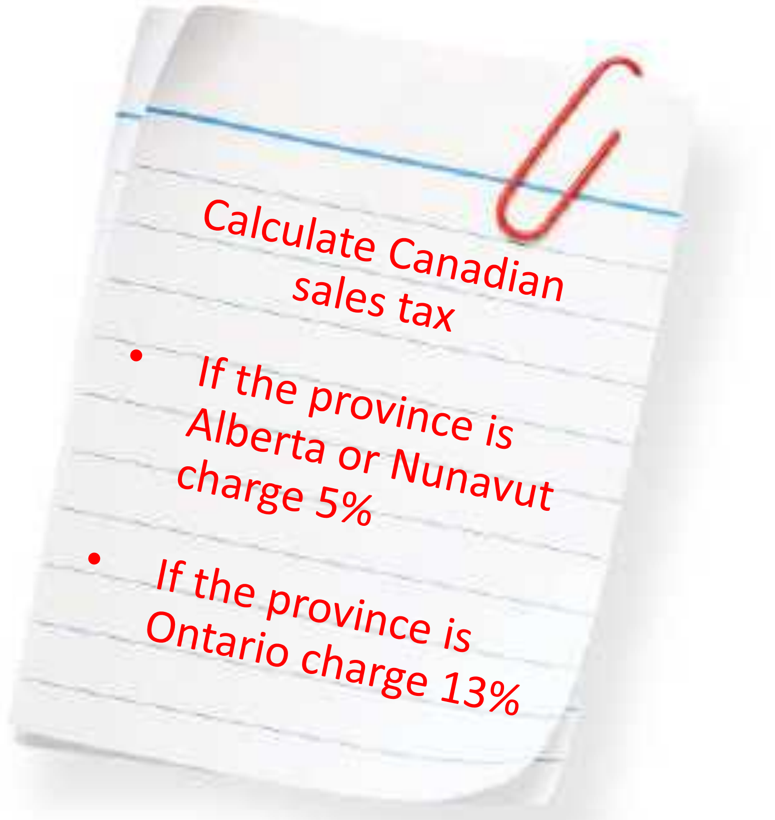
Calculate Canadian sales tax

- If the province is Alberta or Nunavut charge 5%
- If the province is Ontario charge 13%

Judgement

- If only one of the conditions will ever occur you can use a single if statement with elif

```
if province == 'Alberta':
    tax = 0.05
elif province == 'Nunavut':
    tax = 0.05
elif province == 'Ontario':
    tax = 0.13
```



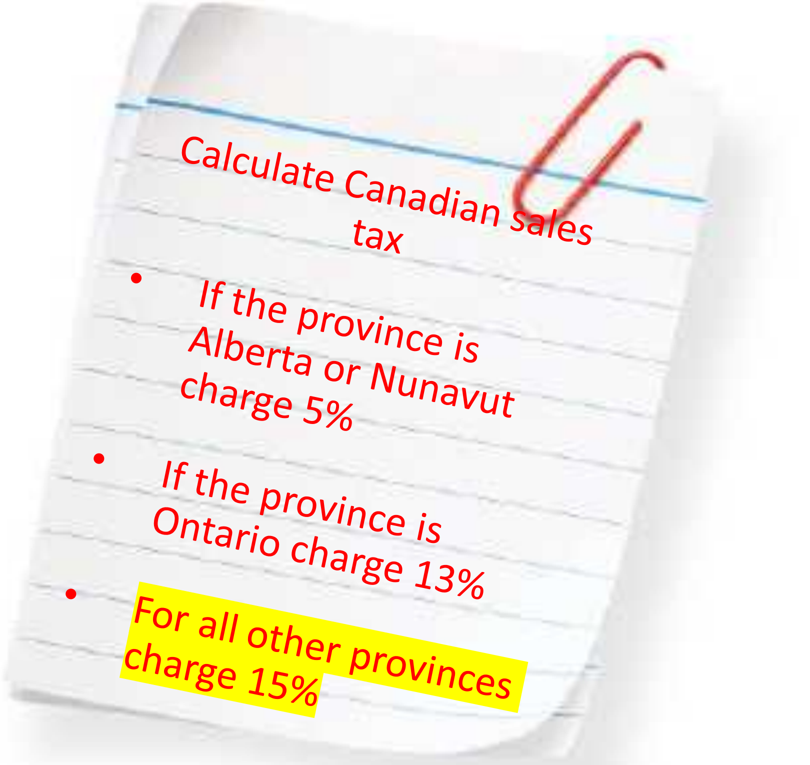
Calculate Canadian sales tax

- If the province is Alberta or Nunavut charge 5%
- If the province is Ontario charge 13%

Judgement

- When you use elif instead of multiple if statements you can add a default action

```
if province == 'Alberta':
    tax = 0.05
elif province == 'Nunavut':
    tax = 0.05
elif province == 'Ontario':
    tax = 0.13
else:
    tax = 0.15
```



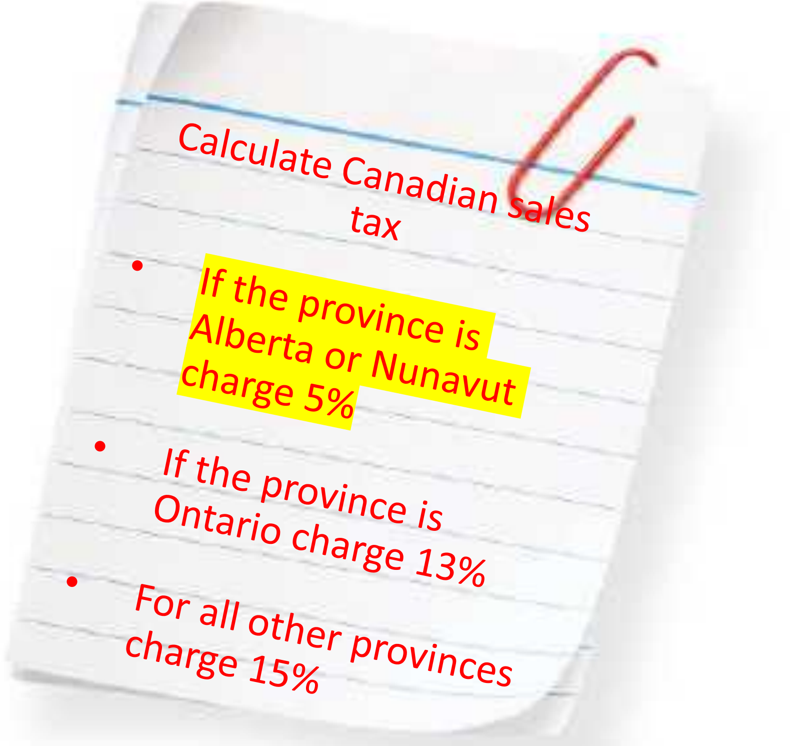
Calculate Canadian sales tax

- If the province is Alberta or Nunavut charge 5%
- If the province is Ontario charge 13%
- For all other provinces charge 15%

Judgement

- If multiple conditions cause the same action they can be combined into a single condition

```
if province == 'Alberta' \
  or province == 'Nunavut':
     tax = 0.05
elif province == 'Ontario':
     tax = 0.13
else:
     tax = 0.15
```



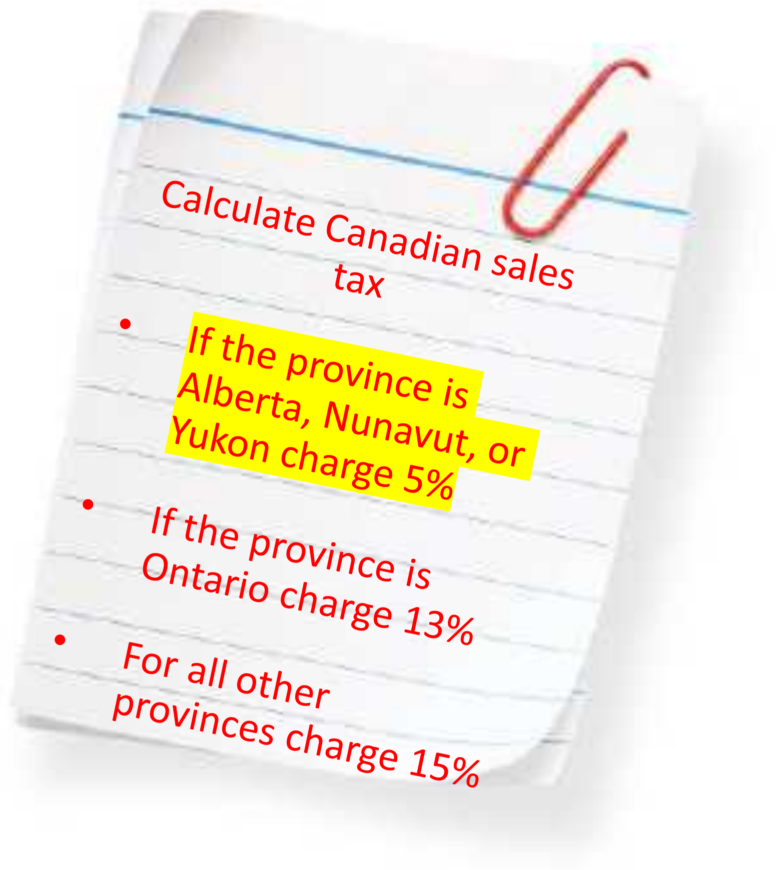
Calculate Canadian sales tax

- If the province is Alberta or Nunavut charge 5%
- If the province is Ontario charge 13%
- For all other provinces charge 15%

Judgement

- If you have a list of possible values to check , you can use the IN operator

```
if province in('Alberta', \
            'Nunavut','Yukon'):
      tax = 0.05
elif province == 'Ontario':
      tax = 0.13
else:
      tax = 0.15
```



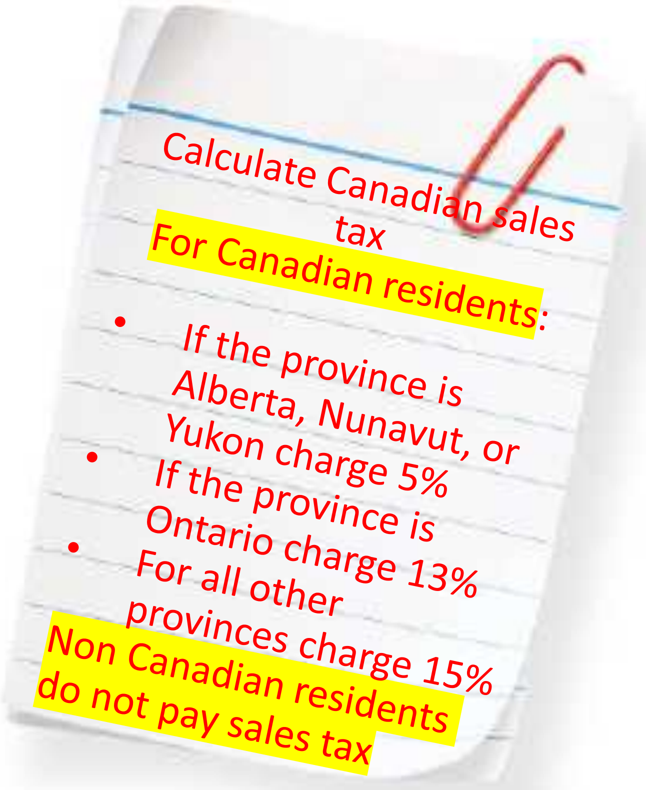
Calculate Canadian sales tax

- If the province is Alberta, Nunavut, or Yukon charge 5%
- If the province is Ontario charge 13%
- For all other provinces charge 15%

Judgement

- If an action depends on a combination of conditions you can nest if statements

```
if country == 'Canada':
    if province in('Alberta',\
        'Nunavut','Yukon'):
        tax = 0.05
    elif province == 'Ontario':
        tax = 0.13
    else:
        tax = 0.15
else:
    tax = 0.0
```



Calculate Canadian sales tax

For Canadian residents:

- If the province is Alberta, Nunavut, or Yukon charge 5%
- If the province is Ontario charge 13%
- For all other provinces charge 15%

Non Canadian residents do not pay sales tax

Judgement

- Sometimes you can combine conditions with AND instead of nesting if statements

```
if gpa >= .85:
    if lowest_grade >= .70:
        print('Well done')
```

```
if gpa >= .85 and lowest_grade >= .70:
    print('Well done')
```

Requirements for honour roll

- Minimum 85% grade point average
- Lowest grade is at least 70%

Judgement

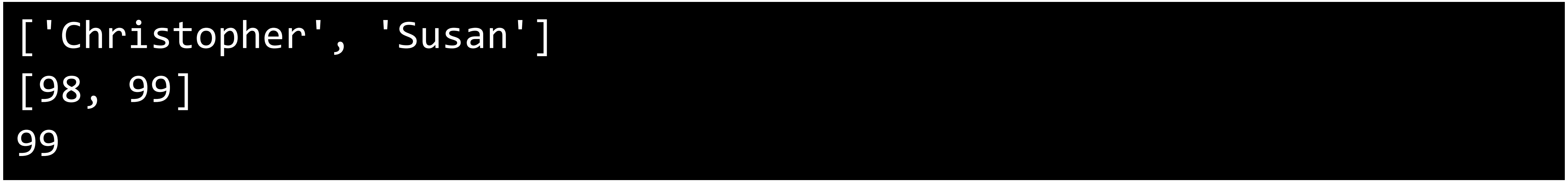
- If you need to remember the results of a condition check later in your code, use Boolean variables as flags

```
if gpa >= .85 and lowest_grade >= .70:
    honour_roll = True
else:
    honour_roll = False
# Somewhere later in your code
if honour_roll:
    print('Well done')
```

Collections

- Lists are collections of items

```
names = ['Christopher', 'Susan']
scores = []
scores.append(98) # Add new item to the end
scores.append(99)
print(names)
print(scores)
print(scores[1]) # Collections are zero-indexed
```



```
['Christopher', 'Susan']
[98, 99]
99
```

Collections

- Arrays are also collections of items

```
from array import array
scores = array('d')
scores.append(97)
scores.append(98)
print(scores)
print(scores[1])
```

```
array('d', [97.0, 98.0])
98.0
```

Collections

- Common operations

```
names = ['Susan', 'Christopher']
print(len(names)) # Get the number of items
names.insert(0, 'Bill') # Insert before index
print(names)
names.sort()
print(names)
```

```
2
['Bill', 'Susan', 'Christopher']
['Bill', 'Christopher', 'Susan']
```

Collections

- Retrieving ranges

```
names = ['Susan', 'Christopher', 'Bill']
presenters = names[0:2] # Get the first two items
# Starting index and number of items to retrieve

print(names)
print(presenters)
```

```
['Susan', 'Christopher', 'Bill']
['Susan', 'Christopher']
```

Collections

- Dictionaries

```
person = {'first': 'Christopher'}
person['last'] = 'Harrison'
print(person)
print(person['first'])
```

```
{'first': 'Christopher', 'last': 'Harrison'}
Christopher
```

Loop

- Loop through a collection

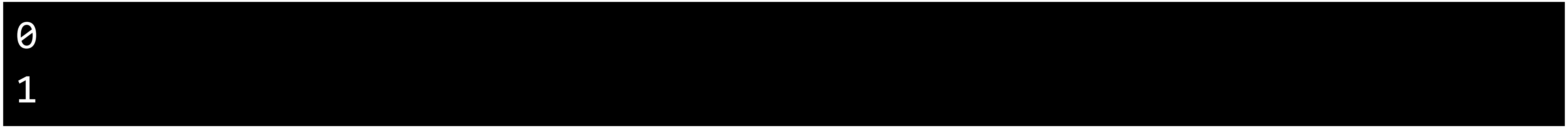
```python
for name in ['Christopher', 'Susan']:
    print(name)
```

```
Christopher
Susan
```

Loop

- Looping a number of times

```
for index in range(0, 2):
    print(index)
```



```
0
1
```

Loop

- Looping with a condition

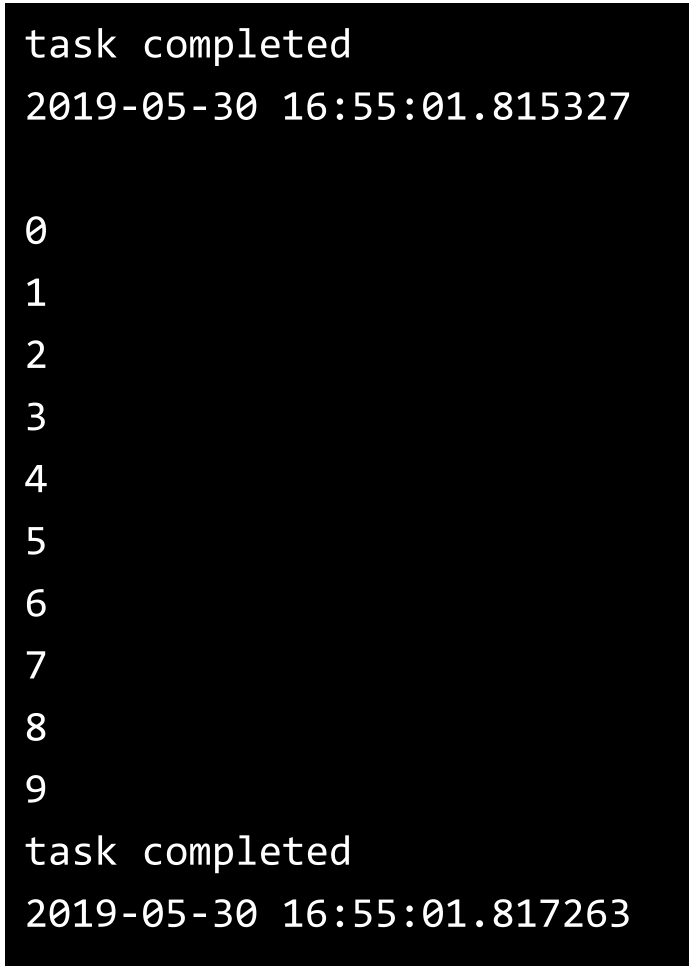
```
names = ['Christopher', 'Susan']
index = 0
while index < len(names):
    print(names[index])
    # Change the condition!!
    index = index + 1
```

```
Christopher
Susan
```

Loop

- Sometimes we copy and paste our code

```
import datetime
# print timestamps to see how long sections of code
# take to run

first_name = 'Susan'
print('task completed')
print(datetime.datetime.now())
print()

for x in range(0,10):
        print(x)
print('task completed')
print(datetime.datetime.now())
print()
```

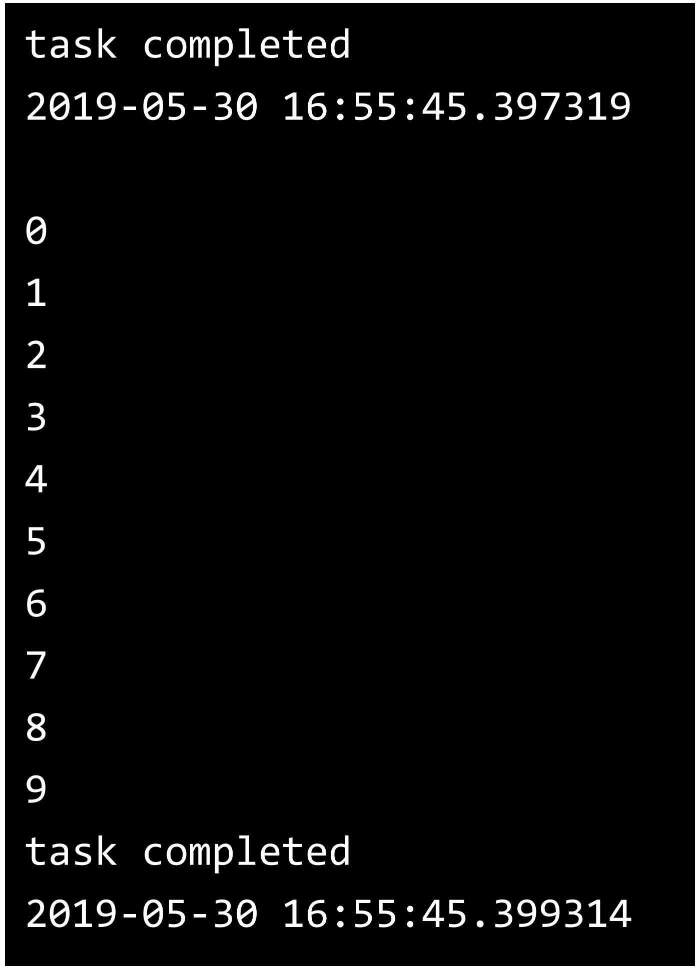


```
task completed
2019-05-30 16:55:01.815327

0
1
2
3
4
5
6
7
8
9
task completed
2019-05-30 16:55:01.817263
```

Loop

- Use functions instead of repeating code

```
import datetime
# Print the current time
def print_time():
	print('task completed')
	print(datetime.datetime.now())
	print()

first_name = 'Susan'
print_time()

for x in range(0,10):
	print(x)
print_time()
```



```
task completed
2019-05-30 16:55:45.397319

0
1
2
3
4
5
6
7
8
9
task completed
2019-05-30 16:55:45.399314
```

Loop

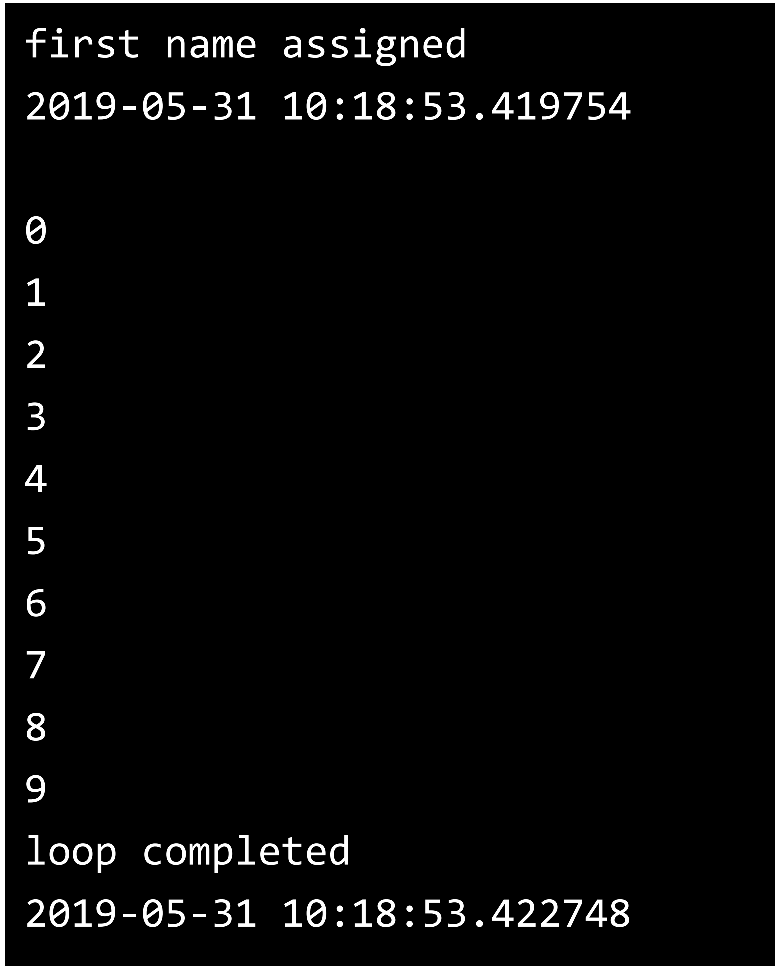
- By moving the code to a function, you reduce rework and the chance of introducing bugs when you change the code you had copied

```
# Import the datetime class from datetime
library
from datetime import datetime
# Print the current time
def print_time():
    print('task completed')
    # Now I don't need the extra datetime
prefix
    print(datetime.now())
    print()
```

Loop

- What if I want a different message displayed?

```
from datetime import datetime
# print timestamps to see how long sections of code
# take to run

first_name = 'Susan'
print('first name assigned')
print(datetime.now())
print()

for x in range(0,10):
	print(x)
print('loop completed')
print(datetime.now())
print()
```

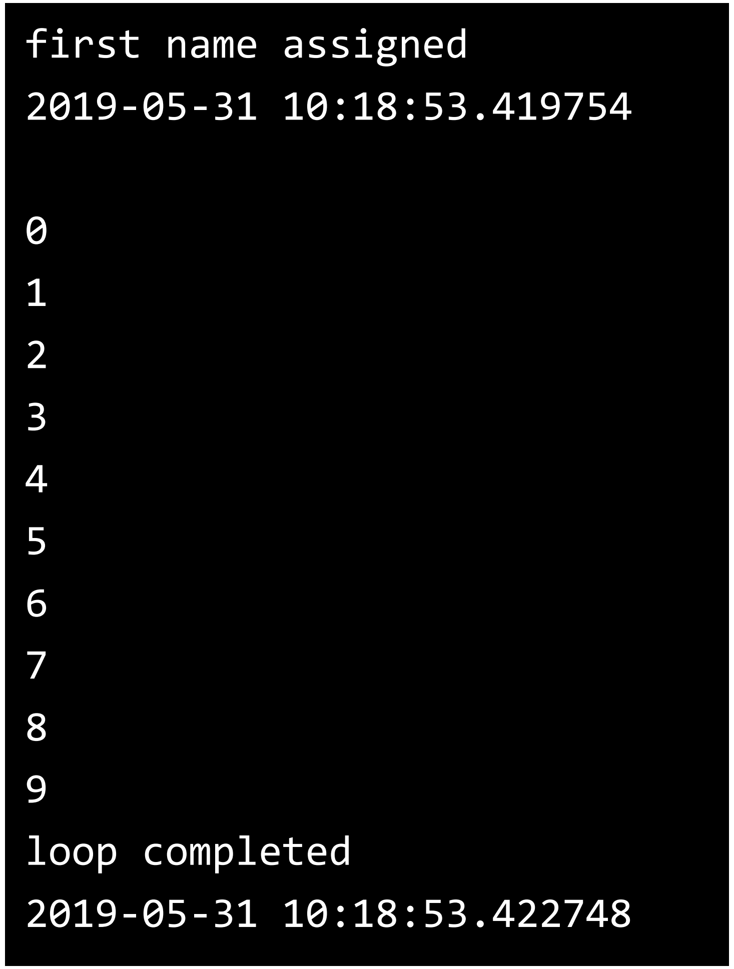


```
first name assigned
2019-05-31 10:18:53.419754

0
1
2
3
4
5
6
7
8
9
loop completed
2019-05-31 10:18:53.422748
```

Loop

- Pass the task name as a parameter

```
from datetime import datetime

# Print the current time and task name
def print_time(task_name):
        print(task_name)
        print(datetime.now())
        print()

first_name = 'Susan'
print_time('first name assigned')

for x in range(0,10):
    print(x)
print_time('loop completed')
```

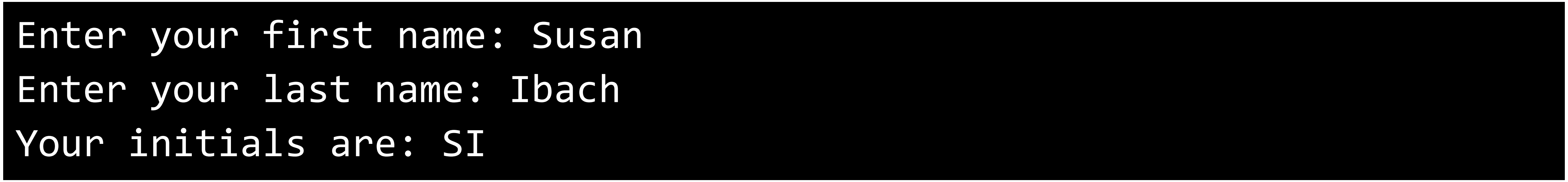


```
first name assigned
2019-05-31 10:18:53.419754

0
1
2
3
4
5
6
7
8
9
loop completed
2019-05-31 10:18:53.422748
```

Input

- Here's another example where the code looks different but we are doing the same logic over and over

```
first_name = input('Enter your first name: ')
first_name_initial = first_name[0:1]
last_name = input('Enter your last name: ')
last_name_initial = last_name[0:1]

print('Your initials are: ' + first_name_initial \
      + last_name_initial)
```

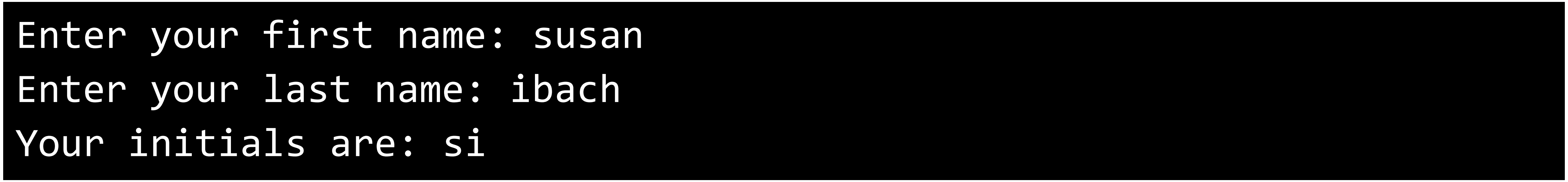


```
Enter your first name: Susan
Enter your last name: Ibach
Your initials are: SI
```

Input

- I can still use a function, but this time my function returns a value

```
def get_initial(name):
        initial = name[0:1]
        return initial

first_name = input('Enter your first name: ')
first_name_initial = get_initial(first_name)

last_name = input('Enter your last name: ')
last_name_initial = get_initial(last_name)

print('Your initials are: ' + first_name_initial \
        + last_name_initial)
```

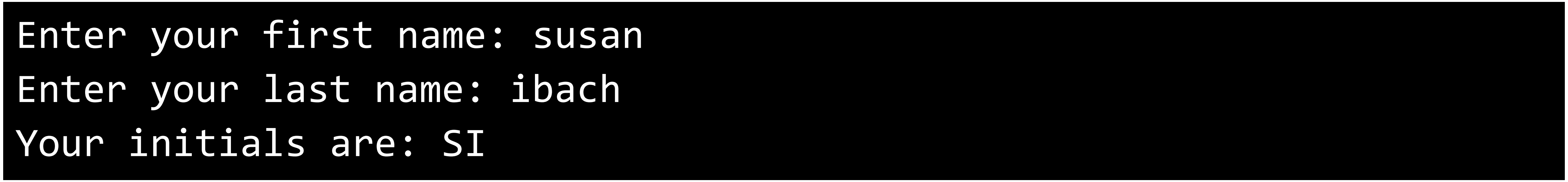


```
Enter your first name: susan
Enter your last name: ibach
Your initials are: si
```

Input

- If you need to change something you only have to change it in one place!

```
def get_initial(name):
    initial = name[0:1].upper()
    return initial

first_name = input('Enter your first name: ')
first_name_initial = get_initial(first_name)

last_name = input('Enter your last name: ')
last_name_initial = get_initial(last_name)
```

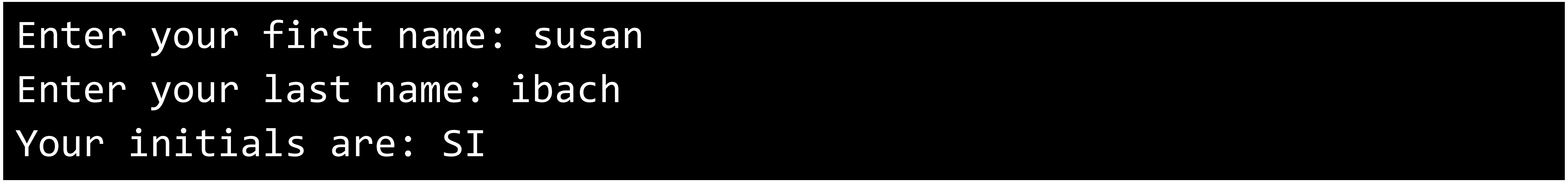


```
Enter your first name: susan
Enter your last name: ibach
Your initials are: SI
```

Input

- Functions that return values allow clever code, but you might trade readability for less code

```
def get_initial(name):
    initial = name[0:1].upper()
    return initial

first_name = input('Enter your first name: ')
last_name = input('Enter your last name: ')

print('Your initials are: ' \
    + get_initial(first_name) \
    + get_initial(last_name))
```

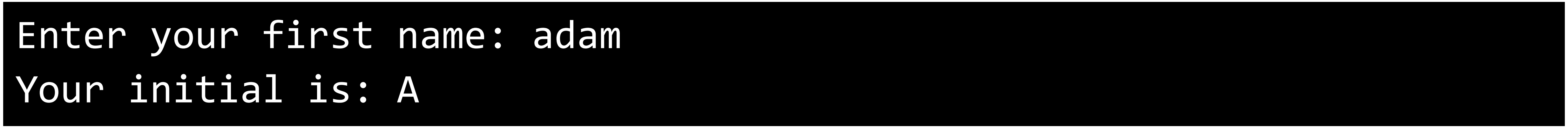


```
Enter your first name: susan
Enter your last name: ibach
Your initials are: SI
```

Input

- We already learned to create functions which accept a parameter and return values

```
def get_initial(name):
    initial = name[0:1].upper()
    return initial

first_name = input('Enter your first name: ')
first_name_initial = get_initial(first_name)

print('Your initial is: ' + first_name_initial)
```



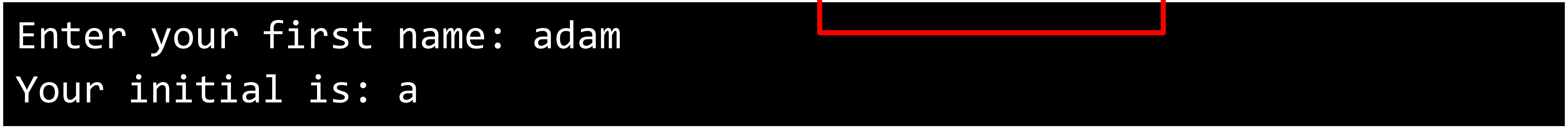
```
Enter your first name: adam
Your initial is: A
```

Function

- Functions can accept multiple parameters

```
def get_initial(name, force_uppercase):
    if force_uppercase:
        initial = name[0:1].upper()
    else:
        initial = name[0:1]
    return initial

first_name = input('Enter your first name: ')
first_name_initial = get_initial(first_name, False)

print('Your initial is: ' + first_name_initial)
```

Pass the parameters in the same order they are listed in the function declaration

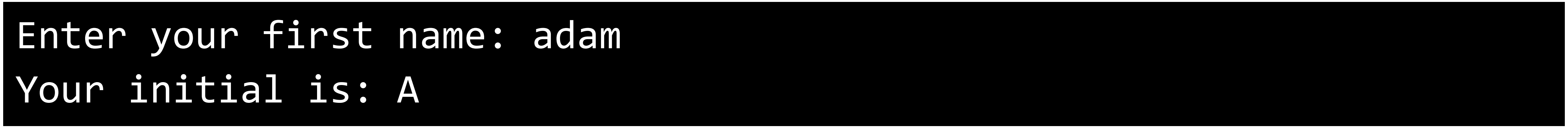


```
Enter your first name: adam
Your initial is: a
```

Function

- You can specify a default value for a parameter

```
def get_initial(name, force_uppercase=True):
    if force_uppercase:
        initial = name[0:1].upper()
    else:
       initial = name[0:1]
    return initial

first_name = input('Enter your first name: ')
first_name_initial = get_initial(first_name)

print('Your initial is: ' + first_name_initial)
```

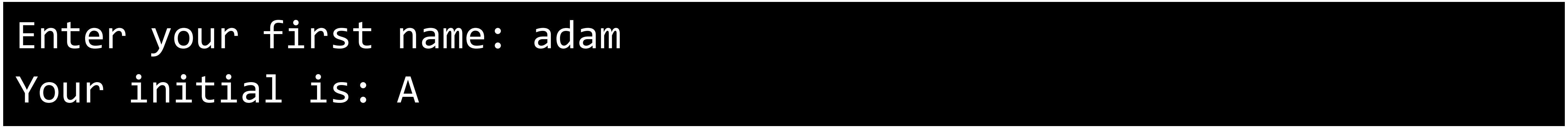


```
Enter your first name: adam
Your initial is: A
```

Function

- You can also assign the values to parameters by name when you call the function

```
def get_initial(name, force_uppercase):
    if force_uppercase:
        initial = name[0:1].upper()
    else:
        initial = name[0:1]
    return initial

first_name = input('Enter your first name: ')
first_name_initial = get_initial(force_uppercase=True,\
                        name=first_name)

print('Your initial is: ' + first_name_initial)
```



```
Enter your first name: adam
Your initial is: A
```

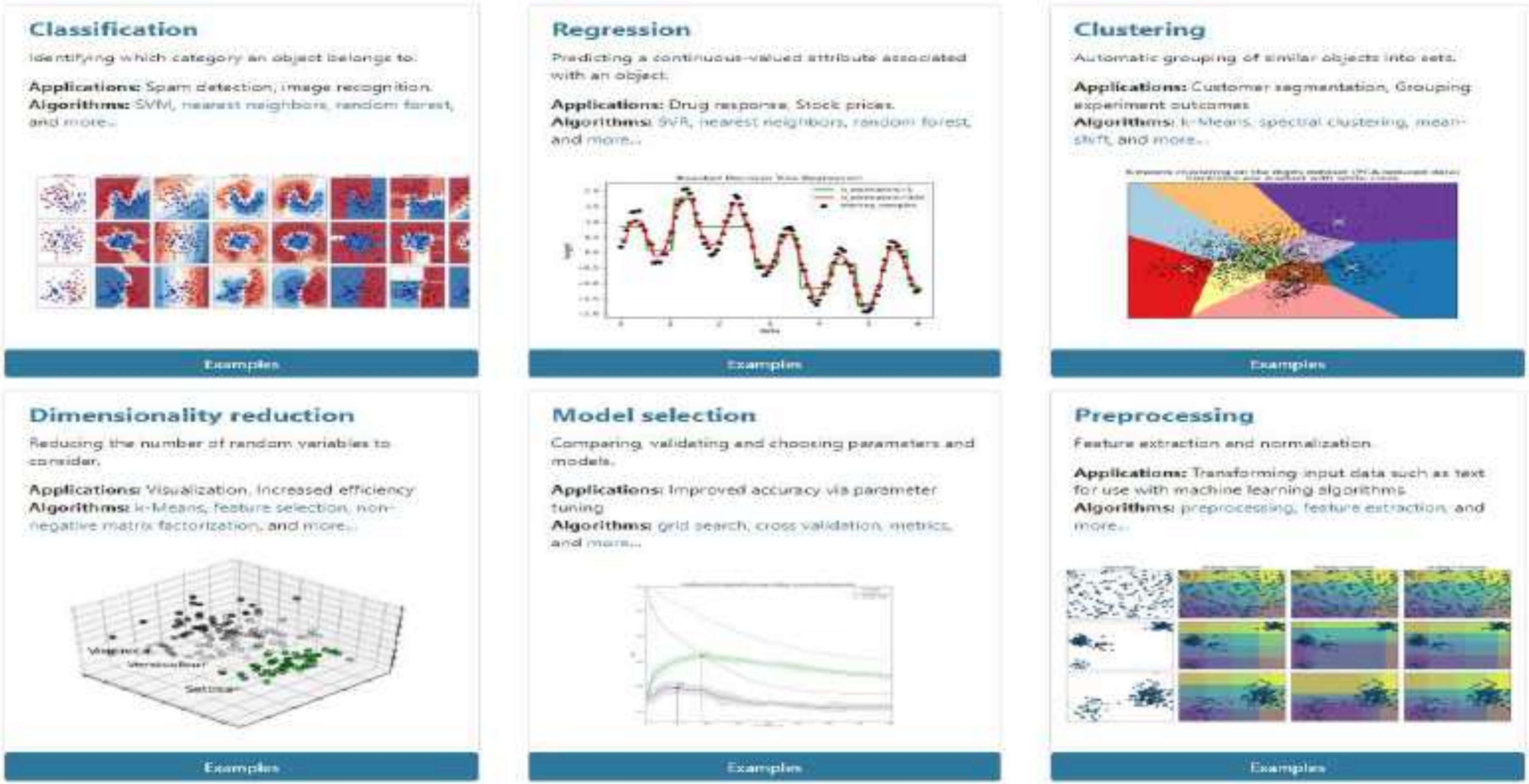
Scikit-learn

What is scikit-learn?

Scikit-learn is a Python module for **machine learning** built on top of SciPy and is distributed under the 3-Clause BSD license. The project was started in 2007 by David Cournapeau as a Google Summer of Code project, and since then many volunteers have contributed. (https://github.com/scikit-learn/scikit-learn)

Scikit-learn (formerly **scikits.learn** and also known as **sklearn**) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. (**Wikipedia**)

Scikit-learn



Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...

Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...

Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...

Examples

https://scikit-learn.org/stable/

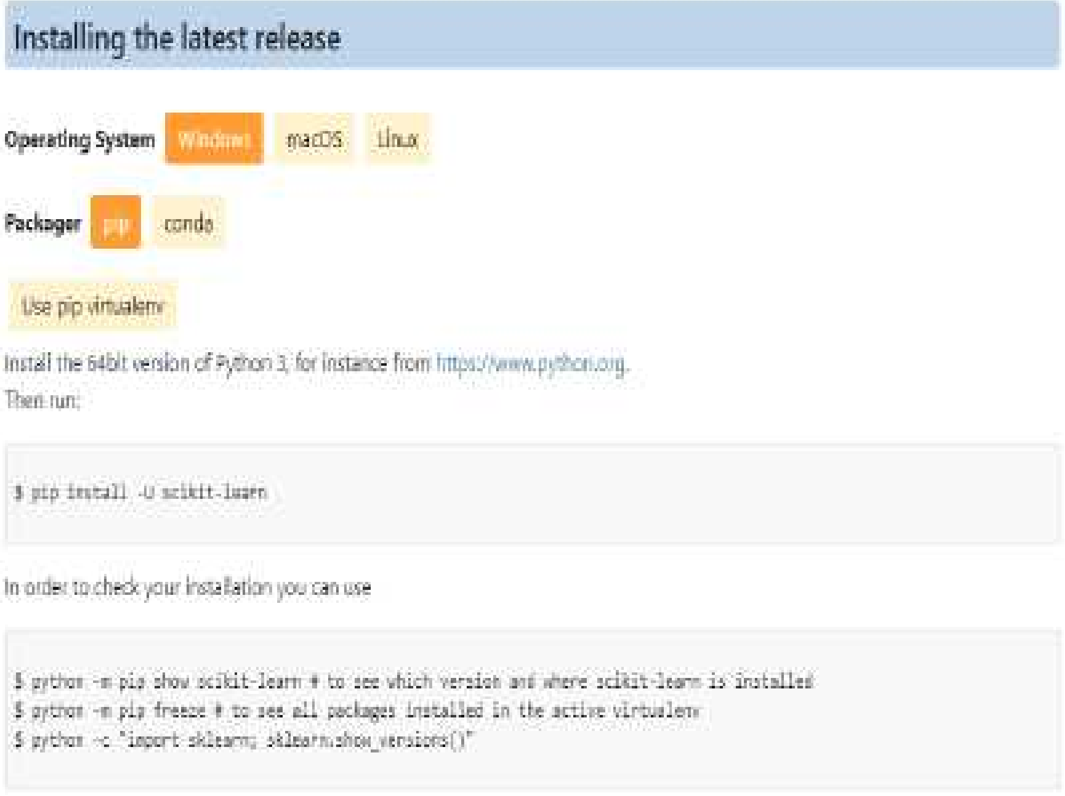
Scikit-learn

Installing scikit-learn

There are different ways to install scikit-learn:

- Install the latest official release. This is the best approach for most users. It will provide a stable version and pre-built packages are available for most platforms.
- Install the version of scikit-learn provided by your operating system or Python distribution. This is a quick option for those who have operating systems or Python distributions that distribute scikit-learn. It might not provide the latest release version.
- Building the package from source. This is best for users who want the latest-and-greatest features and aren't afraid of running brand-new code. This is also needed for users who wish to contribute to the project.

https://scikit-learn.org/stable/install.html



Installing the latest release

Install the 64bit version of Python 3, for instance from https://www.python.org.

Then run:

```
$ pip install -U scikit-learn
```

In order to check your installation you can use

```
$ python -m pip show scikit-learn # to see which version and where scikit-learn is installed
$ python -m pip freeze # to see all packages installed in the active virtualenv
$ python -c "import sklearn; sklearn.show_versions()"
```

Scikit-learn

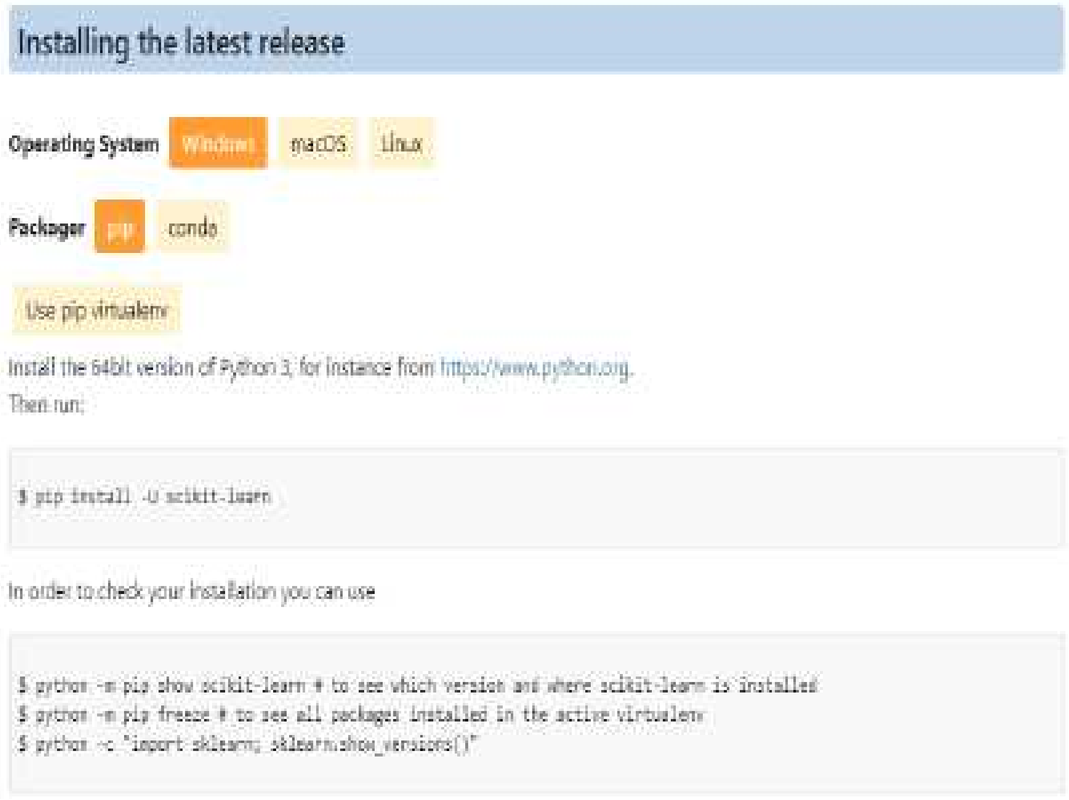
Installing scikit-learn

There are different ways to install scikit-learn:

•Install the latest official release. This is the best approach for most users. It will provide a stable version and pre-built packages are available for most platforms.

•Install the version of scikit-learn provided by your operating system or Python distribution. This is a quick option for those who have operating systems or Python distributions tha distribute scikit-learn. It might not provide the latest release version.

•Building the package from source. This is best for users who want the latest-and-greatest features and aren't afraid of running brand-new code. This is also needed for users who wish to contribute to the project.



Installing the latest release

Install the 64bit version of Python 3, for instance from https://www.python.org.

Then run:

```
$ pip install -U scikit-learn
```

In order to check your installation you can use

```
$ python -m pip show scikit-learn # to see which version and where scikit-learn is installed
$ python -m pip freeze # to see all packages installed in the active virtualenv
$ python -c "import sklearn; sklearn.show_versions()"
```

URL: https://scikit-learn.org/stable/install.html

Scikit-learn

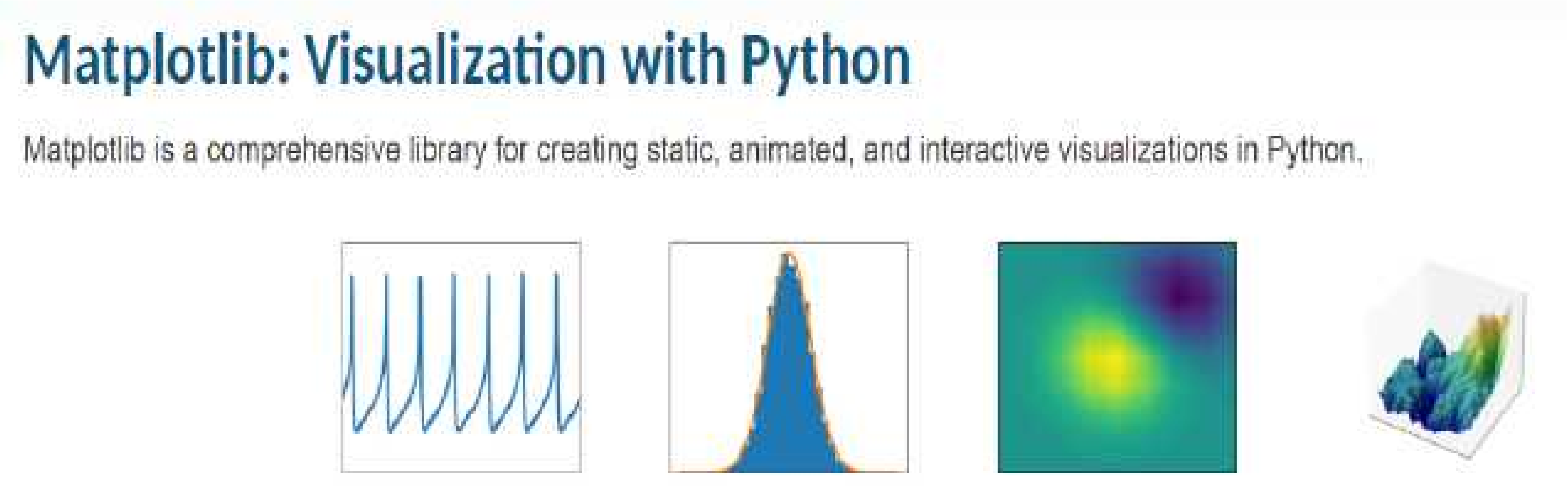
Dependencies

scikit-learn requires:

- Python
- NumPy
- SciPy
- joblib
- threadpoolctl
- **Matplotlib**
- **Seaborn**
- Jupyter

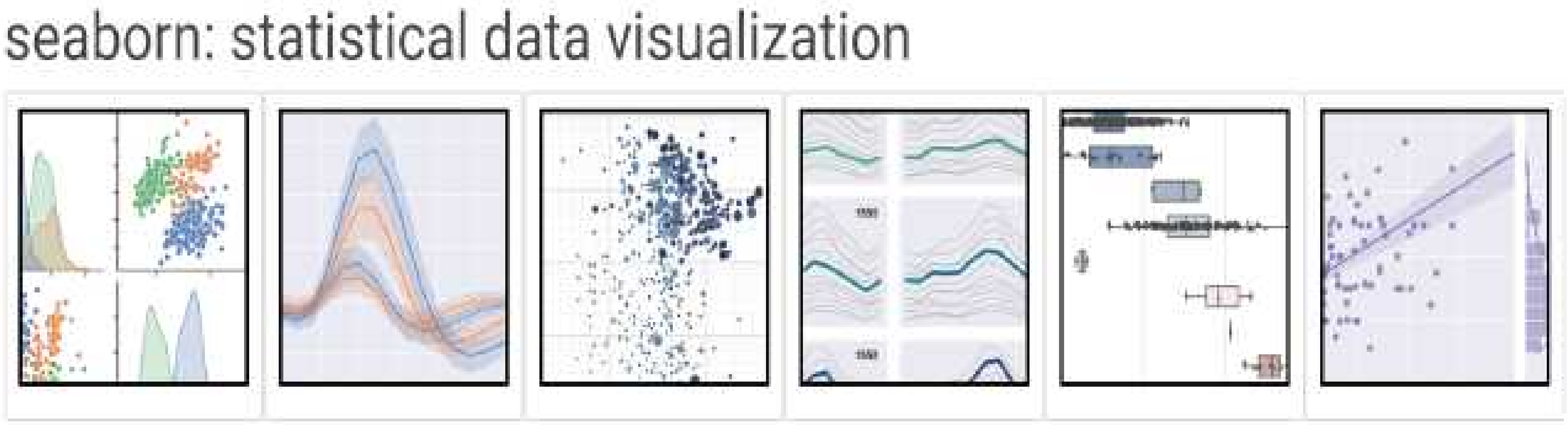
http://seaborn.pydata.org/index.html

https://matplotlib.org/



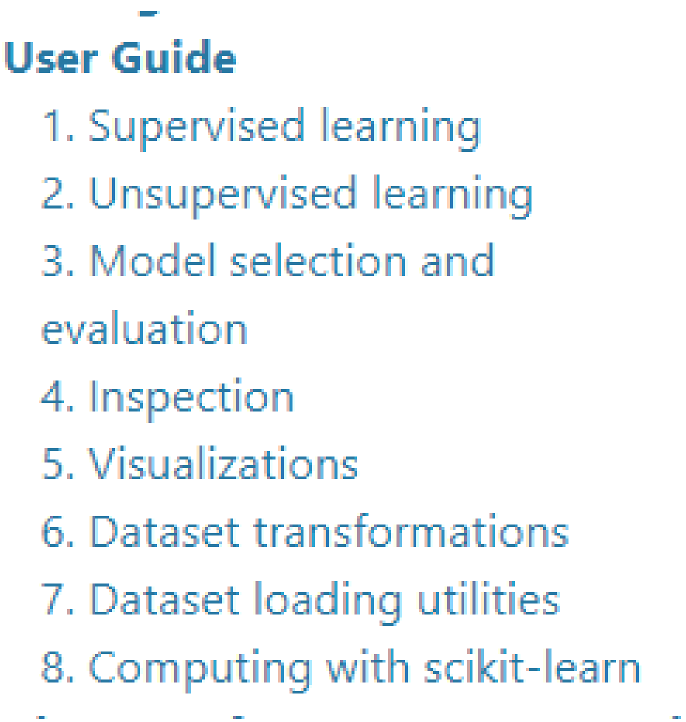
Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.



seaborn: statistical data visualization

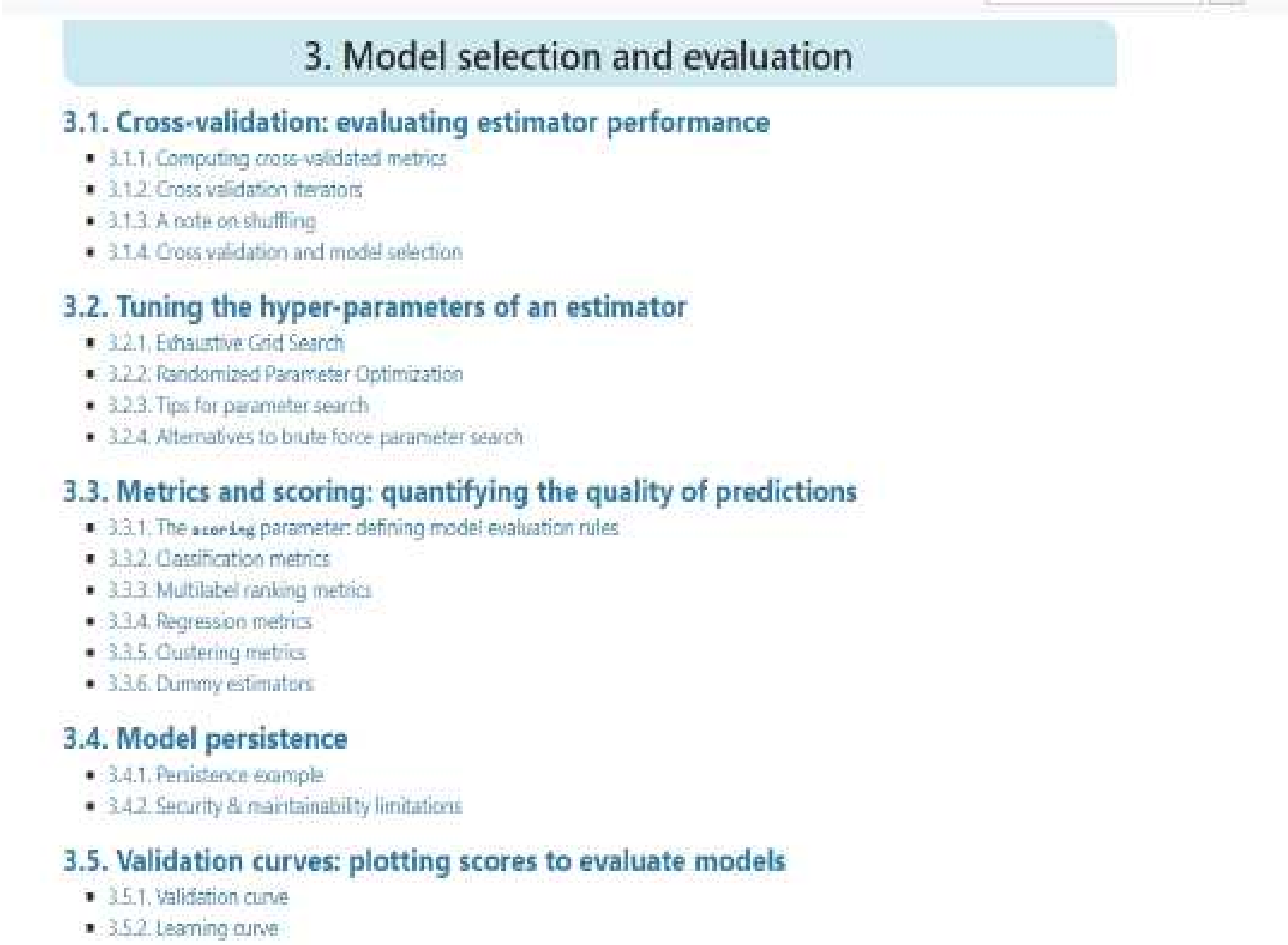
Scikit-learn



User Guide

1. Supervised learning
2. Unsupervised learning
3. Model selection and evaluation
4. Inspection
5. Visualizations
6. Dataset transformations
7. Dataset loading utilities
8. Computing with scikit-learn

https://scikit-learn.org/stable/user_guide.html



3. Model selection and evaluation

3.1. Cross-validation: evaluating estimator performance

- 3.1.1. Computing cross-validated metrics
- 3.1.2. Cross validation iterators
- 3.1.3. A note on shuffling
- 3.1.4. Cross validation and model selection

3.2. Tuning the hyper-parameters of an estimator

- 3.2.1. Exhaustive Grid Search
- 3.2.2. Randomized Parameter Optimization
- 3.2.3. Tips for parameter search
- 3.2.4. Alternatives to brute force parameter search

3.3. Metrics and scoring: quantifying the quality of predictions

- 3.3.1. The `scoring` parameter: defining model evaluation rules
- 3.3.2. Classification metrics
- 3.3.3. Multilabel ranking metrics
- 3.3.4. Regression metrics
- 3.3.5. Clustering metrics
- 3.3.6. Dummy estimators

3.4. Model persistence

- 3.4.1. Persistence example
- 3.4.2. Security & maintainability limitations

3.5. Validation curves: plotting scores to evaluate models

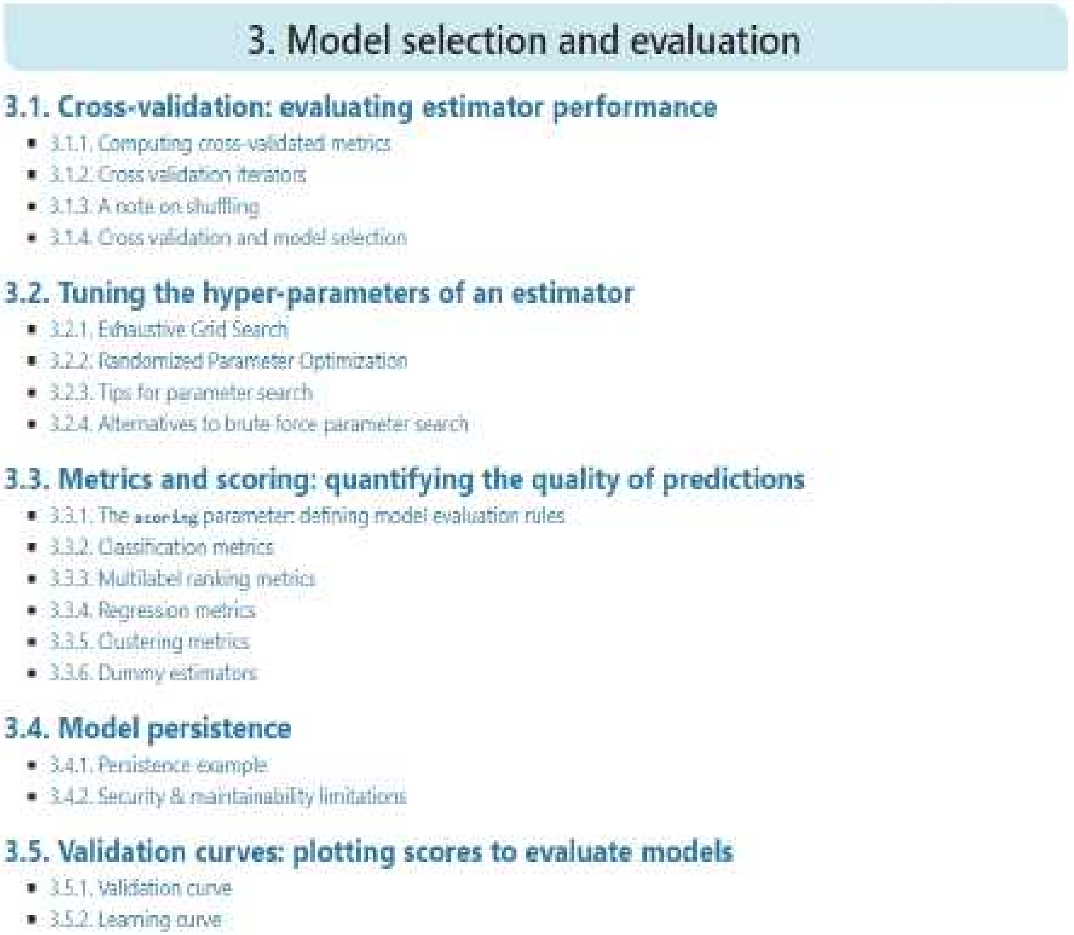
- 3.5.1. Validation curve
- 3.5.2. Learning curve

Scikit-learn

User Guide

1. Supervised learning
2. Unsupervised learning
3. Model selection and evaluation
4. Inspection
5. Visualizations
6. Dataset transformations
7. Dataset loading utilities
8. Computing with scikit-learn

https://scikit-learn.org/stable/user_guide.html



3. Model selection and evaluation

3.1. Cross-validation: evaluating estimator performance

- 3.1.1. Computing cross-validated metrics
- 3.1.2. Cross validation iterators
- 3.1.3. A note on shuffling
- 3.1.4. Cross validation and model selection

3.2. Tuning the hyper-parameters of an estimator

- 3.2.1. Exhaustive Grid Search
- 3.2.2. Randomized Parameter Optimization
- 3.2.3. Tips for parameter search
- 3.2.4. Alternatives to brute force parameter search

3.3. Metrics and scoring: quantifying the quality of predictions

- 3.3.1. The `scoring` parameter: defining model evaluation rules
- 3.3.2. Classification metrics
- 3.3.3. Multilabel ranking metrics
- 3.3.4. Regression metrics
- 3.3.5. Clustering metrics
- 3.3.6. Dummy estimators

3.4. Model persistence

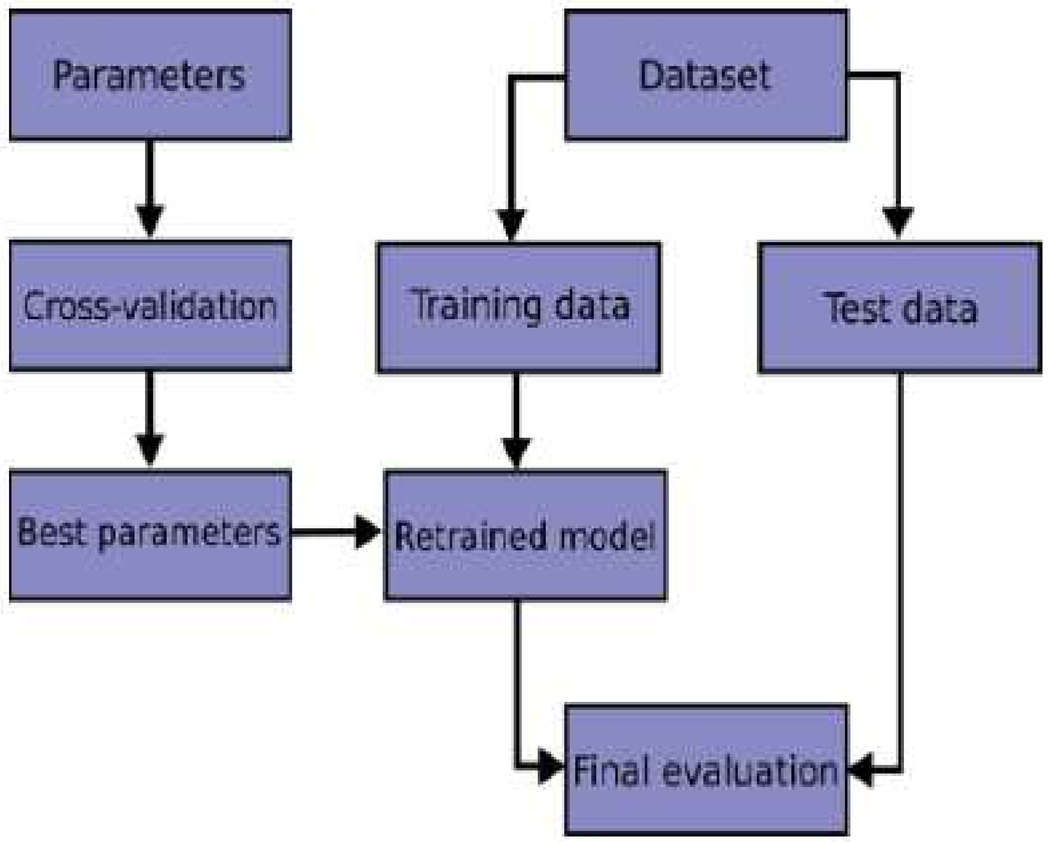
- 3.4.1. Persistence example
- 3.4.2. Security & maintainability limitations

3.5. Validation curves: plotting scores to evaluate models

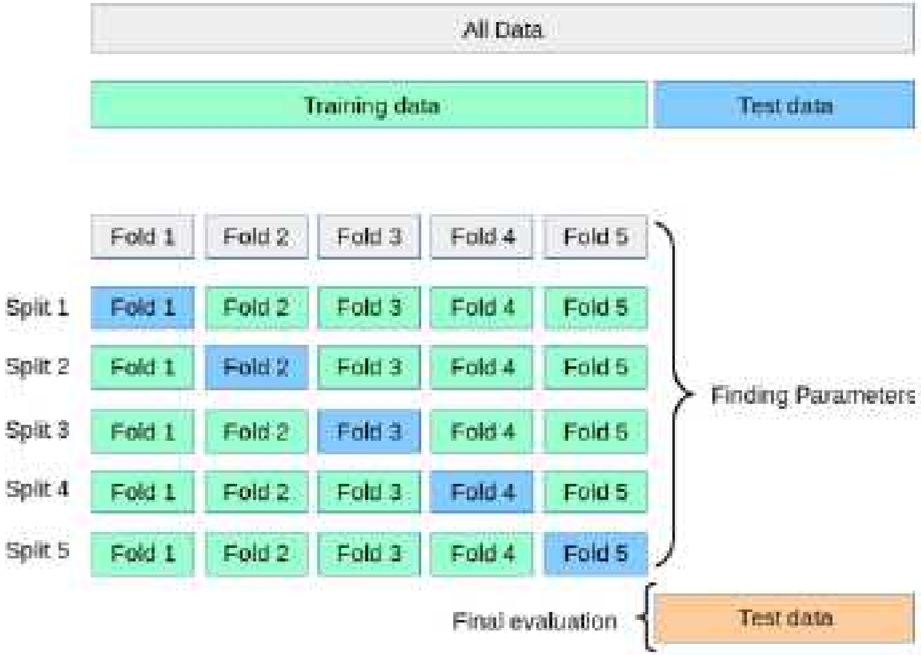
- 3.5.1. Validation curve
- 3.5.2. Learning curve

Cross-validation: evaluating estimator performance

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called overfitting. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set X_test, y_test. Note that the word "experiment" is not intended to denote academic use only, because even in commercial settings machine learning usually starts out experimentally. Here is a flowchart of typical cross validation workflow in model training.



Scikit-learn



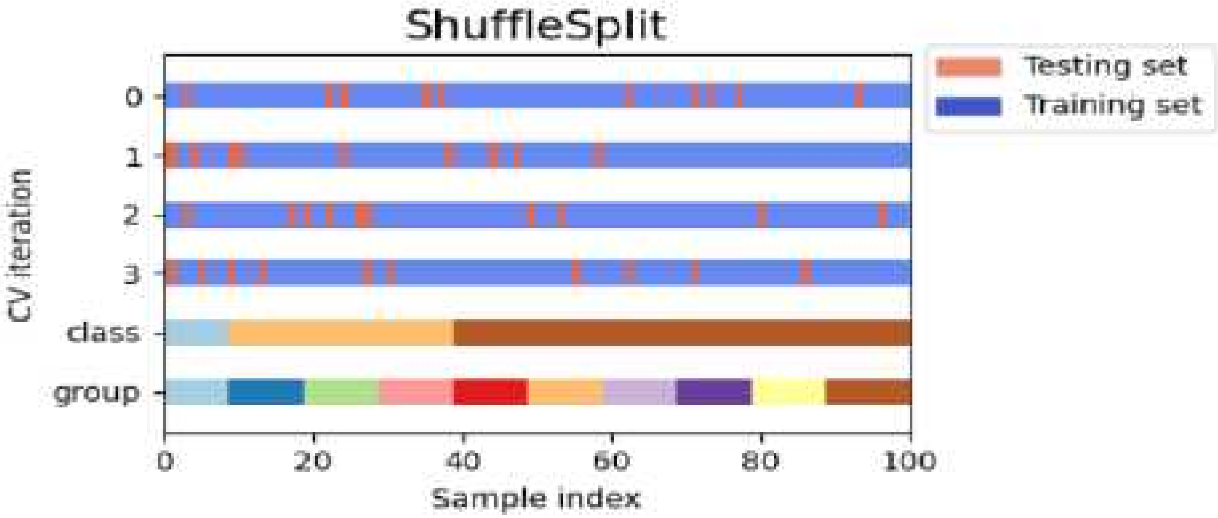
All Data

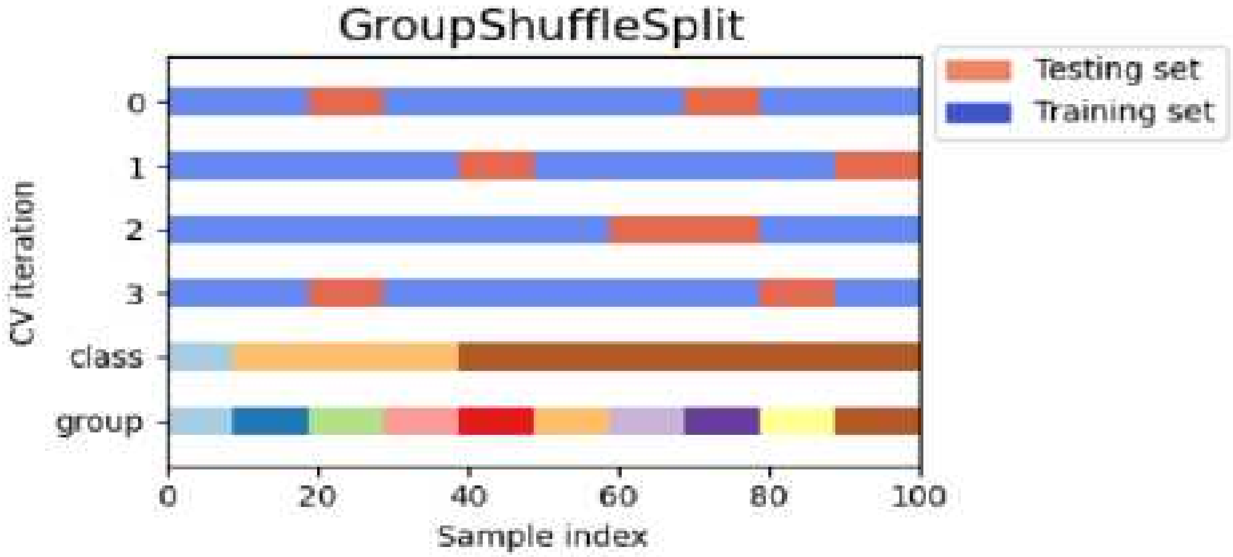
Training data	Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Finding Parameters

Final evaluation — Test data

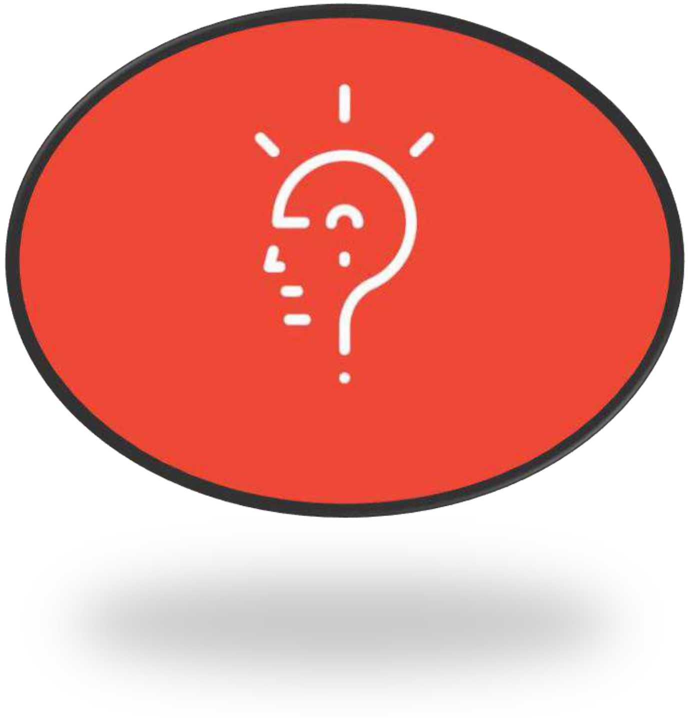




k-fold CV

https://scikit-learn.org/stable/modules/cross_validation.html#computing-cross-validated-metrics

Any Question?



Metrics and scoring: quantifying the quality of predictions

There are 3 different APIs for evaluating the quality of a model's predictions:

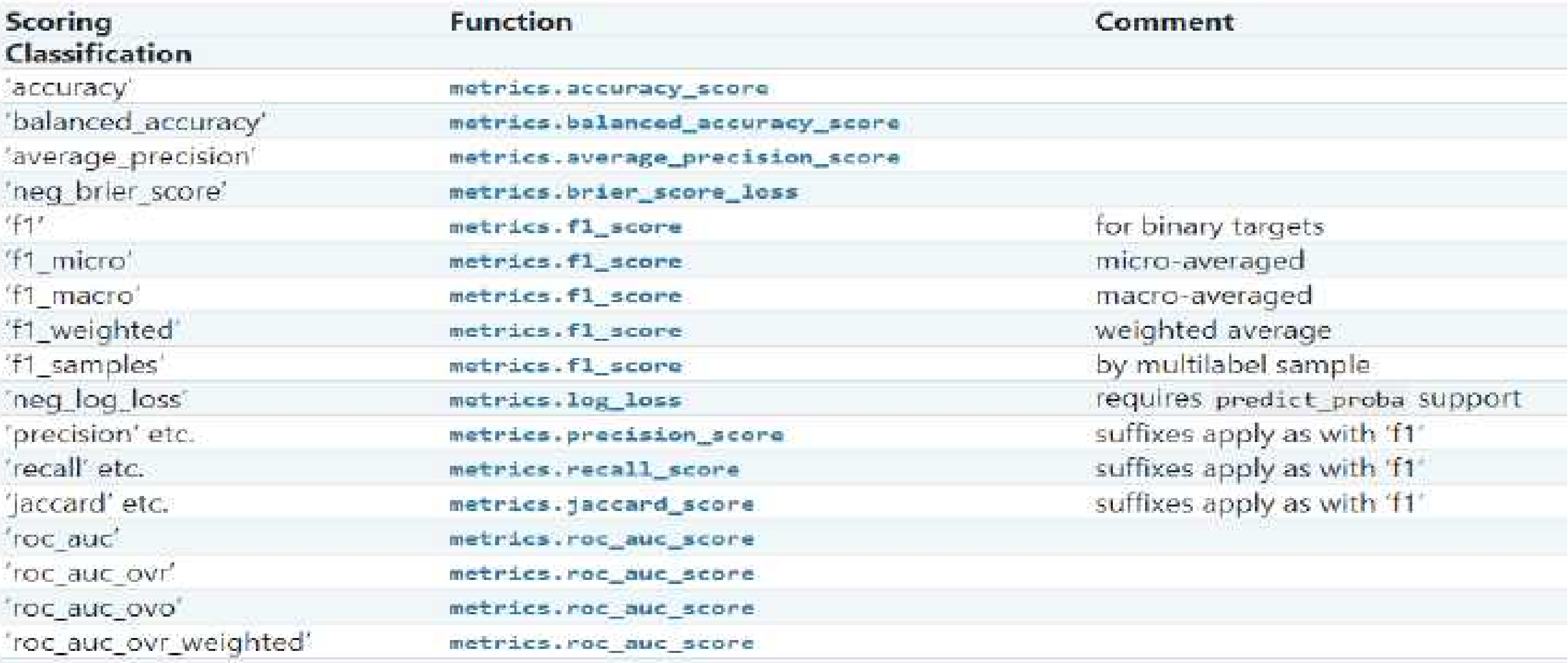
- Estimator score method: Estimators have a score method providing a default evaluation criterion for the problem they are designed to solve. This is not discussed on this page, but in each estimator's documentation.

- Scoring parameter: Model-evaluation tools using cross-validation (such as model_selection. cross_val_score and model_selection.GridSearchCV) rely on an internal scoring strategy. This is discussed in the section The scoring parameter: defining model evaluation rules.

- **Metric functions:** The metrics module implements functions assessing prediction error for specific purposes. These metrics are detailed in sections on Classification metrics, Multilabel ranking metrics, Regression metrics and Clustering metrics.

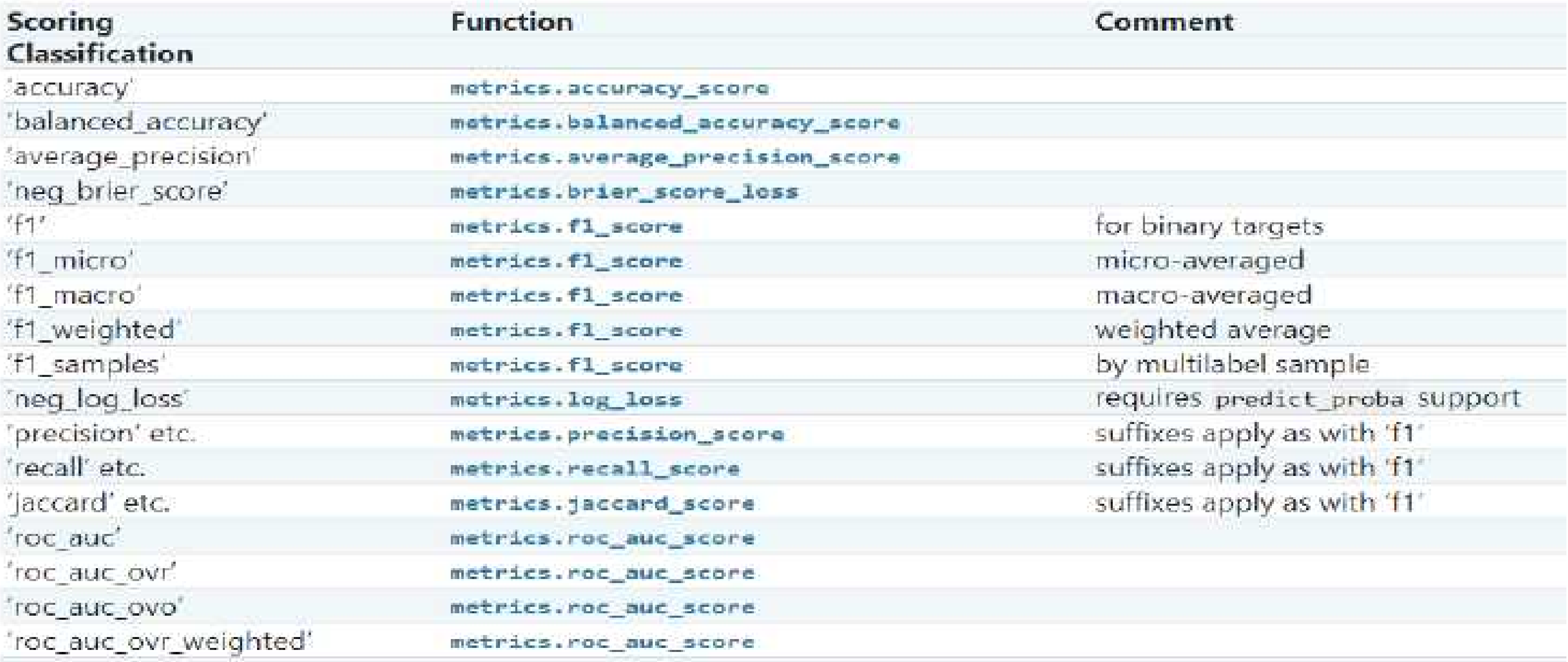
https://scikit-learn.org/stable/modules/model_evaluation.html

Classification metrics



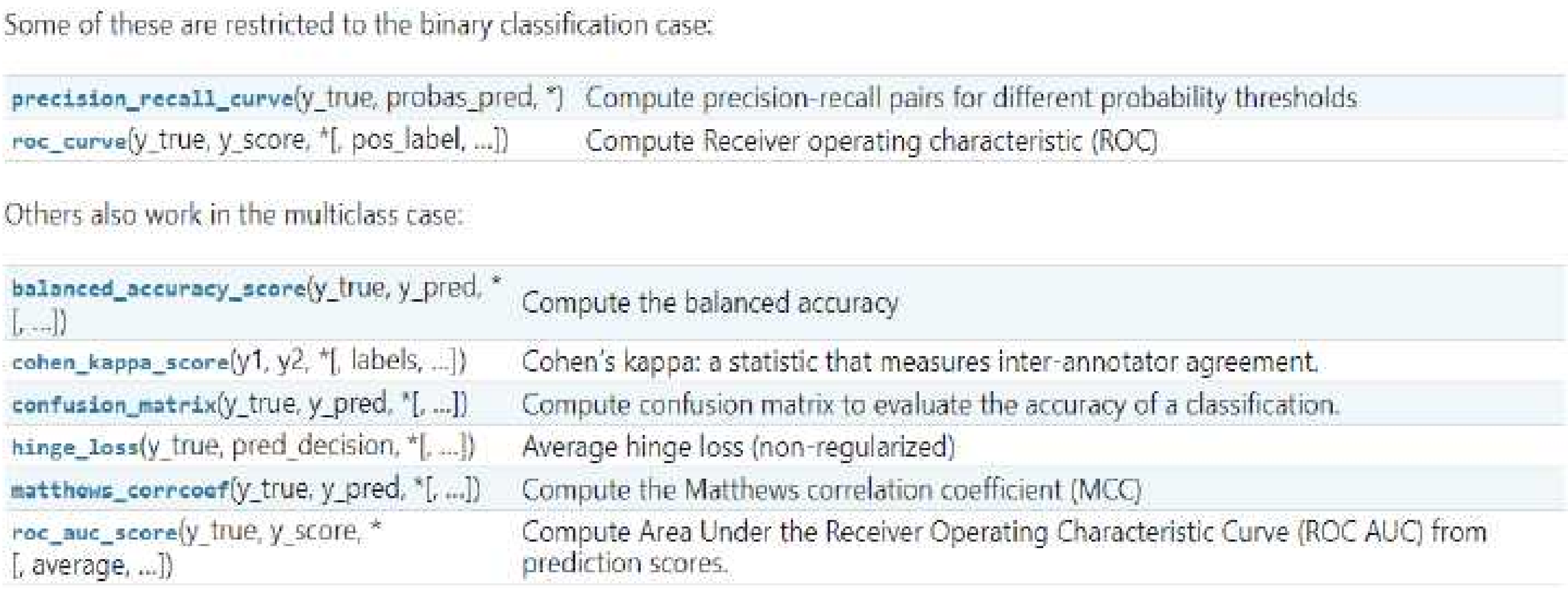
Scoring Classification	Function	Comment
'accuracy'	`metrics.accuracy_score`	
'balanced_accuracy'	`metrics.balanced_accuracy_score`	
'average_precision'	`metrics.average_precision_score`	
'neg_brier_score'	`metrics.brier_score_loss`	
'f1'	`metrics.f1_score`	for binary targets
'f1_micro'	`metrics.f1_score`	micro-averaged
'f1_macro'	`metrics.f1_score`	macro-averaged
'f1_weighted'	`metrics.f1_score`	weighted average
'f1_samples'	`metrics.f1_score`	by multilabel sample
'neg_log_loss'	`metrics.log_loss`	requires `predict_proba` support
'precision' etc.	`metrics.precision_score`	suffixes apply as with 'f1'
'recall' etc.	`metrics.recall_score`	suffixes apply as with 'f1'
'jaccard' etc.	`metrics.jaccard_score`	suffixes apply as with 'f1'
'roc_auc'	`metrics.roc_auc_score`	
'roc_auc_ovr'	`metrics.roc_auc_score`	
'roc_auc_ovo'	`metrics.roc_auc_score`	
'roc_auc_ovr_weighted'	`metrics.roc_auc_score`	

Classification metrics



Scoring	Function	Comment
Classification		
'accuracy'	`metrics.accuracy_score`	
'balanced_accuracy'	`metrics.balanced_accuracy_score`	
'average_precision'	`metrics.average_precision_score`	
'neg_brier_score'	`metrics.brier_score_loss`	
'f1'	`metrics.f1_score`	for binary targets
'f1_micro'	`metrics.f1_score`	micro-averaged
'f1_macro'	`metrics.f1_score`	macro-averaged
'f1_weighted'	`metrics.f1_score`	weighted average
'f1_samples'	`metrics.f1_score`	by multilabel sample
'neg_log_loss'	`metrics.log_loss`	requires `predict_proba` support
'precision' etc.	`metrics.precision_score`	suffixes apply as with 'f1'
'recall' etc.	`metrics.recall_score`	suffixes apply as with 'f1'
'jaccard' etc.	`metrics.jaccard_score`	suffixes apply as with 'f1'
'roc_auc'	`metrics.roc_auc_score`	
'roc_auc_ovr'	`metrics.roc_auc_score`	
'roc_auc_ovo'	`metrics.roc_auc_score`	
'roc_auc_ovr_weighted'	`metrics.roc_auc_score`	

Classification metrics



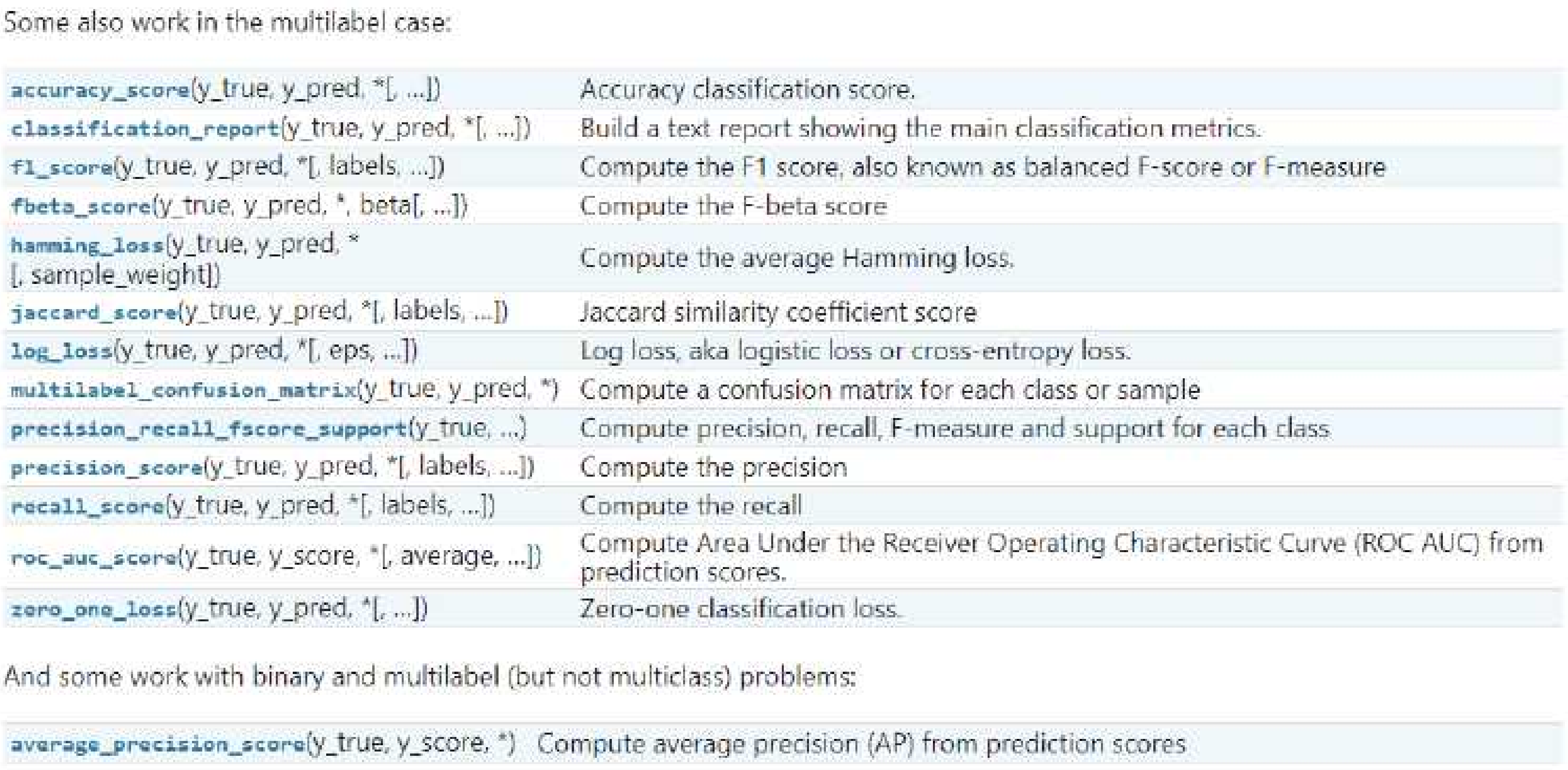
Some of these are restricted to the binary classification case:

precision_recall_curve(y_true, probas_pred, *)	Compute precision-recall pairs for different probability thresholds
roc_curve(y_true, y_score, *[, pos_label, ...])	Compute Receiver operating characteristic (ROC)

Others also work in the multiclass case:

balanced_accuracy_score(y_true, y_pred, *[, ...])	Compute the balanced accuracy
cohen_kappa_score(y1, y2, *[, labels, ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
confusion_matrix(y_true, y_pred, *[, ...])	Compute confusion matrix to evaluate the accuracy of a classification.
hinge_loss(y_true, pred_decision, *[, ...])	Average hinge loss (non-regularized)
matthews_corrcoef(y_true, y_pred, *[, ...])	Compute the Matthews correlation coefficient (MCC)
roc_auc_score(y_true, y_score, *[, average, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

Classification metrics



Some also work in the multilabel case:

accuracy_score(y_true, y_pred, *[, ...])	Accuracy classification score.
classification_report(y_true, y_pred, *[, ...])	Build a text report showing the main classification metrics.
f1_score(y_true, y_pred, *[, labels, ...])	Compute the F1 score, also known as balanced F-score or F-measure
fbeta_score(y_true, y_pred, *, beta[, ...])	Compute the F-beta score
hamming_loss(y_true, y_pred, * [, sample_weight])	Compute the average Hamming loss.
jaccard_score(y_true, y_pred, *[, labels, ...])	Jaccard similarity coefficient score
log_loss(y_true, y_pred, *[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
multilabel_confusion_matrix(y_true, y_pred, *)	Compute a confusion matrix for each class or sample
precision_recall_fscore_support(y_true, ...)	Compute precision, recall, F-measure and support for each class
precision_score(y_true, y_pred, *[, labels, ...])	Compute the precision
recall_score(y_true, y_pred, *[, labels, ...])	Compute the recall
roc_auc_score(y_true, y_score, *[, average, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
zero_one_loss(y_true, y_pred, *[, ...])	Zero-one classification loss.

And some work with binary and multilabel (but not multiclass) problems:

average_precision_score(y_true, y_score, *)	Compute average precision (AP) from prediction scores

Classification metrics

In extending a binary metric to multiclass or multilabel problems, the data is treated as a collection of binary problems, one for each class. There are then a number of ways to average binary metric calculations across the set of classes, each of which may be useful in some scenario. Where available, you should select among these using the average parameter.

"macro" simply calculates the mean of the binary metrics, giving equal weight to each class. In problems where infrequent classes are nonetheless important, macro-averaging may be a means of highlighting their performance. On the other hand, the assumption that all classes are equally important is often untrue, such that macro-averaging will over-emphasize the typically low performance on an infrequent class.

"weighted" accounts for class imbalance by computing the average of binary metrics in which each class's score is weighted by its presence in the true data sample.

"micro" gives each sample-class pair an equal contribution to the overall metric (except as a result of sample-weight). Rather than summing the metric per class, this sums the dividends and divisors that make up the per-class metrics to calculate an overall quotient. Micro-averaging may be preferred in multilabel settings, including multiclass classification where a majority class is to be ignored.

"samples" applies only to multilabel problems. It does not calculate a per-class measure, instead calculating the metric over the true and predicted classes for each sample in the evaluation data, and returning their (sample_weight-weighted) average.

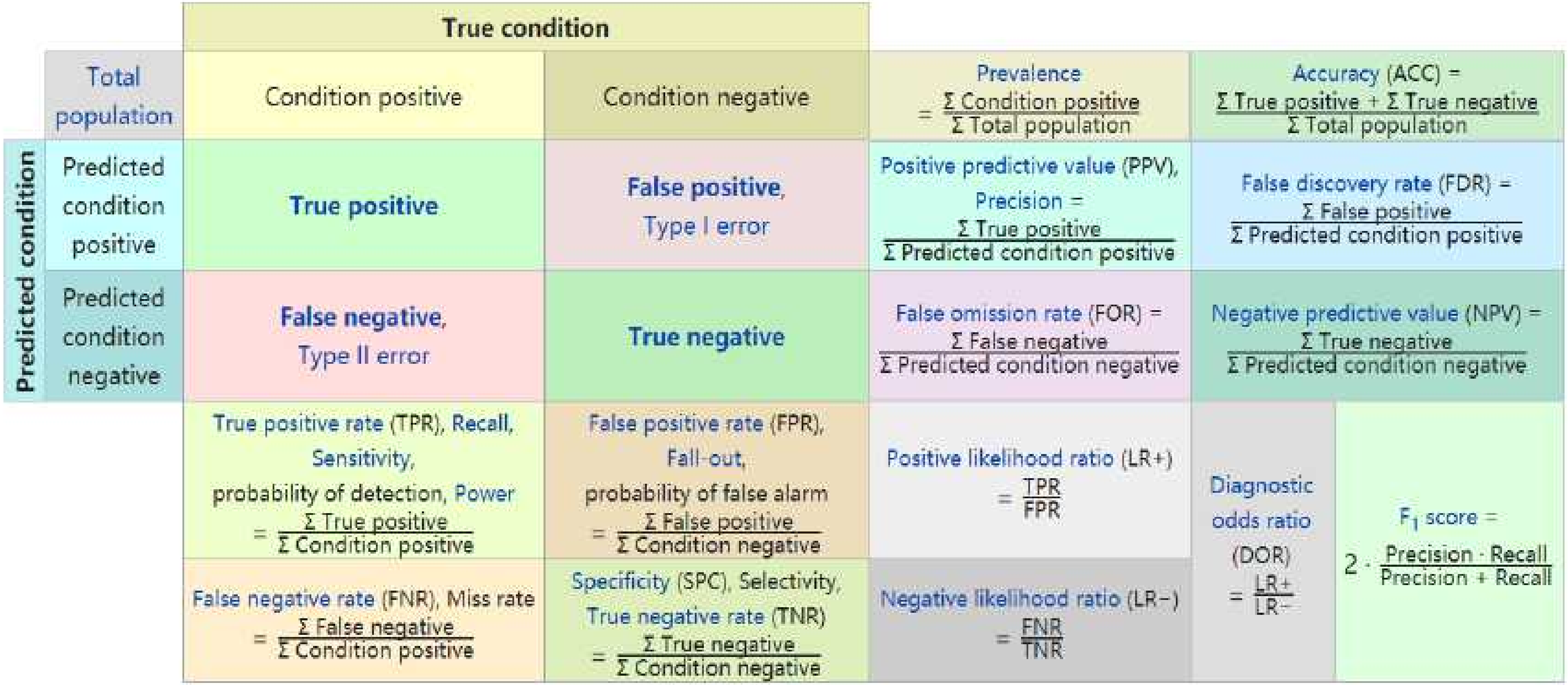
Selecting average=None will return an array with the score for each class.

Classification metrics：Confusion matrix

In the field of machine learning and specifically the problem of statistical classification, a **confusion matrix**, also known as an error matrix,is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a **matching matrix**). Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).

混淆矩阵也称误差矩阵，是表示精度评价的一种标准格式，用n行n列的矩阵形式来表示。具体评价指标有总体精度、制图精度、用户精度等，这些精度指标从不同的侧面反映了图像分类的精度。在人工智能中，混淆矩阵（confusion matrix）是可视化工具，特别用于监督学习，在无监督学习一般叫做匹配矩阵。在图像精度评价中，主要用于比较分类结果和实际测得值，可以把分类结果的精度显示在一个混淆矩阵里面。混淆矩阵是通过将每个实测像元的位置和分类与分类图像中的相应位置和分类相比较计算的。

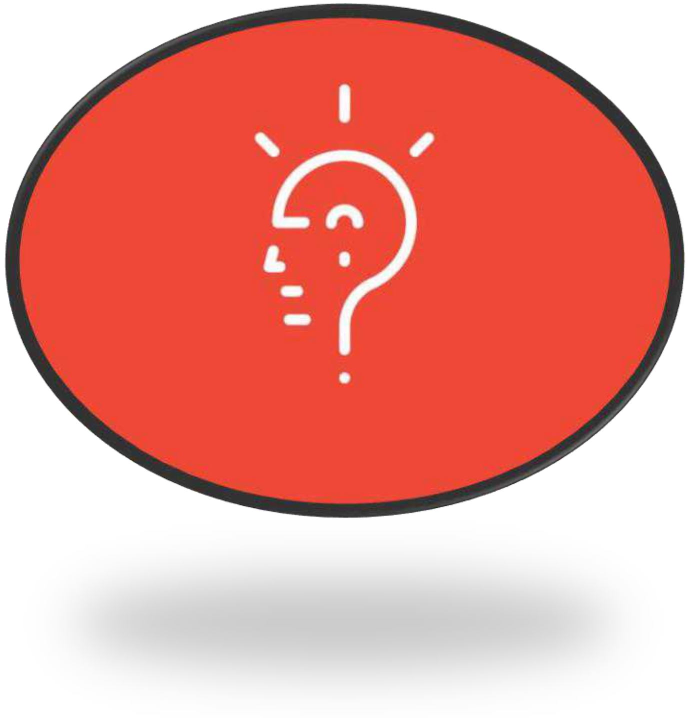
Classification metrics： Confusion matrix



		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR-}}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

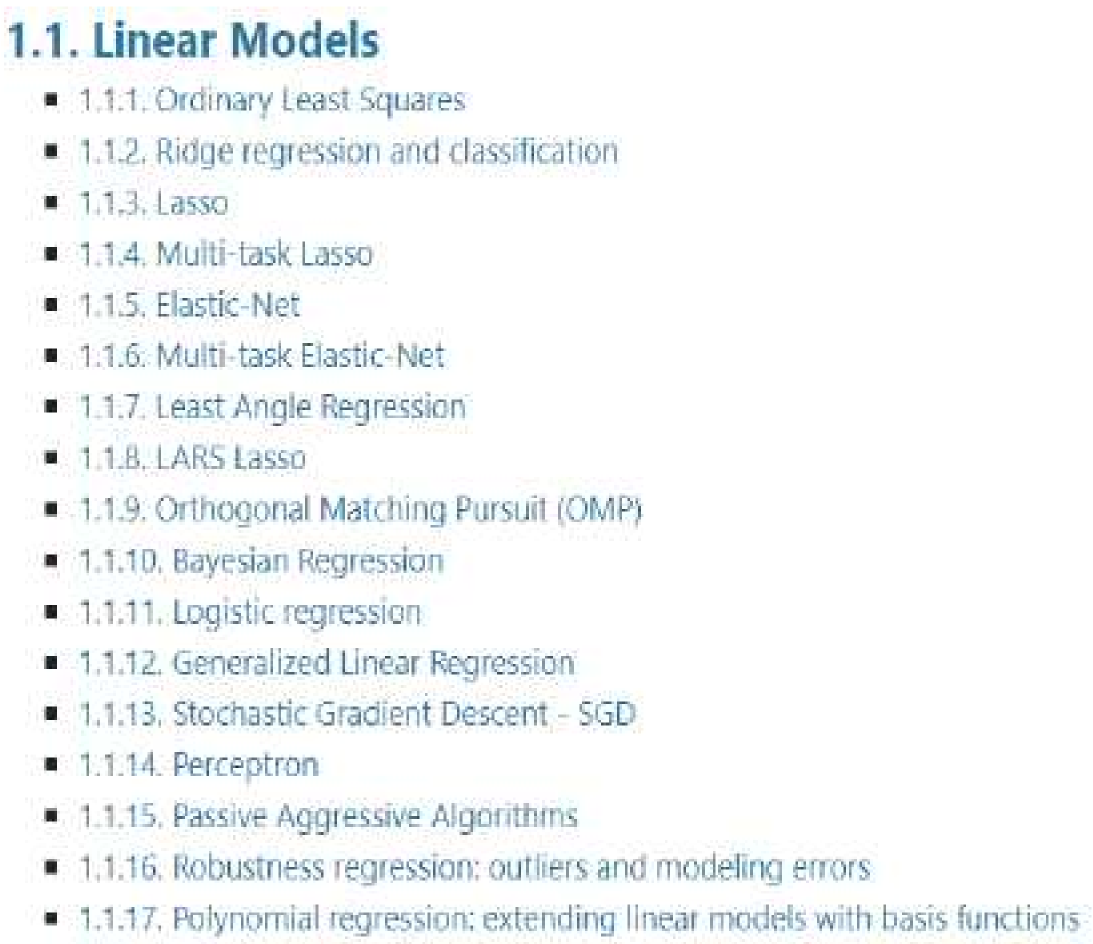
Classification metrics： Confusion matrix

Any Question?



Supervised learning

1. **Linear Models**
2. Linear and Quadratic Discriminant Analysis
3. Kernel ridge regression
4. **Support Vector Machines**
5. Stochastic Gradient Descent
6. **Nearest Neighbors**
7. Gaussian Processes
8. Cross decomposition
9. **Naive Bayes**
10. **Decision Trees**
11. **Ensemble methods**
12. Multiclass and multilabel algorithms
13. **Feature selection**
14. Semi-Supervised
15. Isotonic regression
16. Probability calibration
17. **Neural network models (supervised)**

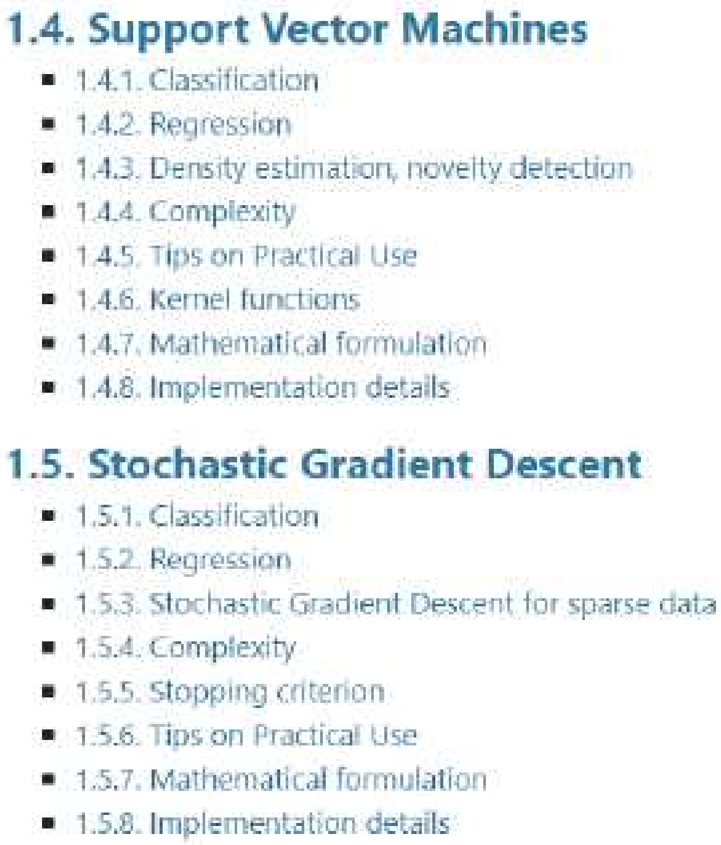


1.1. Linear Models

- 1.1.1. Ordinary Least Squares
- 1.1.2. Ridge regression and classification
- 1.1.3. Lasso
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic-Net
- 1.1.6. Multi-task Elastic-Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
- 1.1.11. Logistic regression
- 1.1.12. Generalized Linear Regression
- 1.1.13. Stochastic Gradient Descent - SGD
- 1.1.14. Perceptron
- 1.1.15. Passive Aggressive Algorithms
- 1.1.16. Robustness regression: outliers and modeling errors
- 1.1.17. Polynomial regression: extending linear models with basis functions

https://scikit-learn.org/stable/supervised_learning.html#

Supervised learning

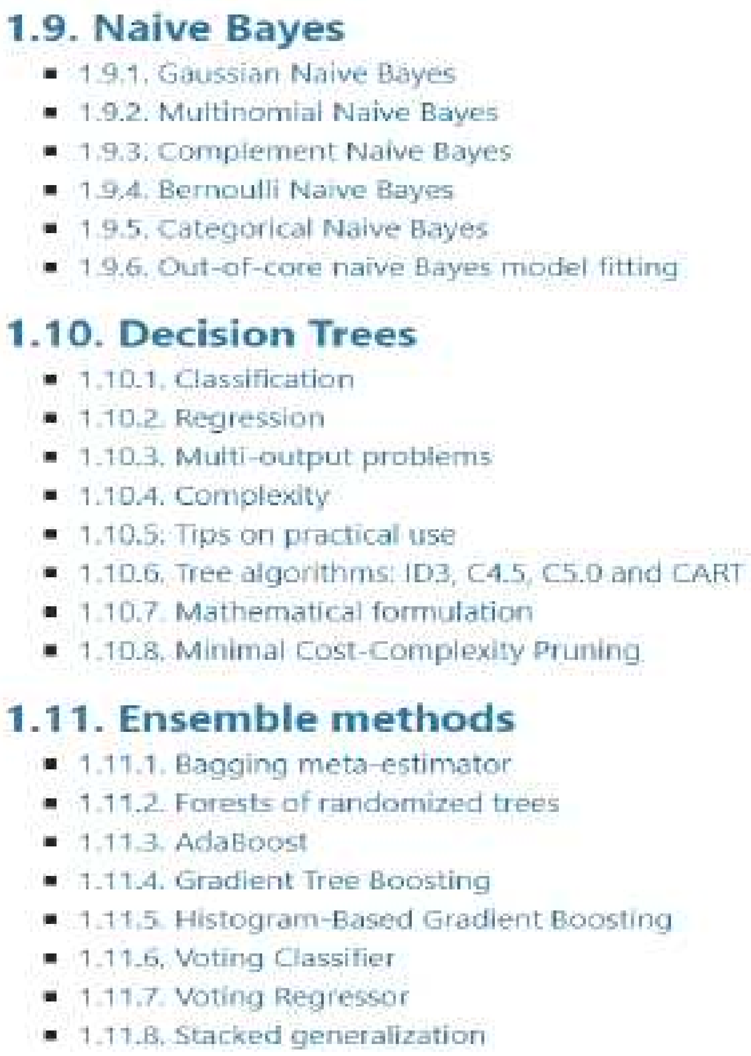


1.4. Support Vector Machines

- 1.4.1. Classification
- 1.4.2. Regression
- 1.4.3. Density estimation, novelty detection
- 1.4.4. Complexity
- 1.4.5. Tips on Practical Use
- 1.4.6. Kernel functions
- 1.4.7. Mathematical formulation
- 1.4.8. Implementation details

1.5. Stochastic Gradient Descent

- 1.5.1. Classification
- 1.5.2. Regression
- 1.5.3. Stochastic Gradient Descent for sparse data
- 1.5.4. Complexity
- 1.5.5. Stopping criterion
- 1.5.6. Tips on Practical Use
- 1.5.7. Mathematical formulation
- 1.5.8. Implementation details



1.9. Naive Bayes

- 1.9.1. Gaussian Naive Bayes
- 1.9.2. Multinomial Naive Bayes
- 1.9.3. Complement Naive Bayes
- 1.9.4. Bernoulli Naive Bayes
- 1.9.5. Categorical Naive Bayes
- 1.9.6. Out-of-core naive Bayes model fitting

1.10. Decision Trees

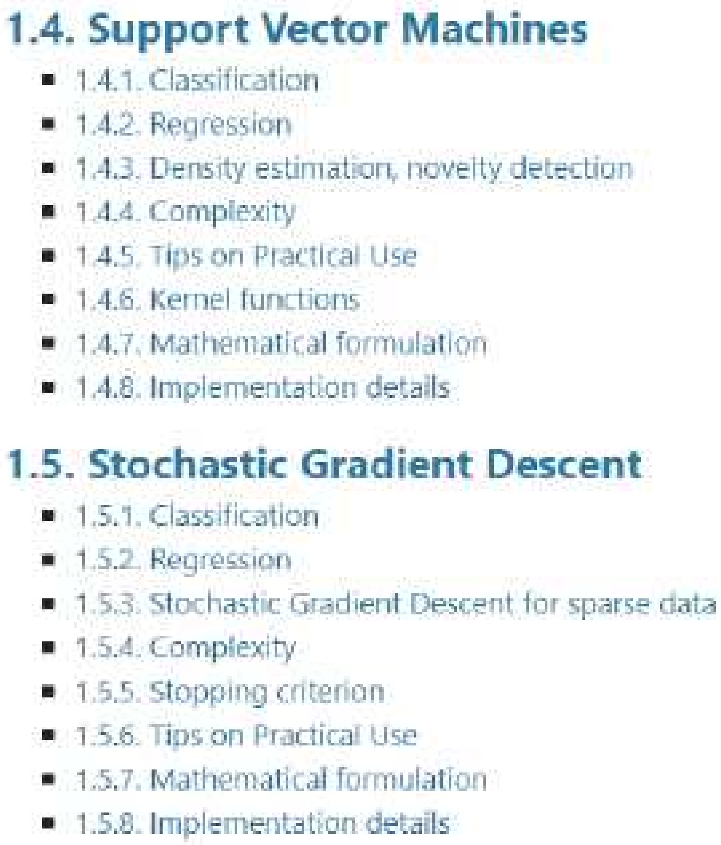
- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity
- 1.10.5. Tips on practical use
- 1.10.6. Tree algorithms: ID3, C4.5, C5.0 and CART
- 1.10.7. Mathematical formulation
- 1.10.8. Minimal Cost-Complexity Pruning

1.11. Ensemble methods

- 1.11.1. Bagging meta-estimator
- 1.11.2. Forests of randomized trees
- 1.11.3. AdaBoost
- 1.11.4. Gradient Tree Boosting
- 1.11.5. Histogram-Based Gradient Boosting
- 1.11.6. Voting Classifier
- 1.11.7. Voting Regressor
- 1.11.8. Stacked generalization

https://scikit-learn.org/stable/supervised_learning.html#

Supervised Learning

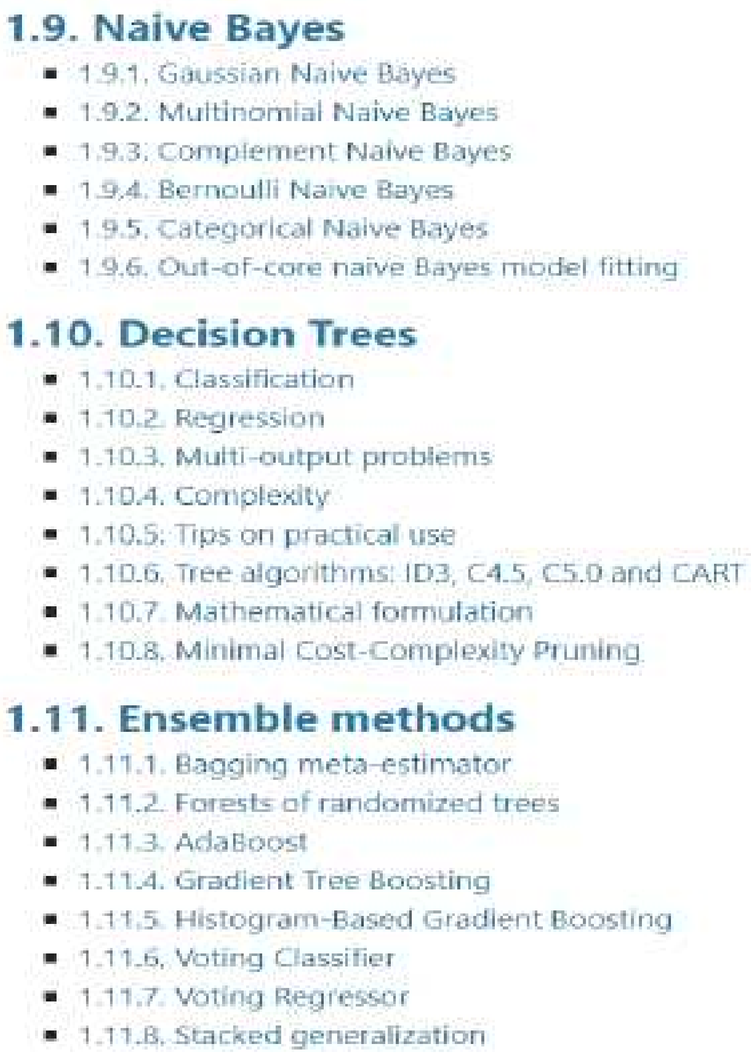


1.4. Support Vector Machines

- 1.4.1. Classification
- 1.4.2. Regression
- 1.4.3. Density estimation, novelty detection
- 1.4.4. Complexity
- 1.4.5. Tips on Practical Use
- 1.4.6. Kernel functions
- 1.4.7. Mathematical formulation
- 1.4.8. Implementation details

1.5. Stochastic Gradient Descent

- 1.5.1. Classification
- 1.5.2. Regression
- 1.5.3. Stochastic Gradient Descent for sparse data
- 1.5.4. Complexity
- 1.5.5. Stopping criterion
- 1.5.6. Tips on Practical Use
- 1.5.7. Mathematical formulation
- 1.5.8. Implementation details



1.9. Naive Bayes

- 1.9.1. Gaussian Naive Bayes
- 1.9.2. Multinomial Naive Bayes
- 1.9.3. Complement Naive Bayes
- 1.9.4. Bernoulli Naive Bayes
- 1.9.5. Categorical Naive Bayes
- 1.9.6. Out-of-core naive Bayes model fitting

1.10. Decision Trees

- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity
- 1.10.5. Tips on practical use
- 1.10.6. Tree algorithms: ID3, C4.5, C5.0 and CART
- 1.10.7. Mathematical formulation
- 1.10.8. Minimal Cost-Complexity Pruning

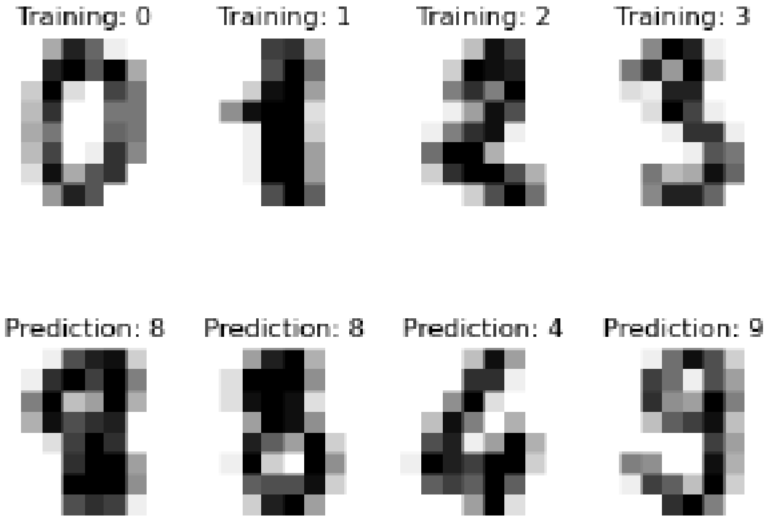
1.11. Ensemble methods

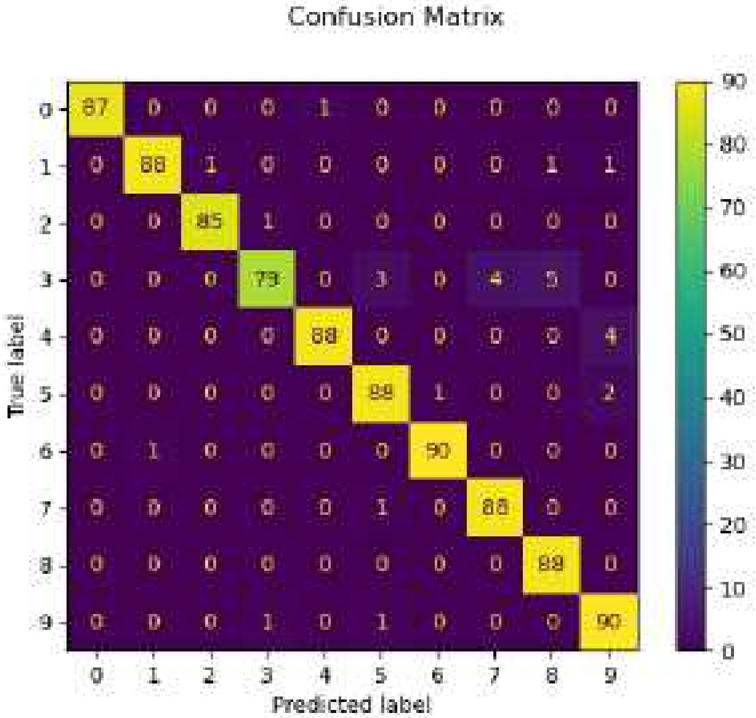
- 1.11.1. Bagging meta-estimator
- 1.11.2. Forests of randomized trees
- 1.11.3. AdaBoost
- 1.11.4. Gradient Tree Boosting
- 1.11.5. Histogram-Based Gradient Boosting
- 1.11.6. Voting Classifier
- 1.11.7. Voting Regressor
- 1.11.8. Stacked generalization

https://scikit-learn.org/stable/supervised_learning.html#

Exercise: Plot the confusion matrix

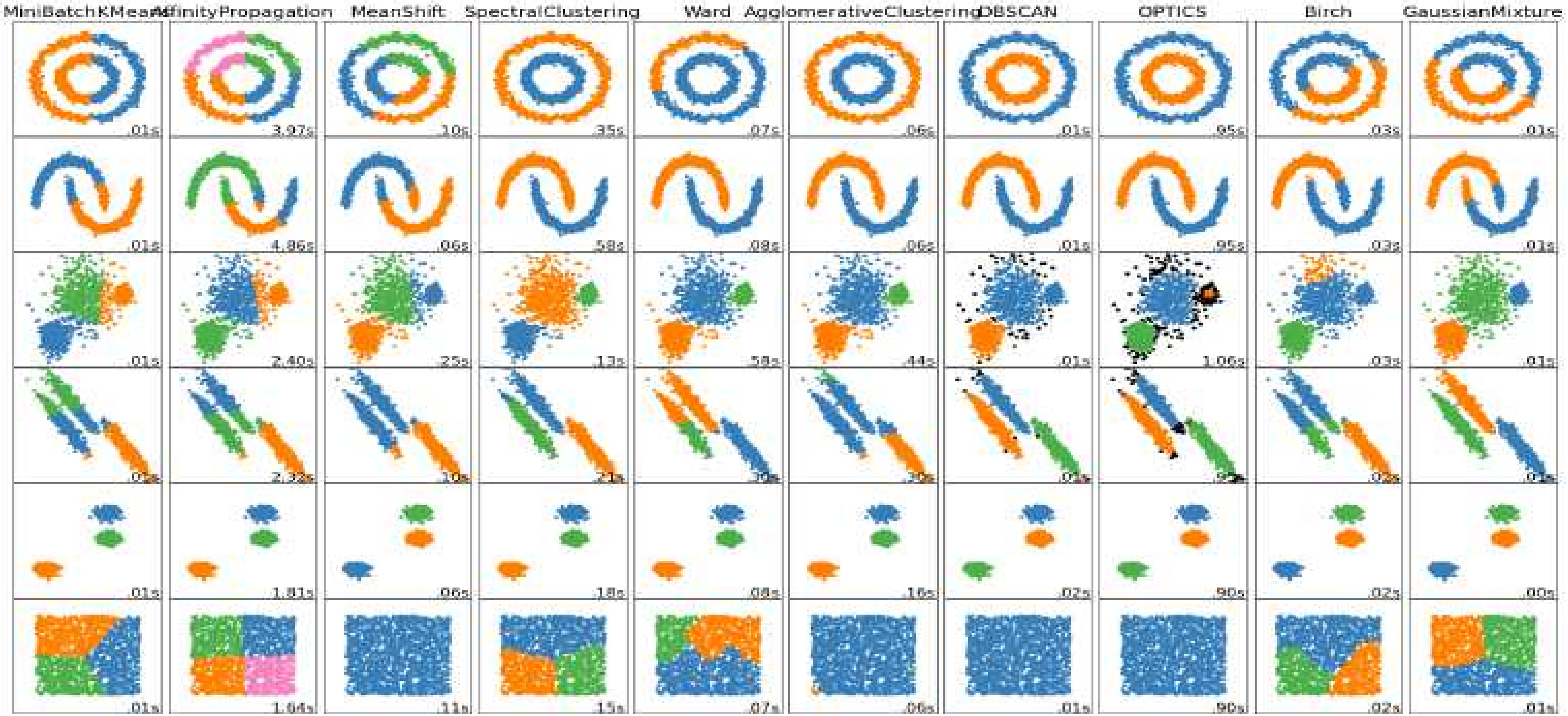
Recognizing hand-written digits





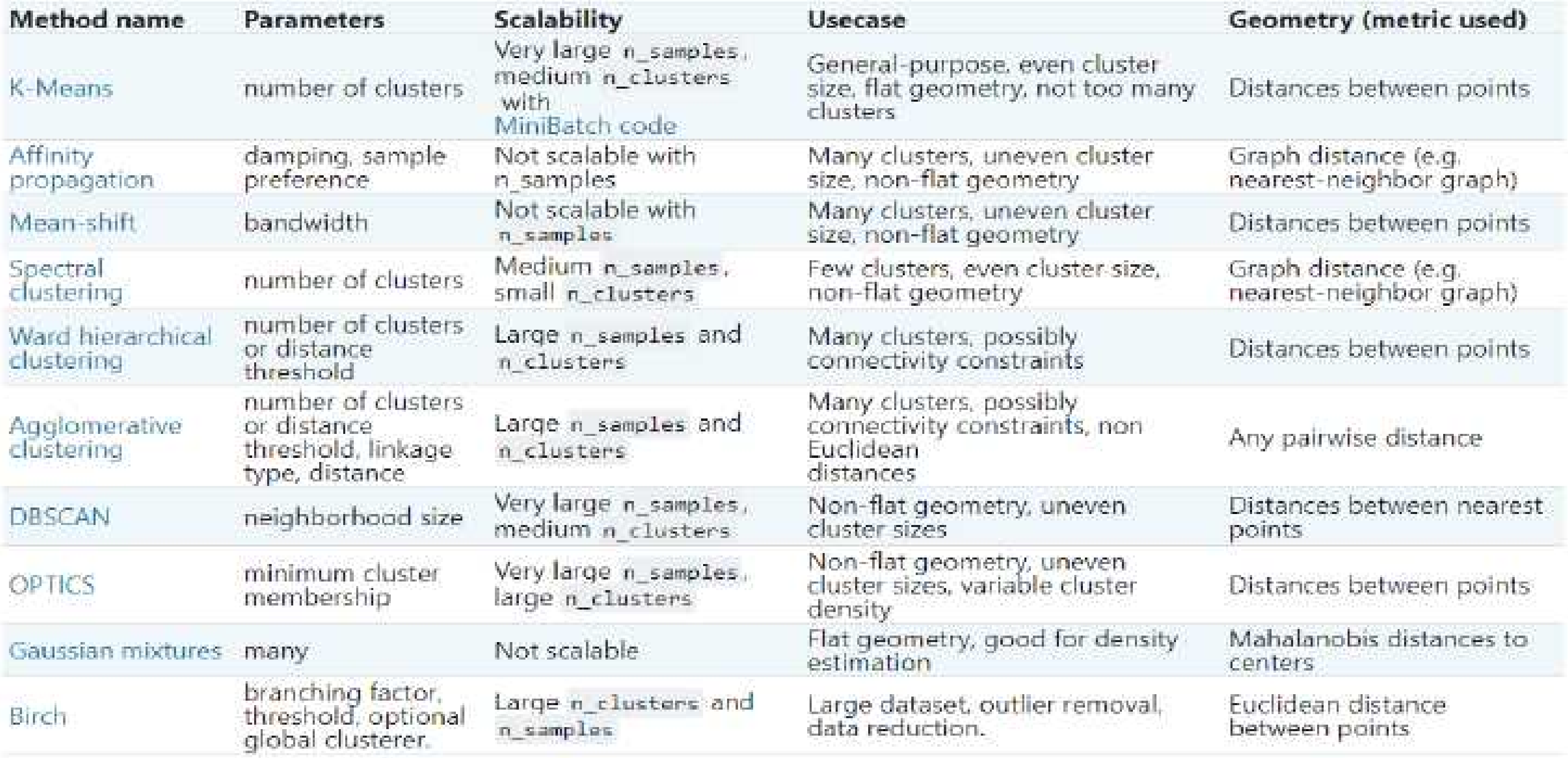
	0	1	2	3	4	5	6	7	8	9
0	87	0	0	0	1	0	0	0	0	0
1	0	88	1	0	0	0	0	0	1	1
2	0	0	85	1	0	0	0	0	0	0
3	0	0	0	79	0	3	0	4	5	0
4	0	0	0	0	88	0	0	0	0	4
5	0	0	0	0	0	88	1	0	0	2
6	0	1	0	0	0	0	90	0	0	0
7	0	0	0	0	0	1	0	88	0	0
8	0	0	0	0	0	0	0	0	88	0
9	0	0	0	1	0	1	0	0	0	90

Clustering



https://scikit-learn.org/stable/modules/clustering.html#clustering

Clustering



Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large n_samples, medium n_clusters with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium n_samples, small n_clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes	Distances between nearest points
OPTICS	minimum cluster membership	Very large n_samples, large n_clusters	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large n_clusters and n_samples	Large dataset, outlier removal, data reduction.	Euclidean distance between points

https://scikit-learn.org/stable/modules/clustering.html#clustering

Clustering

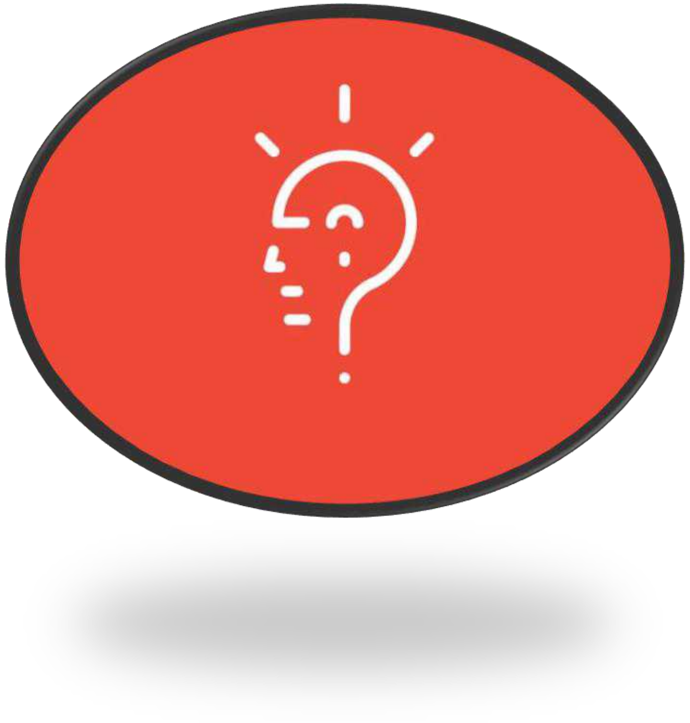
Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including pattern recognition, image analysis, information retrieval, bioinformatics, data compression, computer graphics and machine learning.

聚类分析指将物理或抽象对象的集合分组为由类似的对象组成的多个类的分析过程。它是一种重要的人类行为。聚类分析的目标就是在相似的基础上收集数据来分类。聚类源于很多领域，包括数学，计算机科学，统计学，生物学和经济学。在不同的应用领域，很多聚类技术都得到了发展，这些技术方法被用作描述数据，衡量不同数据源间的相似性，以及把数据源分类到不同的簇中。

聚类与分类的不同在于，聚类所要求划分的类是未知的。
聚类是将数据分类到不同的类或者簇这样的一个过程，所以同一个簇中的对象有很大的相似性，而不同簇间的对象有很大的相异性。

https://scikit-learn.org/stable/modules/clustering.html#clustering

Any Question?



Preprocessing data

The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that **is more suitable for the downstream estimators.**

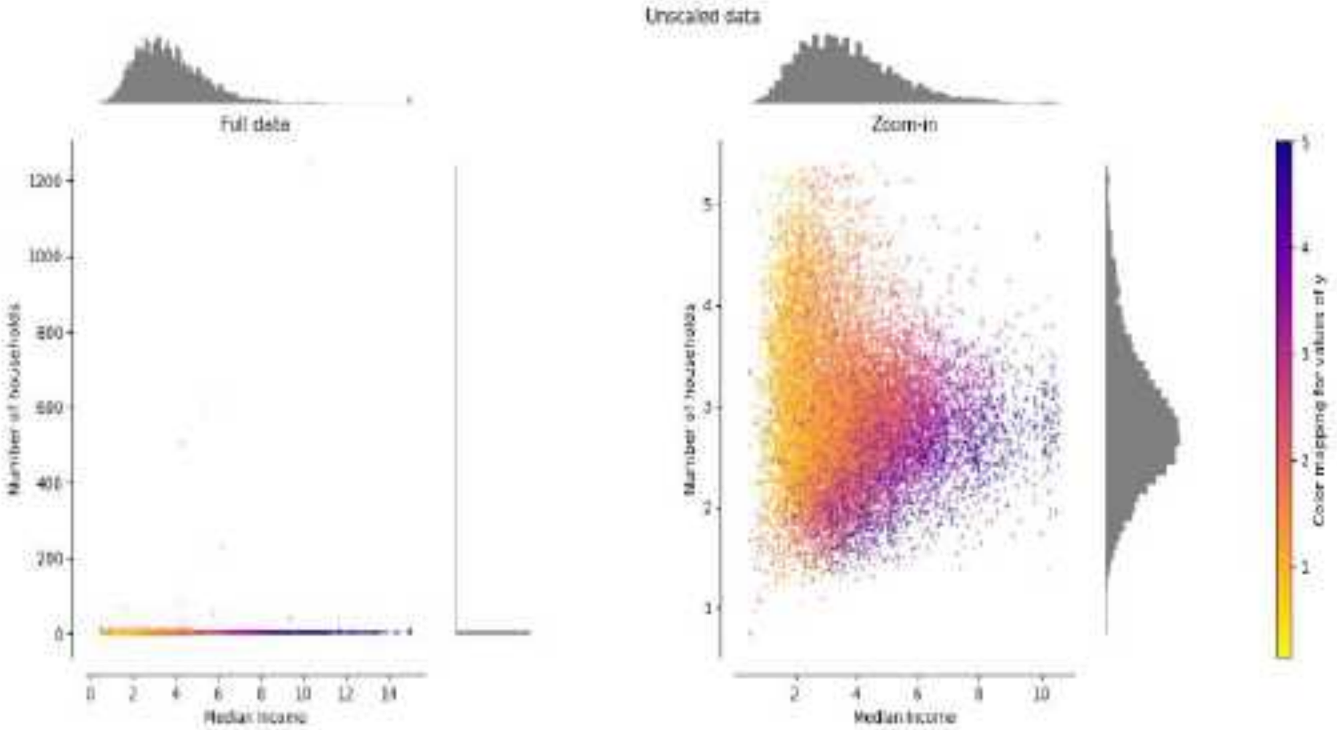
In general, learning algorithms **benefit from standardization of the data set.** If some outliers are present in the set, robust scalers or transformers are more appropriate. The behaviors of the different scalers, transformers, and normalizers on a dataset containing marginal outliers is highlighted in Compare the effect of different scalers on data with outliers.

1. Standardization, or mean removal and variance scaling
2. Non-linear transformation
3. Normalization
4. Encoding categorical features
5. Discretization
6. Imputation of missing values
7. Generating polynomial features
8. Custom transformers

https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing

Compare the effect of different scalers on data with outliers

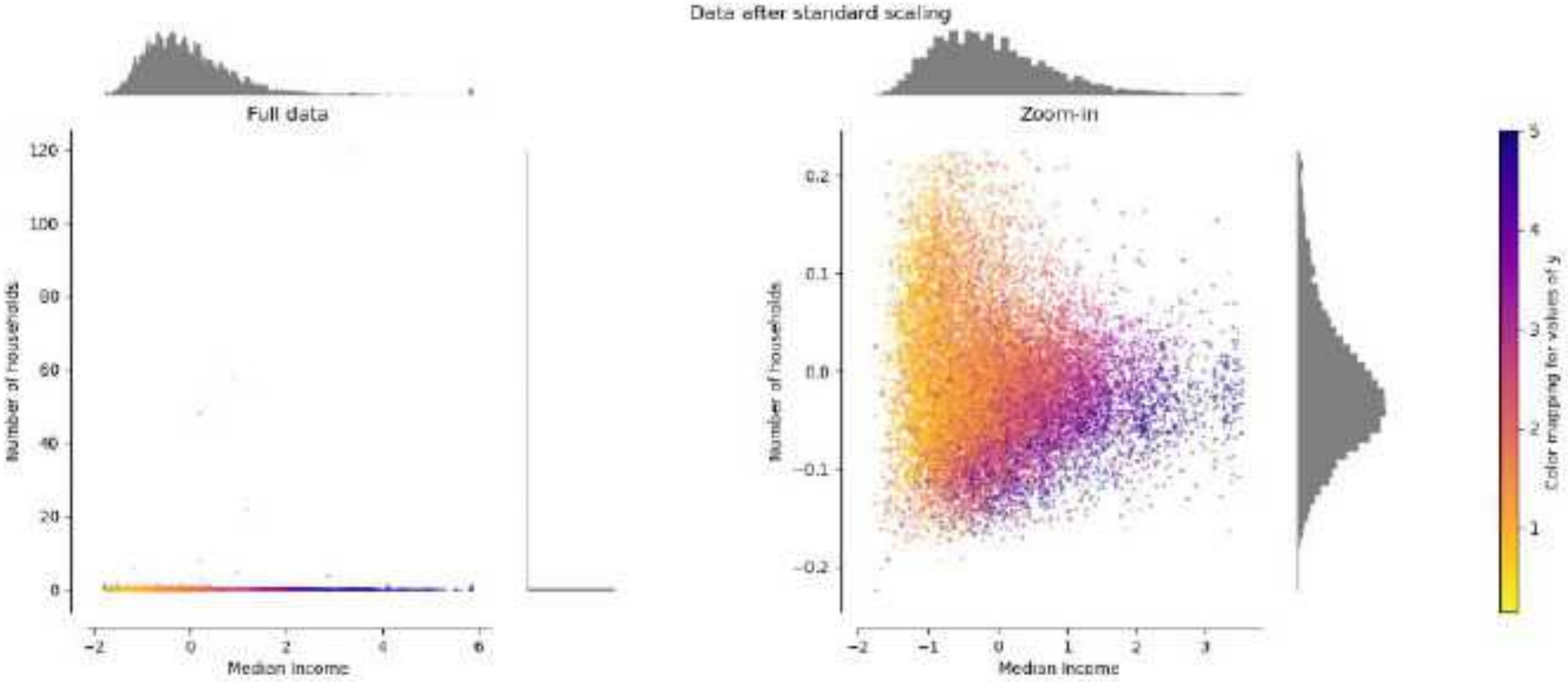
1. **StandardScaler**
2. **MinMaxScaler**
3. MaxAbsScaler
4. RobustScaler
5. PowerTransformer
6. QuantileTransformer (Gaussian output)
7. QuantileTransformer (uniform output)
8. Normalizer



Original data (California housing dataset)

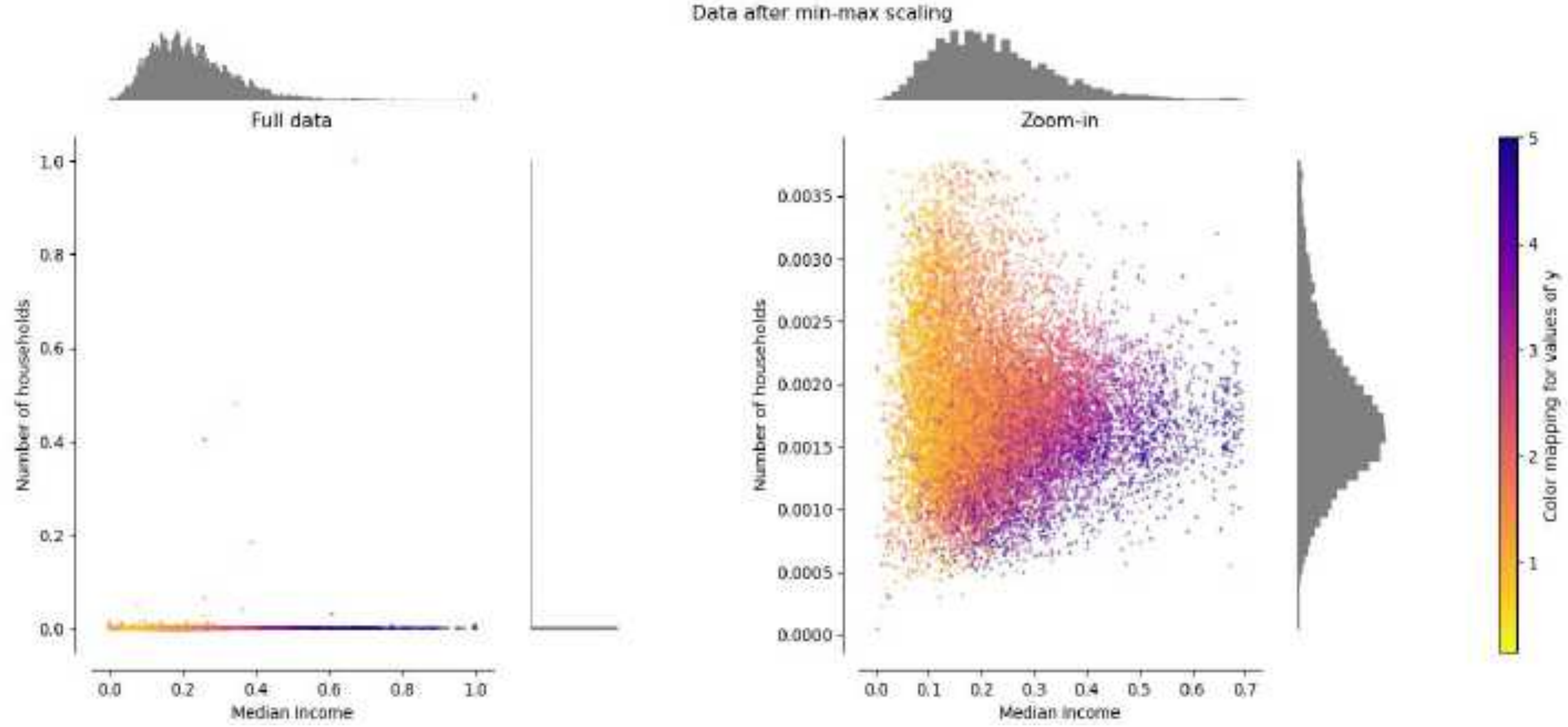
https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py

Compare the effect of different scalers on data with outliers



StandardScaler removes the mean and scales the data to unit variance. However, the outliers have an influence when computing the empirical mean and standard deviation which shrink the range of the feature values as shown in the left figure above

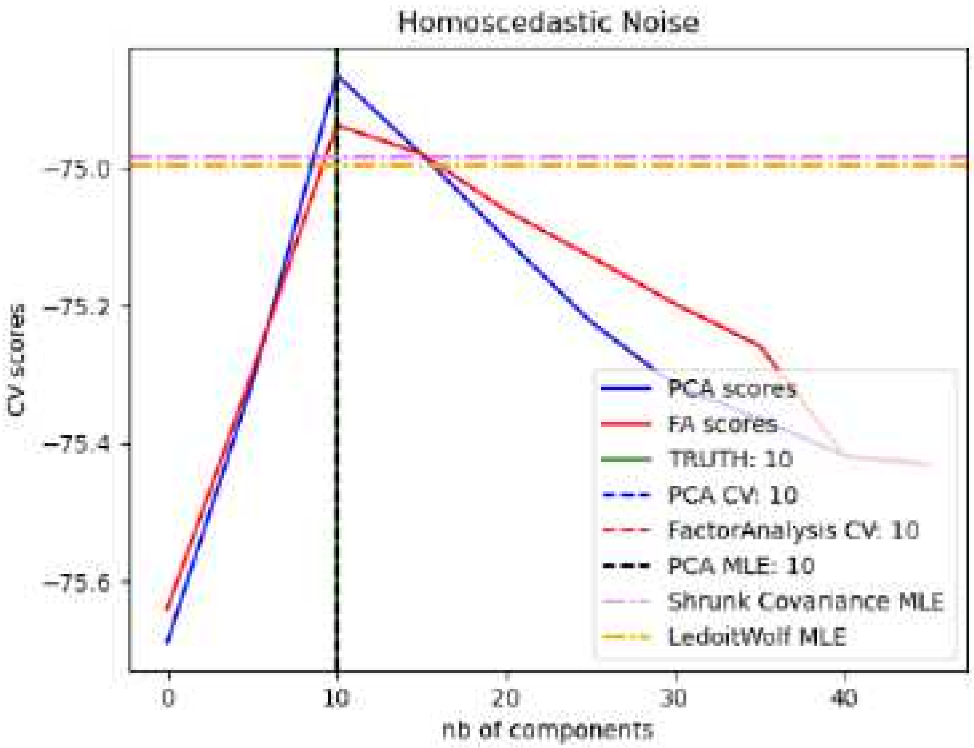
Compare the effect of different scalers on data with outliers



MinMaxScaler rescales the data set such that all feature values are in the range [0, 1] as shown in the right panel below. However, this scaling compress all inliers in the narrow range [0, 0.005] for the transformed number of households.

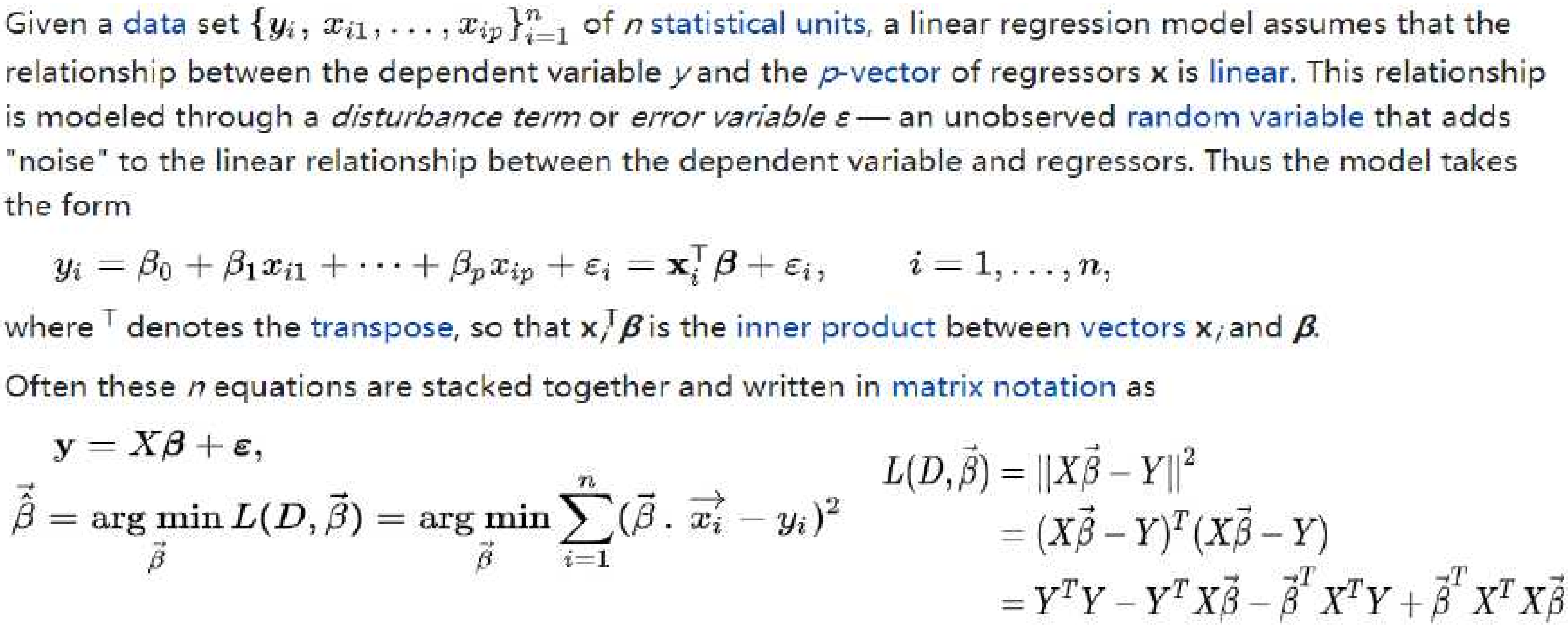
Compare the effect of different scalers on data with outliers

1. Principal component analysis (PCA)
2. Truncated singular value decomposition and latent semantic analysis
3. Dictionary Learning
4. Factor Analysis
5. Independent component analysis (ICA)
6. Non-negative matrix factorization (NMF or NNMF)
7. Latent Dirichlet Allocation (LDA)



Exercise: training linear models

https://github.com/ageron/handson-ml2/blob/master/04_training_linear_models.ipynb



Given a data set $\{y_i,\, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ of *n* statistical units, a linear regression model assumes that the relationship between the dependent variable *y* and the *p*-vector of regressors **x** is linear. This relationship is modeled through a *disturbance term* or *error variable ε* — an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathsf{T}} \boldsymbol\beta + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where $^{\mathsf{T}}$ denotes the transpose, so that $\mathbf{x}_i^{\mathsf{T}}\boldsymbol\beta$ is the inner product between vectors $\mathbf{x}_i$ and $\boldsymbol\beta$.

Often these *n* equations are stacked together and written in matrix notation as

$$\mathbf{y} = X\boldsymbol\beta + \boldsymbol\varepsilon,$$

$$\vec{\hat{\beta}} = \arg\min_{\vec\beta} L(D, \vec\beta) = \arg\min_{\vec\beta} \sum_{i=1}^{n} (\vec\beta \,.\, \vec{x_i} - y_i)^2$$

$$\begin{aligned} L(D,\vec\beta) &= \|X\vec\beta - Y\|^2 \\ &= (X\vec\beta - Y)^T (X\vec\beta - Y) \\ &= Y^TY - Y^TX\vec\beta - \vec\beta^T X^TY + \vec\beta^T X^T X\vec\beta \end{aligned}$$

Exercise: MNIST Classification

Reference: https://github.com/ageron/handson-ml2/blob/master/03_classification.ipynb

Any Question?

PyTorch

目录

1 History and similar frameworks

2 Architecture of AI model

3 Common APIs

4 Some learning resources

1 History and similar frameworks

- 2002--Torch
- 2011--Torch7

Merits： flexible、dynamic、
user-friendly

Demerits： based on Lua

Caffe

- 2013--Caffe
- 2017--Caffe2

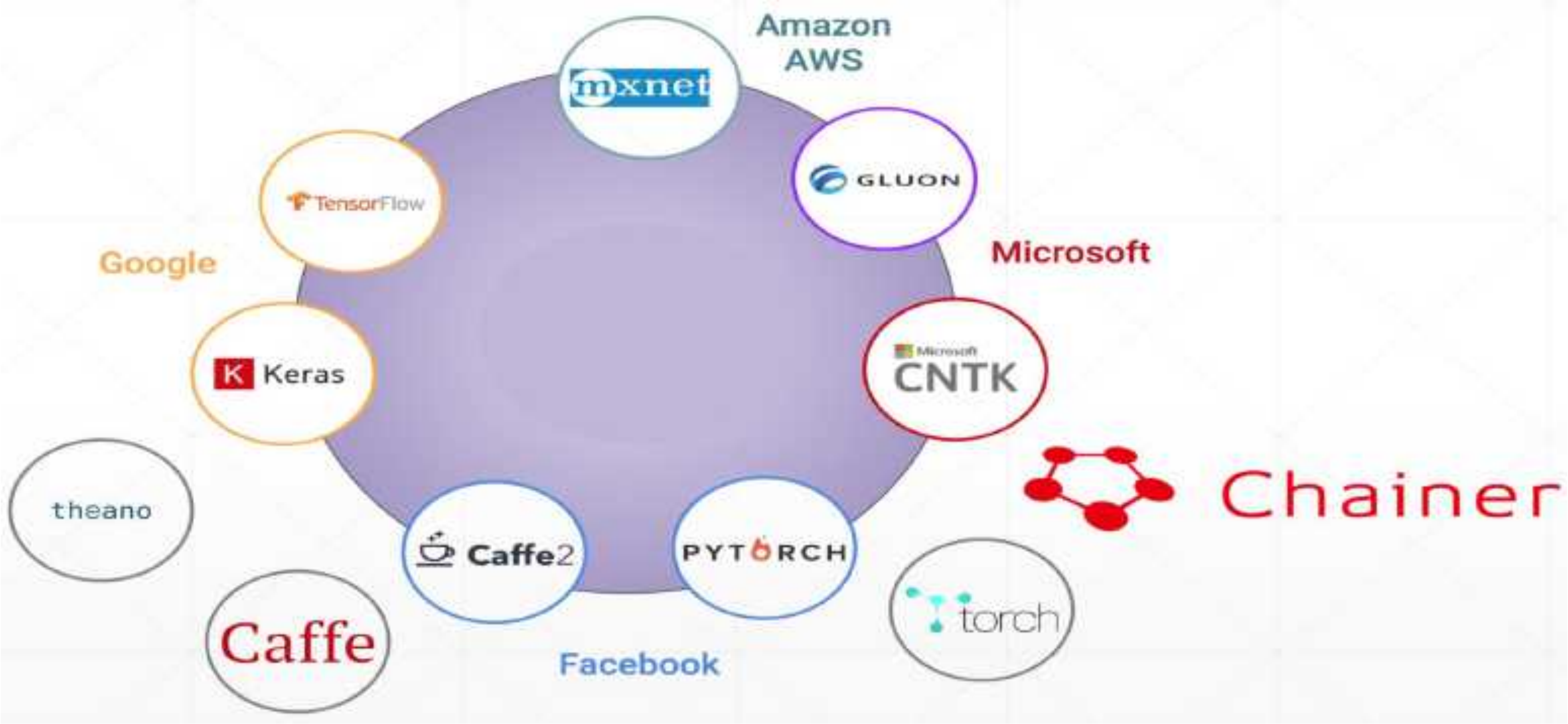
Merits： fast(based on C++)

Demerits： not flexible

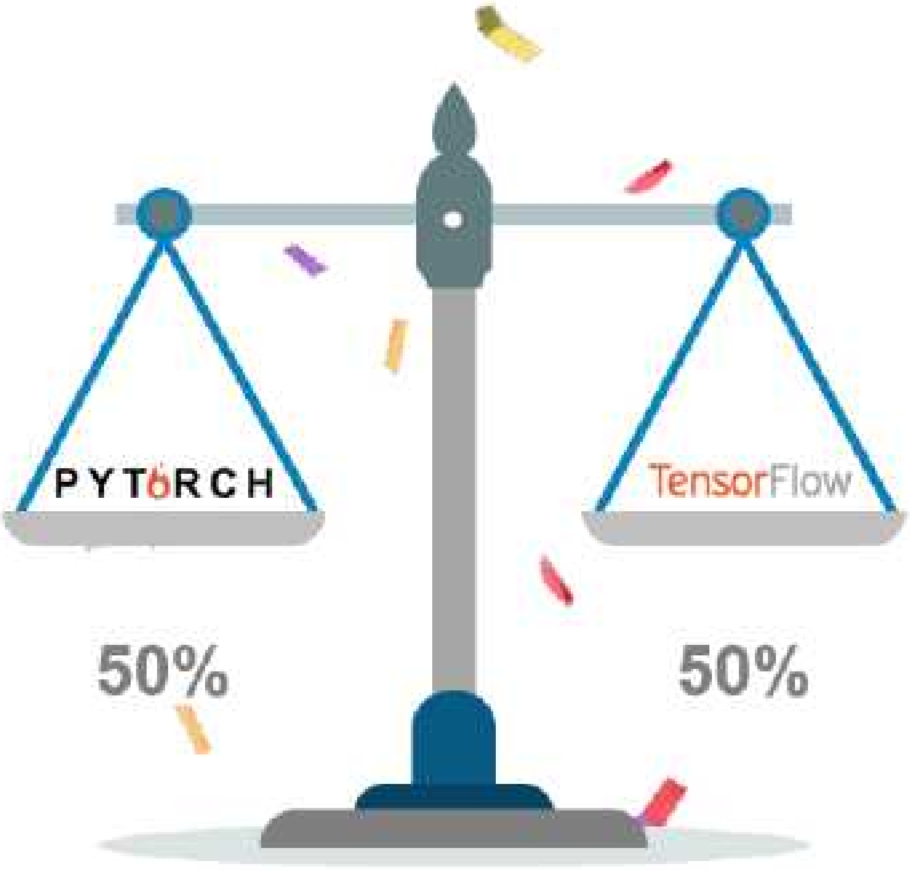
- 2017--PyTorch
- 2018--PyTorch v0.4

PyTorch

1 History and similar frameworks

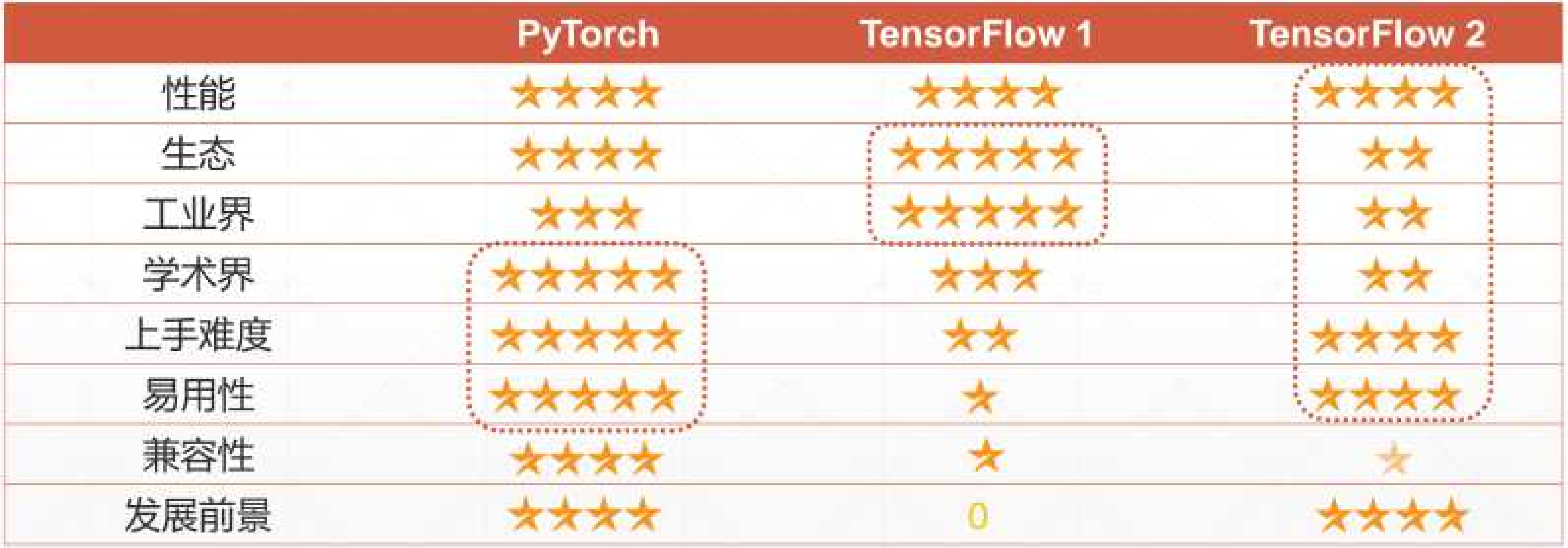


1 History and similar frameworks



1 History and similar frameworks

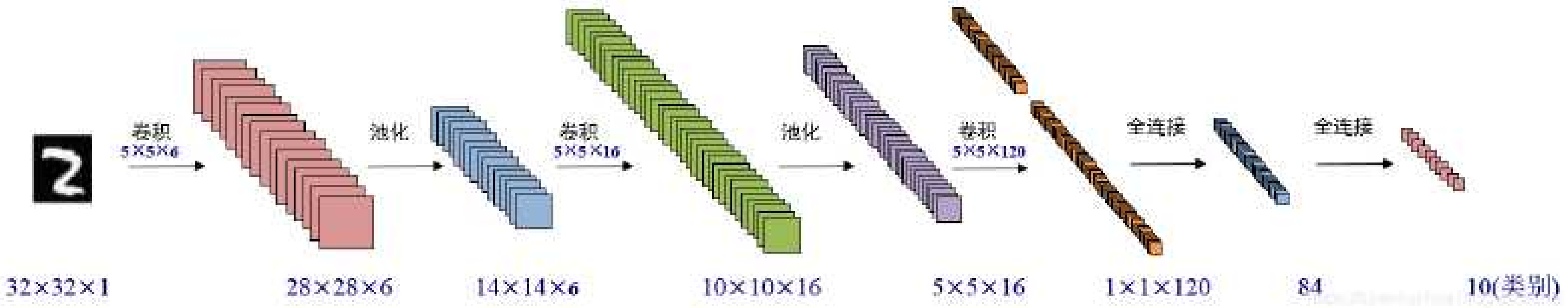
Evaluation



	PyTorch	TensorFlow 1	TensorFlow 2
性能	★★★★	★★★★	★★★★
生态	★★★★	★★★★★	★★
工业界	★★★	★★★★★	★★
学术界	★★★★★	★★★	★★
上手难度	★★★★★	★★	★★★★
易用性	★★★★★	★	★★★★
兼容性	★★★★	★	★
发展前景	★★★★	0	★★★★

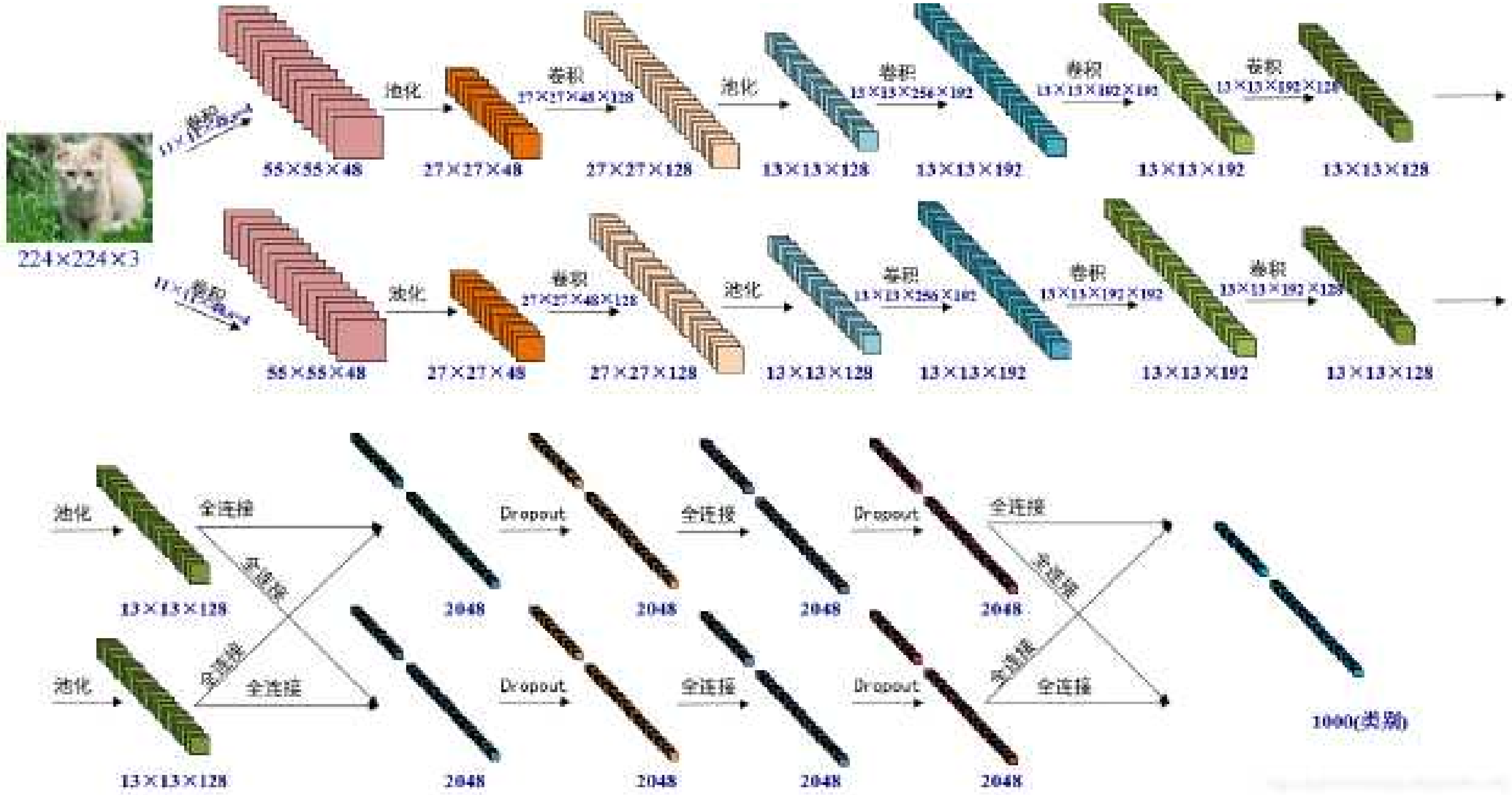
2 Archetecture of AI model

How to build a model?



2 Archetecture of AI model

How to build a model?



2 Archetecture of AI model

How to build a model?

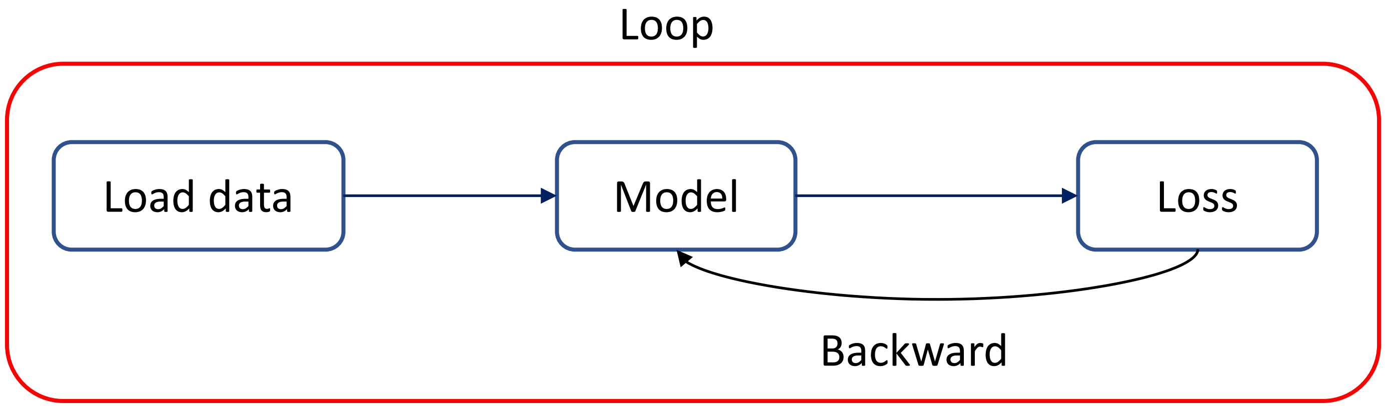
A model may have:

- Convolutional layers
- Fully connetected layers
- Batch normolization layers
- Pooling layers
- Other operations: concat/split/add ...

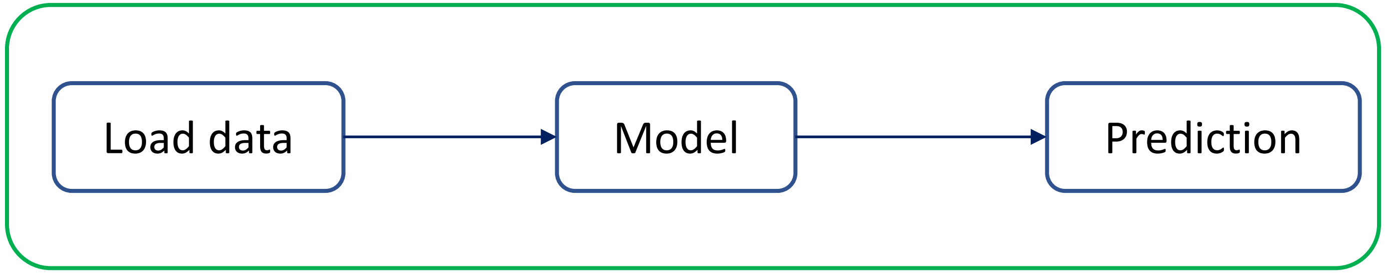
2 Archetecture of AI model

How to train a model?

Train:

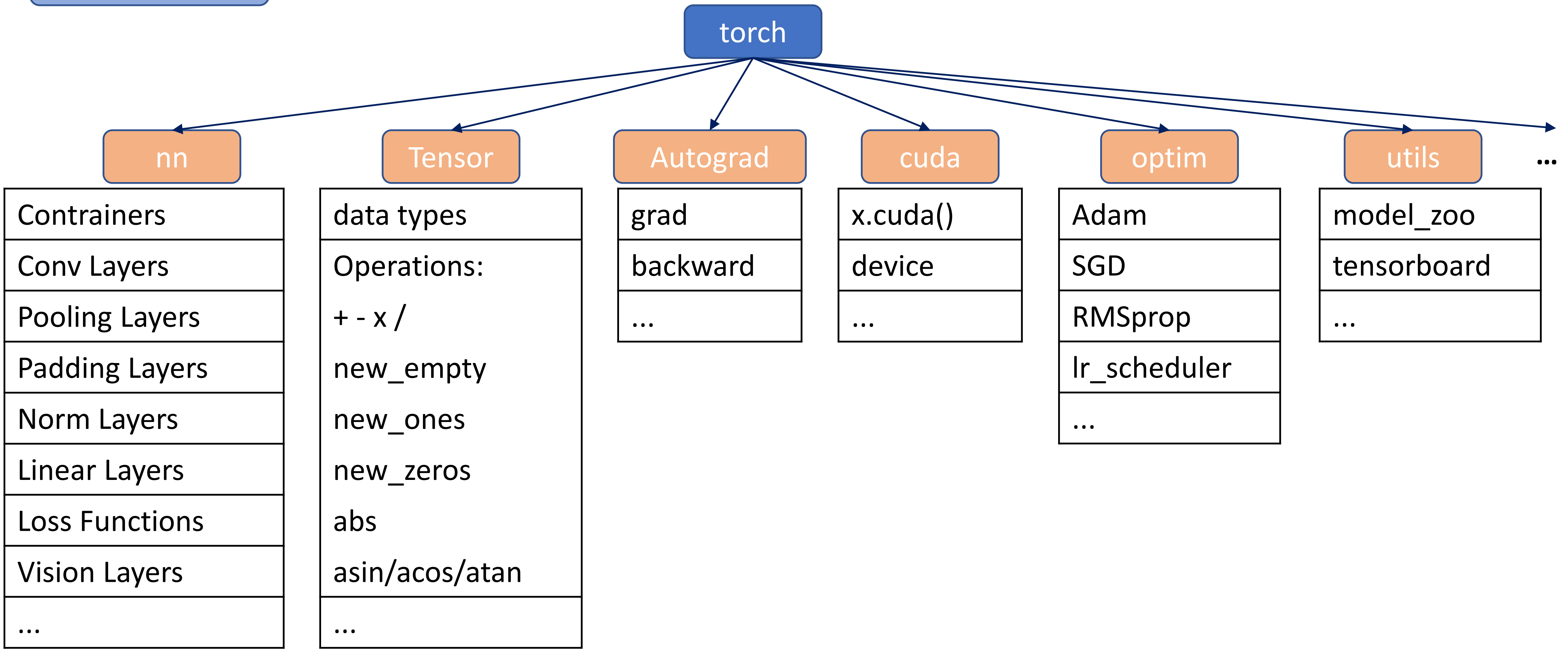


Predict:



3 Common APIs

import torch



torch

nn
- Contrainers
- Conv Layers
- Pooling Layers
- Padding Layers
- Norm Layers
- Linear Layers
- Loss Functions
- Vision Layers
- ...

Tensor
- data types
- Operations:
- + - x /
- new_empty
- new_ones
- new_zeros
- abs
- asin/acos/atan
- ...

Autograd
- grad
- backward
- ...

cuda
- x.cuda()
- device
- ...

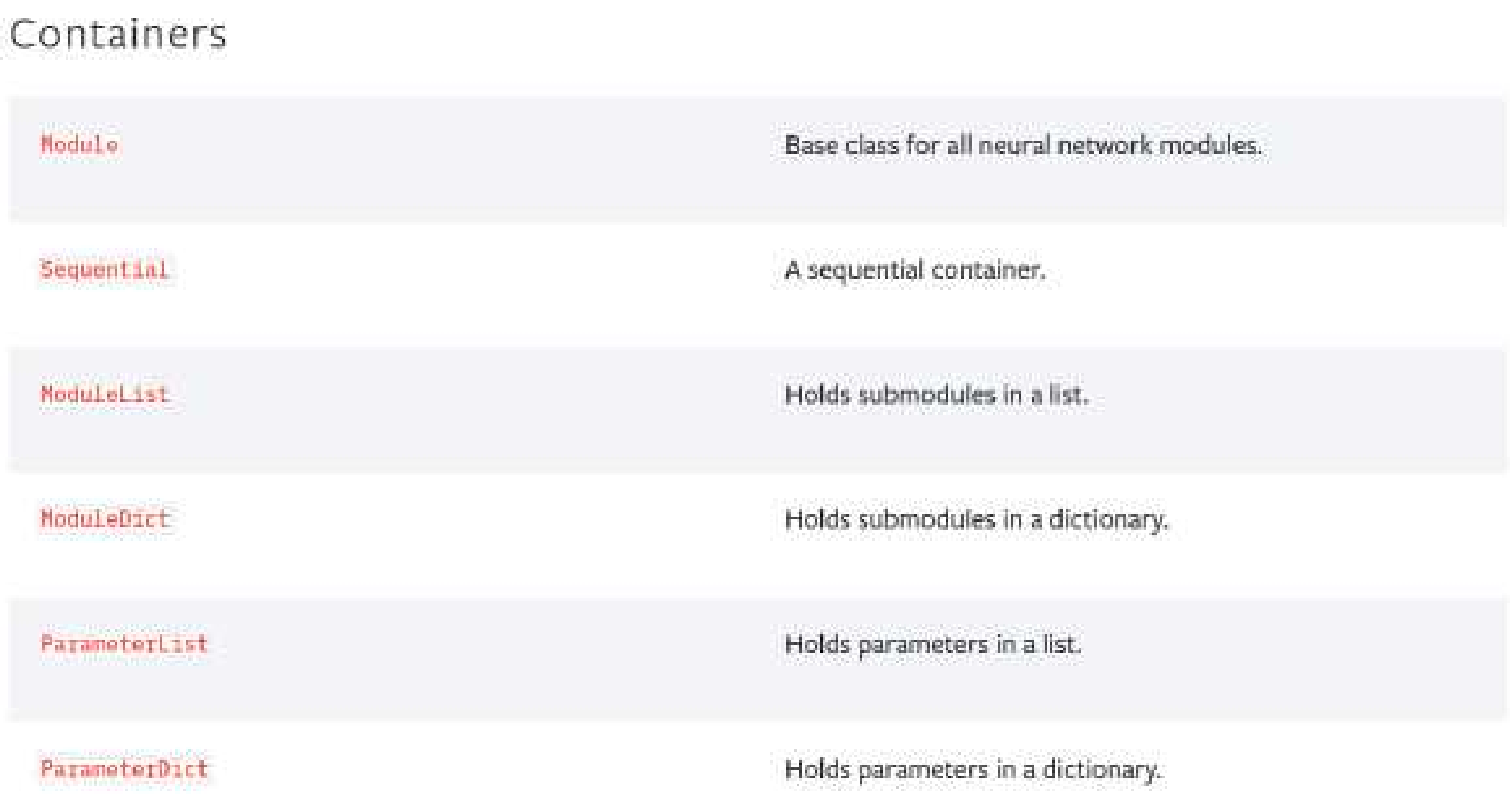
optim
- Adam
- SGD
- RMSprop
- lr_scheduler
- ...

utils
- model_zoo
- tensorboard
- ...

...

3 Common APIs

torch.nn

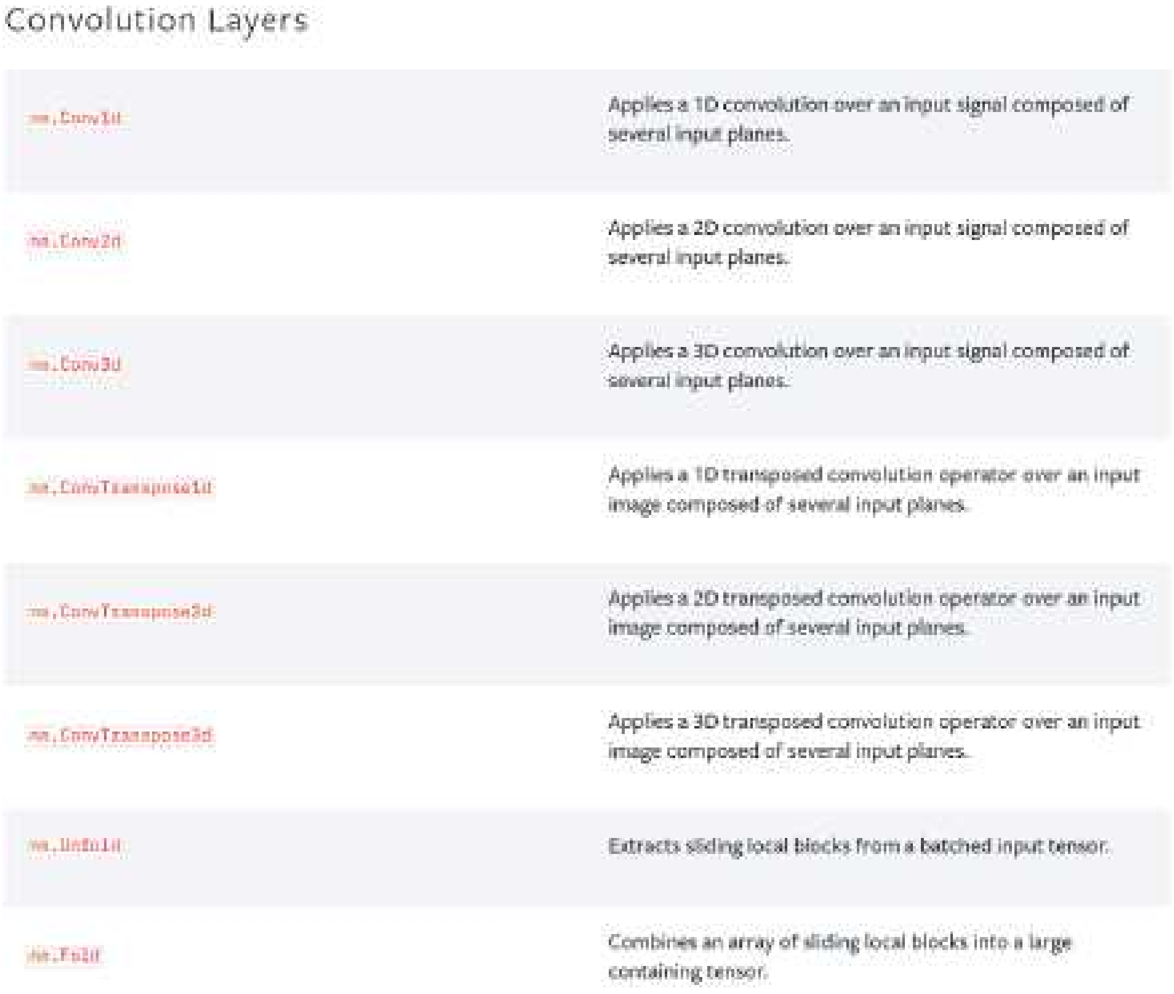


Containers

`Module`	Base class for all neural network modules.
`Sequential`	A sequential container.
`ModuleList`	Holds submodules in a list.
`ModuleDict`	Holds submodules in a dictionary.
`ParameterList`	Holds parameters in a list.
`ParameterDict`	Holds parameters in a dictionary.

3 Common APIs

torch.nn

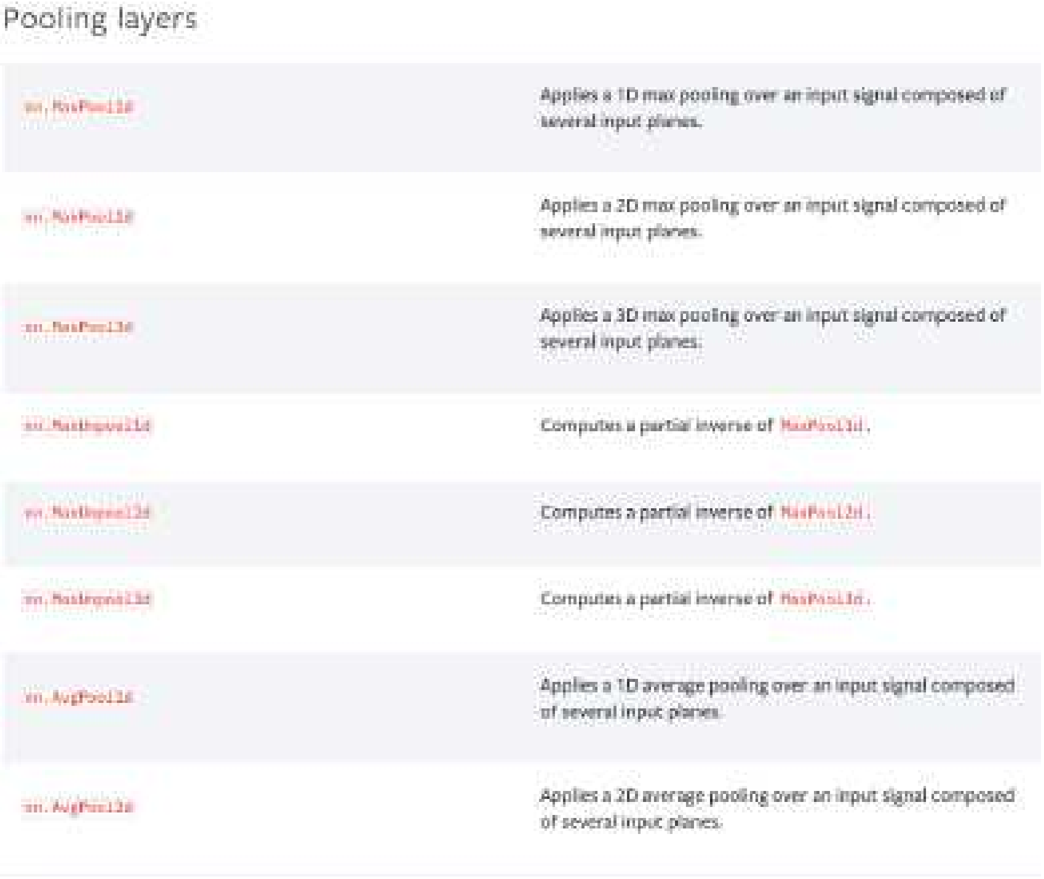


Convolution Layers

nn.Conv1d	Applies a 1D convolution over an input signal composed of several input planes.
nn.Conv2d	Applies a 2D convolution over an input signal composed of several input planes.
nn.Conv3d	Applies a 3D convolution over an input signal composed of several input planes.
nn.ConvTranspose1d	Applies a 1D transposed convolution operator over an input image composed of several input planes.
nn.ConvTranspose2d	Applies a 2D transposed convolution operator over an input image composed of several input planes.
nn.ConvTranspose3d	Applies a 3D transposed convolution operator over an input image composed of several input planes.
nn.Unfold	Extracts sliding local blocks from a batched input tensor.
nn.Fold	Combines an array of sliding local blocks into a large containing tensor.

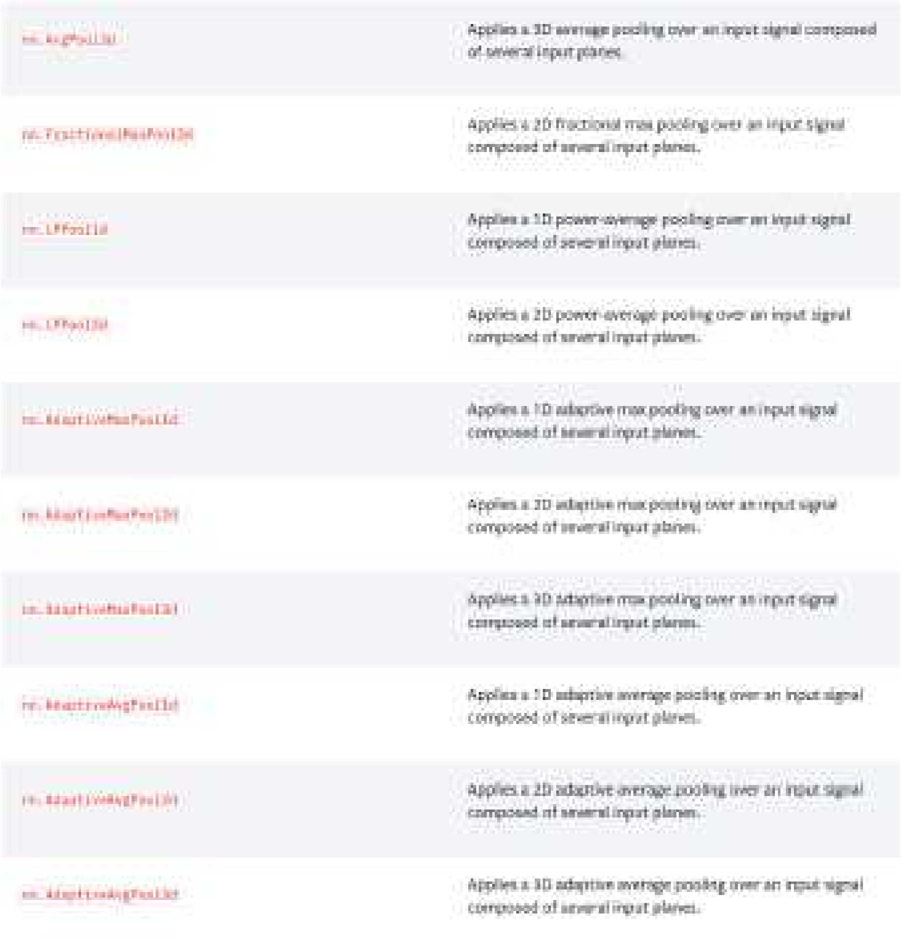
3 Common APIs

torch.nn



Pooling layers

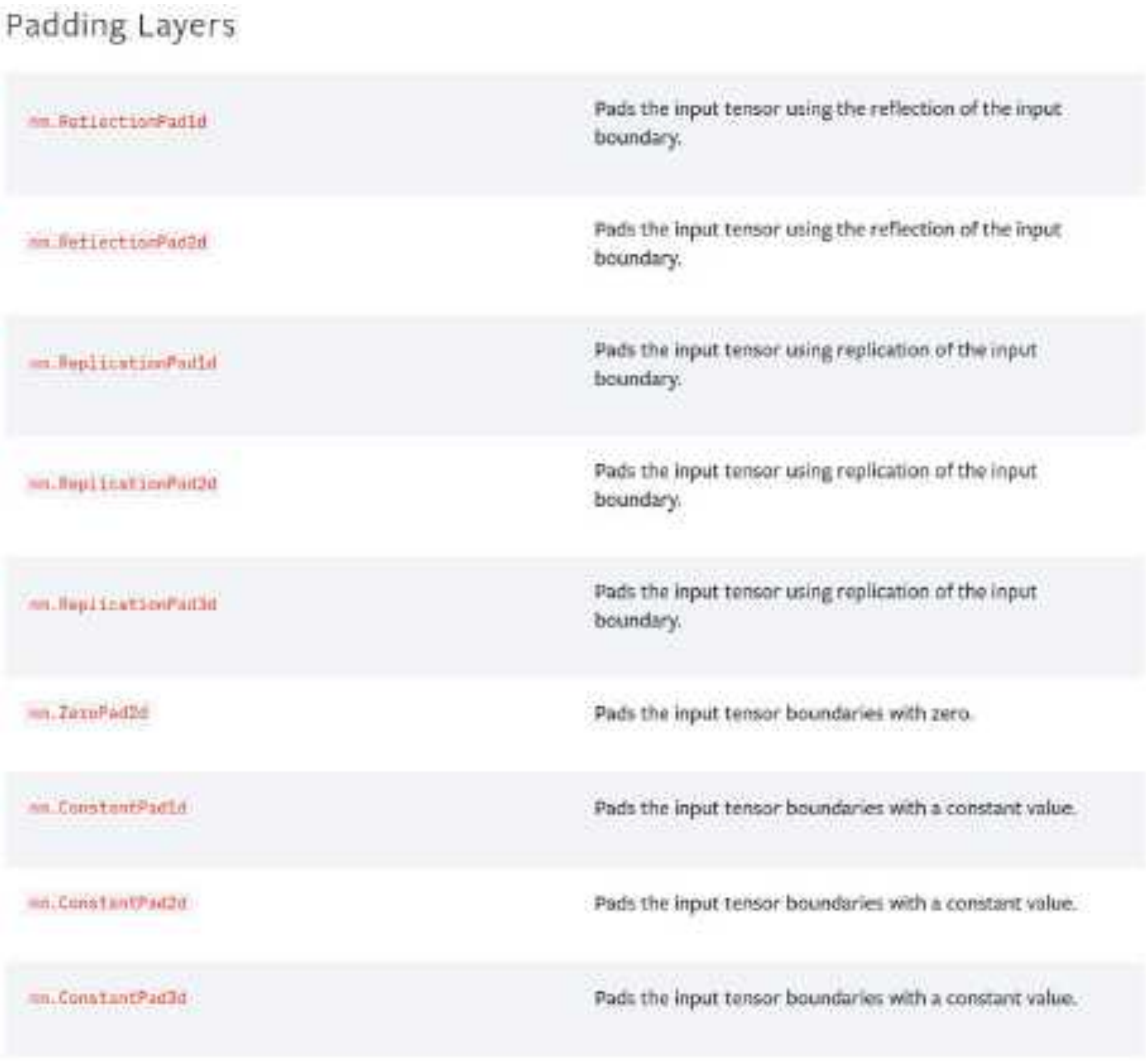
nn.MaxPool1d	Applies a 1D max pooling over an input signal composed of several input planes.
nn.MaxPool2d	Applies a 2D max pooling over an input signal composed of several input planes.
nn.MaxPool3d	Applies a 3D max pooling over an input signal composed of several input planes.
nn.MaxUnpool1d	Computes a partial inverse of MaxPool1d.
nn.MaxUnpool2d	Computes a partial inverse of MaxPool2d.
nn.MaxUnpool3d	Computes a partial inverse of MaxPool3d.
nn.AvgPool1d	Applies a 1D average pooling over an input signal composed of several input planes.
nn.AvgPool2d	Applies a 2D average pooling over an input signal composed of several input planes.



nn.AvgPool3d	Applies a 3D average pooling over an input signal composed of several input planes.
nn.FractionalMaxPool2d	Applies a 2D fractional max pooling over an input signal composed of several input planes.
nn.LPPool1d	Applies a 1D power-average pooling over an input signal composed of several input planes.
nn.LPPool2d	Applies a 2D power-average pooling over an input signal composed of several input planes.
nn.AdaptiveMaxPool1d	Applies a 1D adaptive max pooling over an input signal composed of several input planes.
nn.AdaptiveMaxPool2d	Applies a 2D adaptive max pooling over an input signal composed of several input planes.
nn.AdaptiveMaxPool3d	Applies a 3D adaptive max pooling over an input signal composed of several input planes.
nn.AdaptiveAvgPool1d	Applies a 1D adaptive average pooling over an input signal composed of several input planes.
nn.AdaptiveAvgPool2d	Applies a 2D adaptive average pooling over an input signal composed of several input planes.
nn.AdaptiveAvgPool3d	Applies a 3D adaptive average pooling over an input signal composed of several input planes.

3 Common APIs

torch.nn

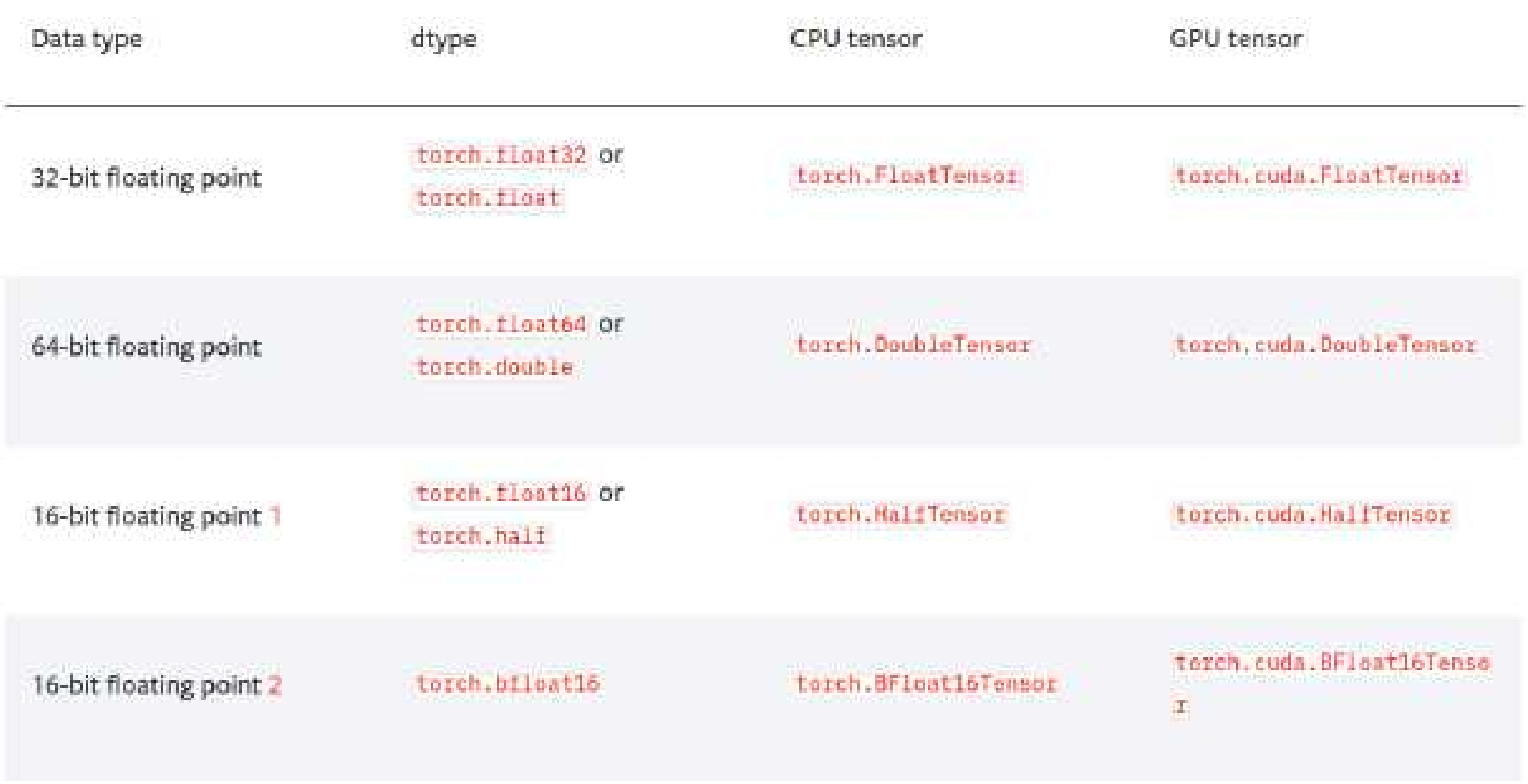


Padding Layers

nn.ReflectionPad1d	Pads the input tensor using the reflection of the input boundary.
nn.ReflectionPad2d	Pads the input tensor using the reflection of the input boundary.
nn.ReplicationPad1d	Pads the input tensor using replication of the input boundary.
nn.ReplicationPad2d	Pads the input tensor using replication of the input boundary.
nn.ReplicationPad3d	Pads the input tensor using replication of the input boundary.
nn.ZeroPad2d	Pads the input tensor boundaries with zero.
nn.ConstantPad1d	Pads the input tensor boundaries with a constant value.
nn.ConstantPad2d	Pads the input tensor boundaries with a constant value.
nn.ConstantPad3d	Pads the input tensor boundaries with a constant value.

3 Common APIs

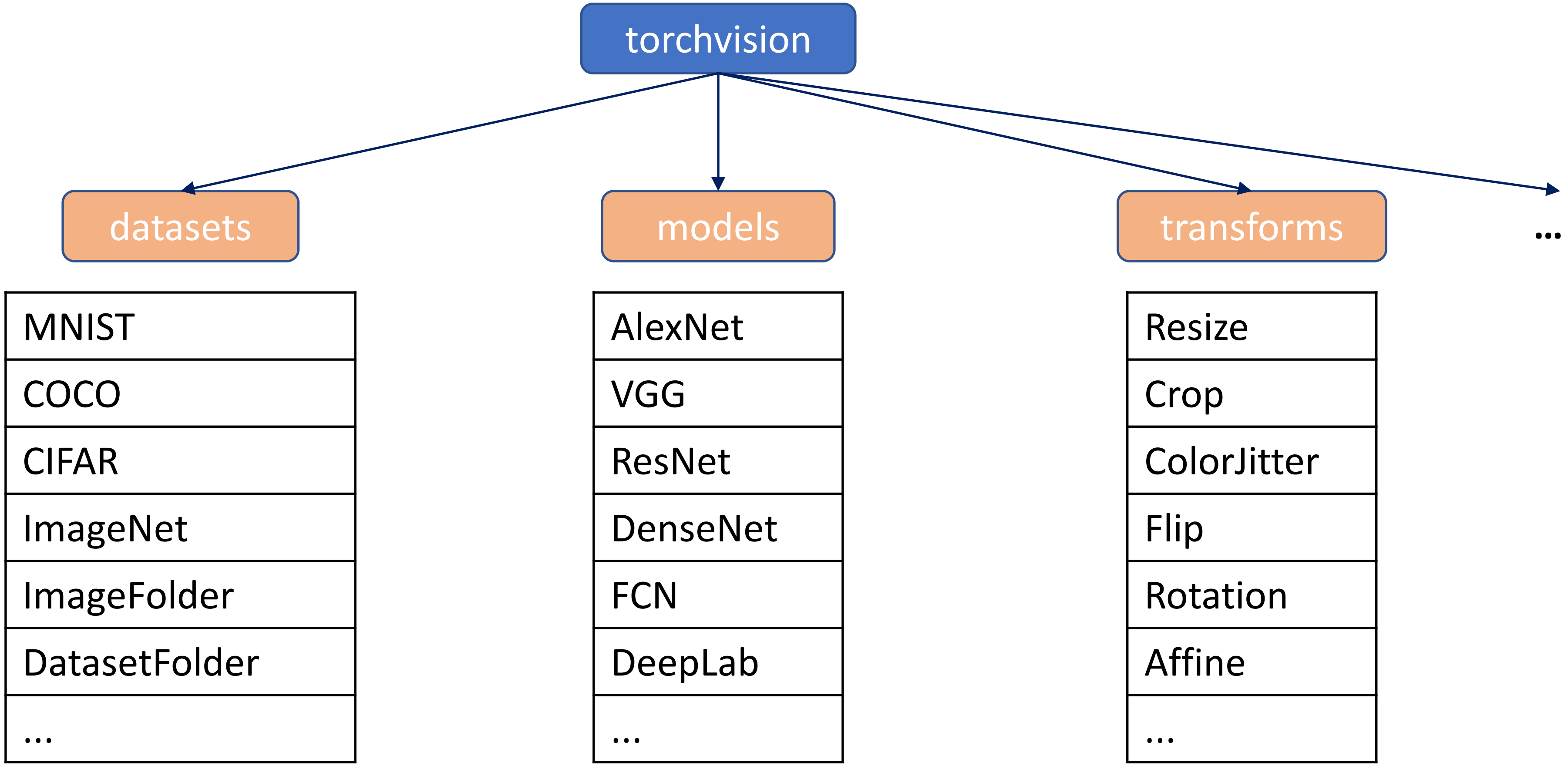
torch.Tensor



Data type	dtype	CPU tensor	GPU tensor
32-bit floating point	`torch.float32` or `torch.float`	`torch.FloatTensor`	`torch.cuda.FloatTensor`
64-bit floating point	`torch.float64` or `torch.double`	`torch.DoubleTensor`	`torch.cuda.DoubleTensor`
16-bit floating point [1]	`torch.float16` or `torch.half`	`torch.HalfTensor`	`torch.cuda.HalfTensor`
16-bit floating point [2]	`torch.bfloat16`	`torch.BFloat16Tensor`	`torch.cuda.BFloat16Tensor`

3 Common APIs

import torchvision



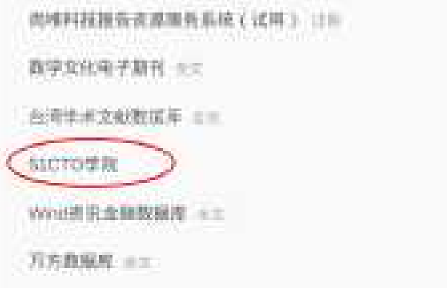
datasets	models	transforms
MNIST	AlexNet	Resize
COCO	VGG	Crop
CIFAR	ResNet	ColorJitter
ImageNet	DenseNet	Flip
ImageFolder	FCN	Rotation
DatasetFolder	DeepLab	Affine
...	...	...

4 Learning resources

Learning resources

1

2


尚唯科技报告资源服务系统（试用）

数字文化电子期刊

台湾华术文献数据库

51CTO学院

Wind资讯金融数据库

万方数据库

3

4 Learning resources

Learning resources

Course

中国大学MOOC

网易云课堂

我的职业课堂

腾讯课堂

bilibili

coursera

Docs

CSDN

知

PyTorch

pytorch.org

Code

GitHub

gitee

Tensorflow-Introduction

1

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.It is used for both research and production at Google.

https://en.wikipedia.org/wiki/TensorFlow

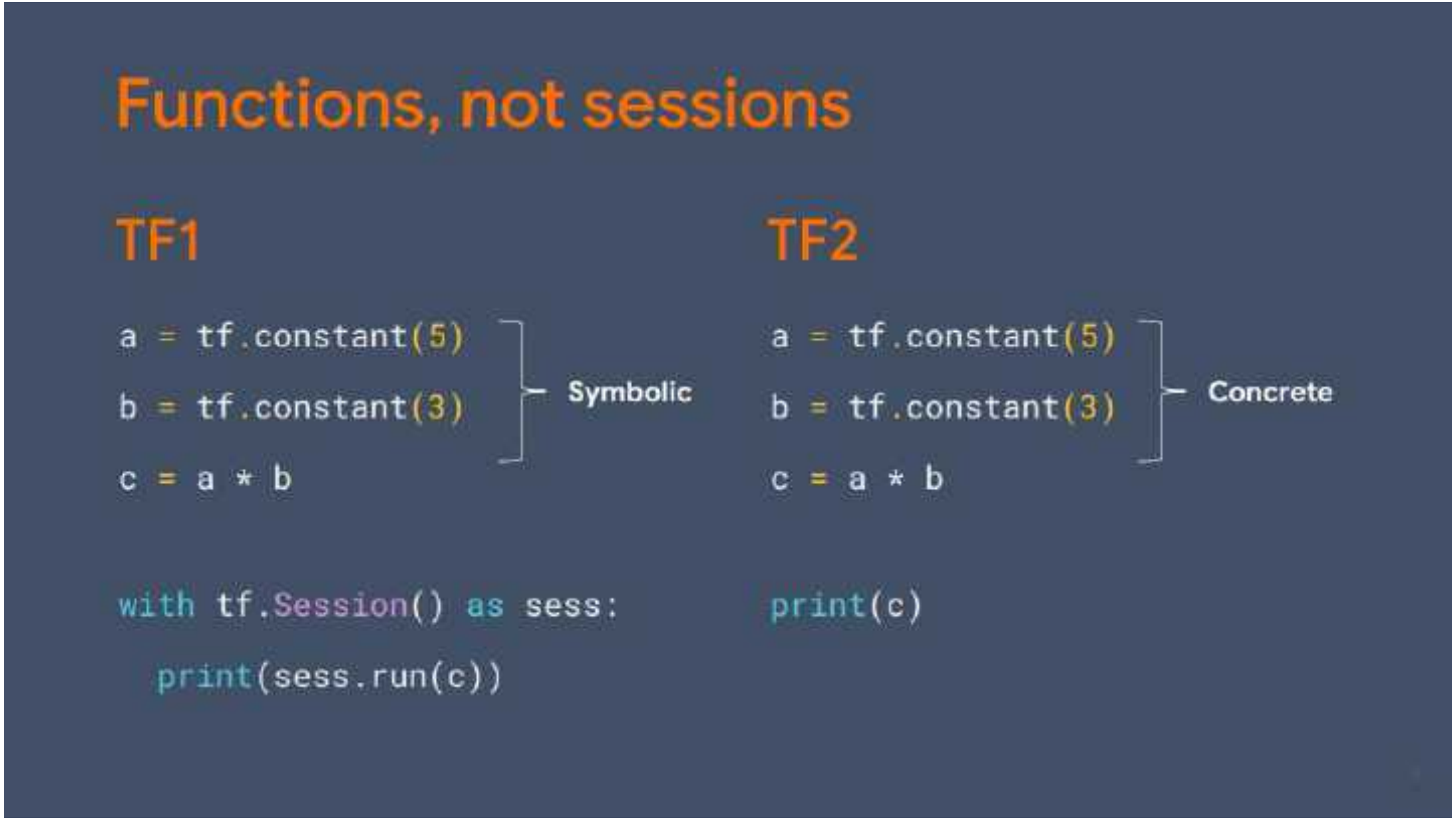
Tensorflow-Development

2015年11月，Google正式发布了Tensorflow的白皮书并开源TensorFlow 0.1 版本。

2017年02月，Tensorflow正式发布了1.0.0版本，同时也标志着稳定版的诞生。

2019年10月，TensorFlow在经历七个多月(2019年3月1日-2019年10月1日)的2.0 Alpha 版本的更新迭代后发布 2.0 正式版。

Tensorflow1.0 vs Tensorflow2.0



Functions, not sessions

TF1

```
a = tf.constant(5)
b = tf.constant(3)
c = a * b

with tf.Session() as sess:
  print(sess.run(c))
```

Symbolic

TF2

```
a = tf.constant(5)
b = tf.constant(3)
c = a * b

print(c)
```

Concrete

Tensorflow1.0 vs Tensorflow2.0

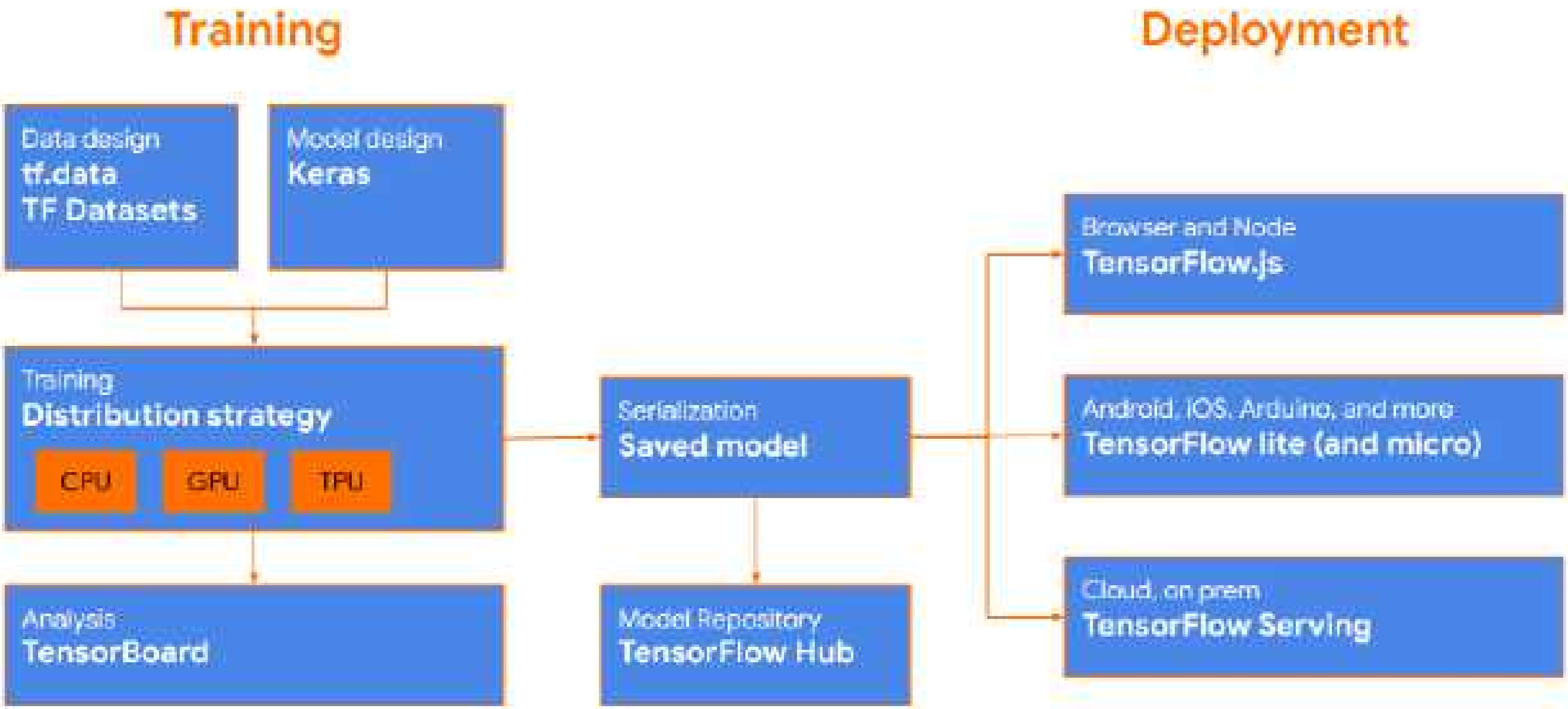
TF 1.x

- Difficult to debug
- API confusion
- It is difficult to get started, but it is still difficult to enter
- A large number of researchers turn to PyTorch

TF 2.x

- TF+Keras
- Easy to use

Tensorflow2.0



https://www.youtube.com/watch?v=5ECD8J3dvDQ

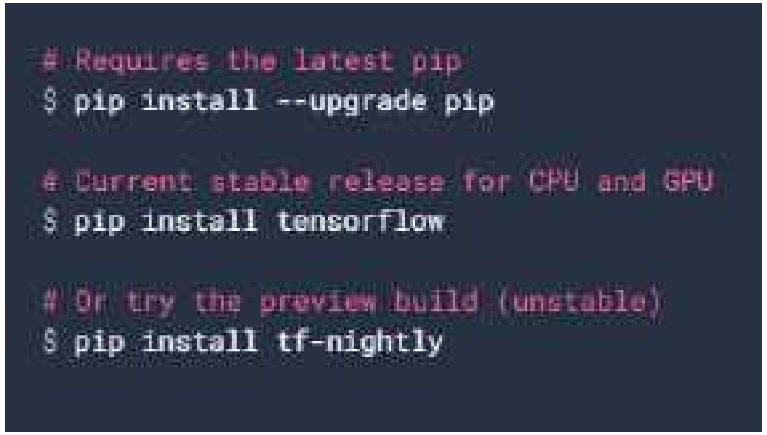
Tensorflow2.0-Install

Install TensorFlow 2

TensorFlow is tested and supported on the following 64-bit

systems:

Python 3.5–3.8
Ubuntu 16.04 or later
Windows 7 or later (with C++ redistributable)
macOS 10.12.6 (Sierra) or later (no GPU support)
Raspbian 9.0 or later



```
# Requires the latest pip
$ pip install --upgrade pip

# Current stable release for CPU and GPU
$ pip install tensorflow

# Or try the preview build (unstable)
$ pip install tf-nightly
```

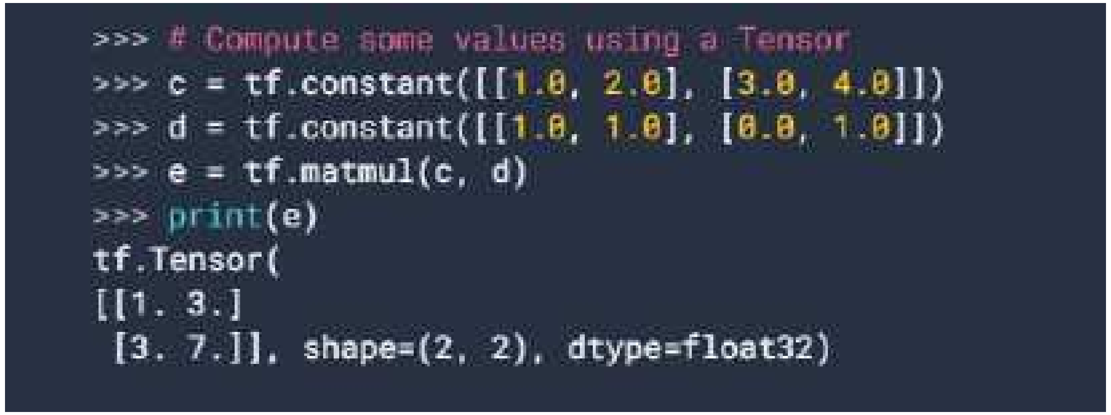
https://tensorflow.google.cn/install

Tensorflow2.0-Tensor

When writing a TensorFlow program, the main object that is manipulated and passed around is the tf.Tensor.

A tf.Tensor has the following properties:

- a single data type (float32, int32, or string, for example)
- a shape

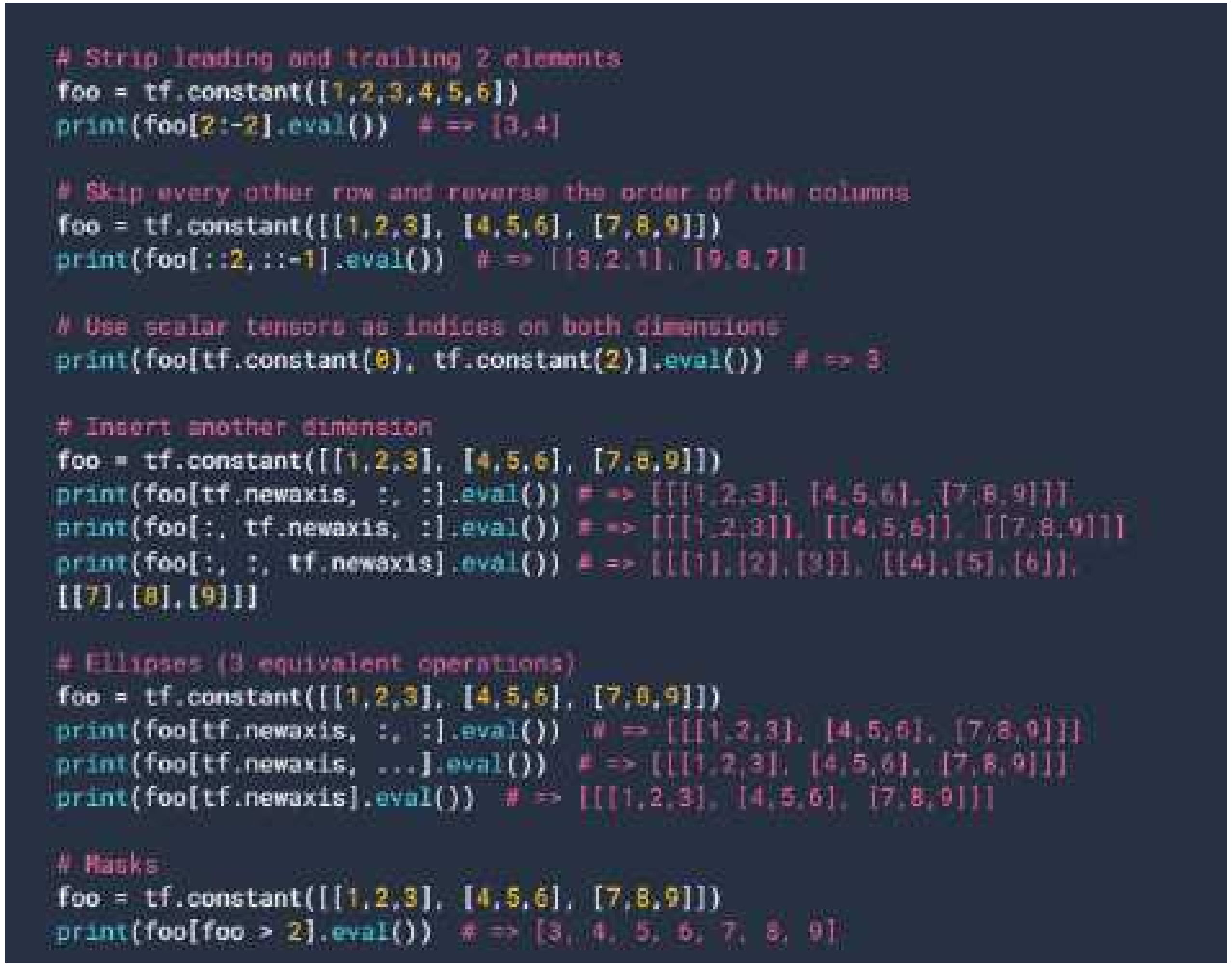


```
>>> # Compute some values using a Tensor
>>> c = tf.constant([[1.0, 2.0], [3.0, 4.0]])
>>> d = tf.constant([[1.0, 1.0], [0.0, 1.0]])
>>> e = tf.matmul(c, d)
>>> print(e)
tf.Tensor(
[[1. 3.]
 [3. 7.]], shape=(2, 2), dtype=float32)
```

https://tensorflow.google.cn/api_docs/python/tf/Tensor

Tensorflow2.0-Tensors

Some useful examples:

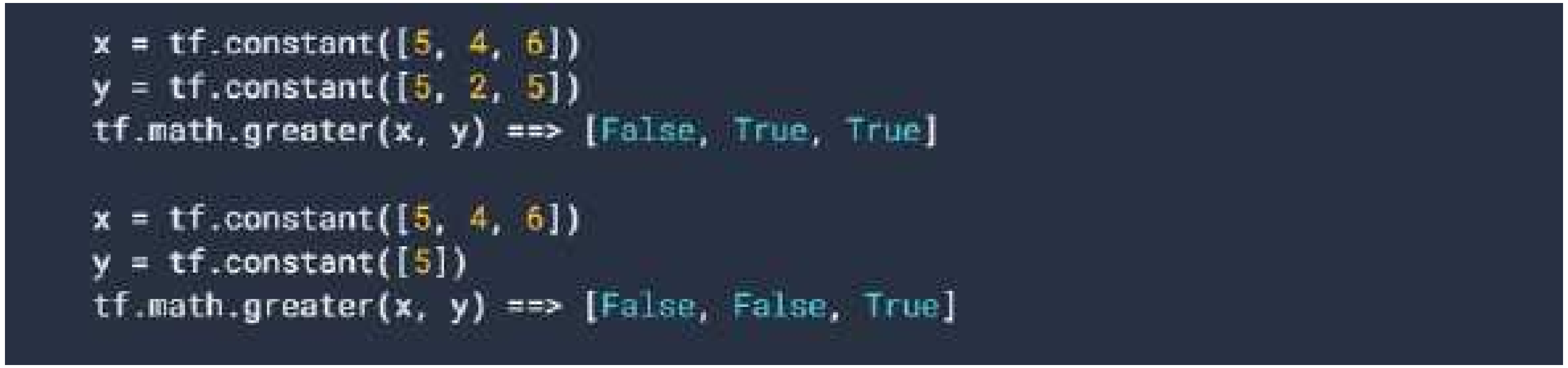


```
# Strip leading and trailing 2 elements
foo = tf.constant([1,2,3,4,5,6])
print(foo[2:-2].eval())  # => [3,4]

# Skip every other row and reverse the order of the columns
foo = tf.constant([[1,2,3], [4,5,6], [7,8,9]])
print(foo[::2,::-1].eval())  # => [[3,2,1], [9,8,7]]

# Use scalar tensors as indices on both dimensions
print(foo[tf.constant(0), tf.constant(2)].eval())  # => 3

# Insert another dimension
foo = tf.constant([[1,2,3], [4,5,6], [7,8,9]])
print(foo[tf.newaxis, :, :].eval()) # => [[[1,2,3], [4,5,6], [7,8,9]]]
print(foo[:, tf.newaxis, :].eval()) # => [[[1,2,3]], [[4,5,6]], [[7,8,9]]]
print(foo[:, :, tf.newaxis].eval()) # => [[[1],[2],[3]], [[4],[5],[6]],
[[7],[8],[9]]]

# Ellipses (3 equivalent operations)
foo = tf.constant([[1,2,3], [4,5,6], [7,8,9]])
print(foo[tf.newaxis, :, :].eval())  # => [[[1,2,3], [4,5,6], [7,8,9]]]
print(foo[tf.newaxis, ...].eval())  # => [[[1,2,3], [4,5,6], [7,8,9]]]
print(foo[tf.newaxis].eval())  # => [[[1,2,3], [4,5,6], [7,8,9]]]

# Masks
foo = tf.constant([[1,2,3], [4,5,6], [7,8,9]])
print(foo[foo > 2].eval())  # => [3, 4, 5, 6, 7, 8, 9]
```

Get items in a tensor

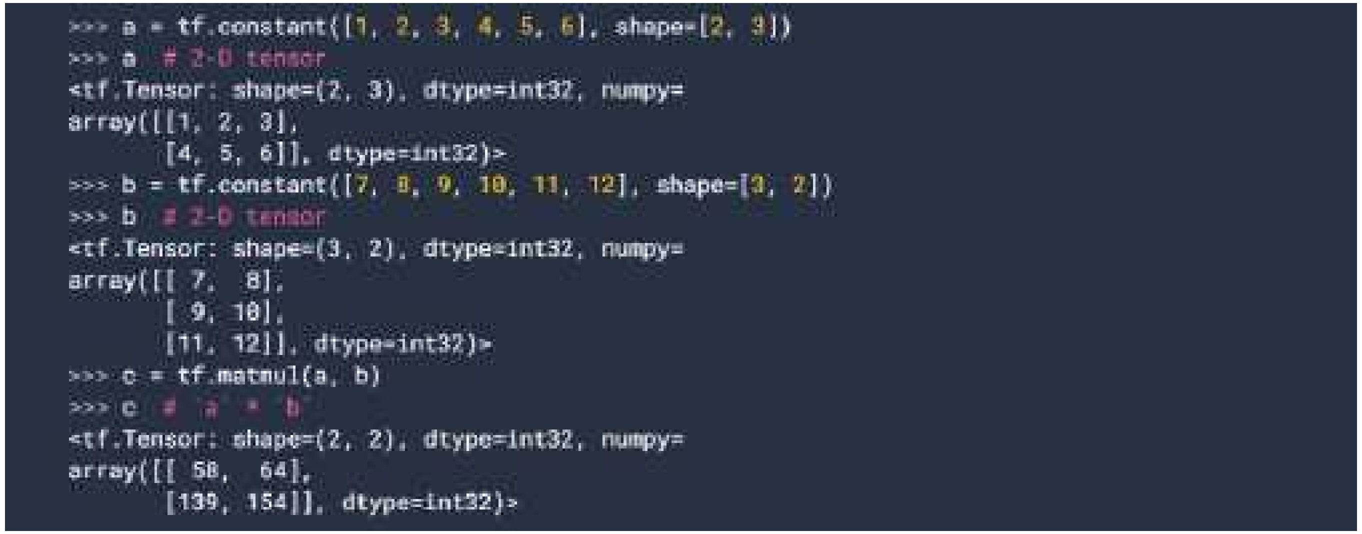
Tensorflow2.0-Tensors

Some useful examples:



```
x = tf.constant([5, 4, 6])
y = tf.constant([5, 2, 5])
tf.math.greater(x, y) ==> [False, True, True]

x = tf.constant([5, 4, 6])
y = tf.constant([5])
tf.math.greater(x, y) ==> [False, False, True]
```

Returns the truth value of (x > y) element-wise.

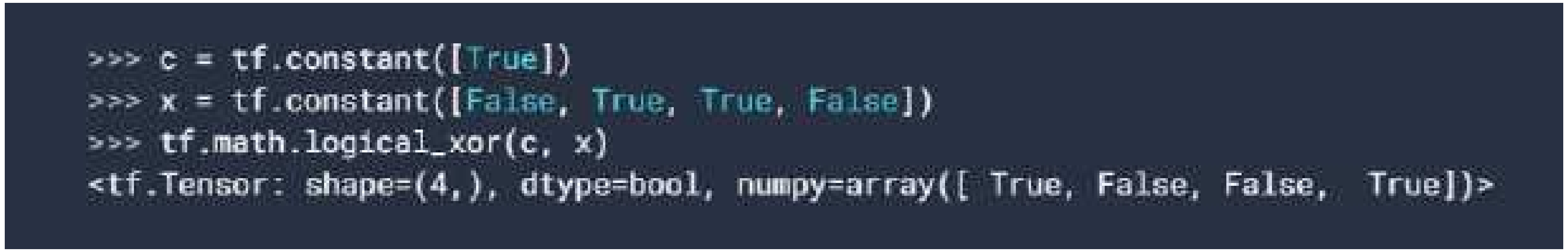


```
>>> a = tf.constant([1, 2, 3, 4, 5, 6], shape=[2, 3])
>>> a  # 2-D tensor
<tf.Tensor: shape=(2, 3), dtype=int32, numpy=
array([[1, 2, 3],
       [4, 5, 6]], dtype=int32)>
>>> b = tf.constant([7, 8, 9, 10, 11, 12], shape=[3, 2])
>>> b  # 2-D tensor
<tf.Tensor: shape=(3, 2), dtype=int32, numpy=
array([[ 7,  8],
       [ 9, 10],
       [11, 12]], dtype=int32)>
>>> c = tf.matmul(a, b)
>>> c  # `a` * `b`
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[ 58,  64],
       [139, 154]], dtype=int32)>
```

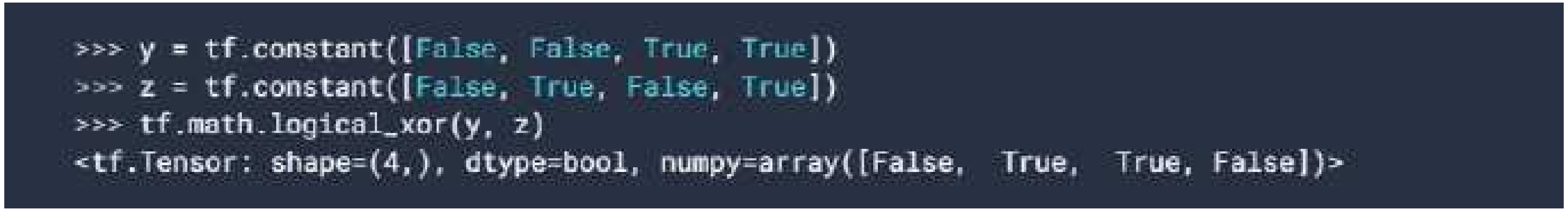
2-D tensor matrix multiplication

Tensorflow2.0-Tensors

Some useful examples:



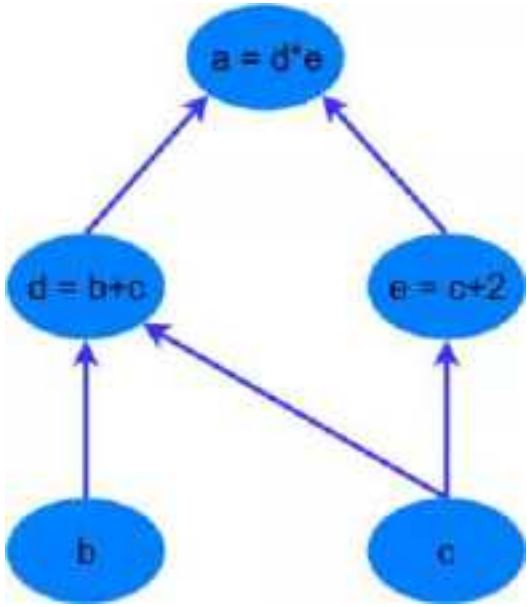
```
>>> c = tf.constant([True])
>>> x = tf.constant([False, True, True, False])
>>> tf.math.logical_xor(c, x)
<tf.Tensor: shape=(4,), dtype=bool, numpy=array([ True, False, False,  True])>
```

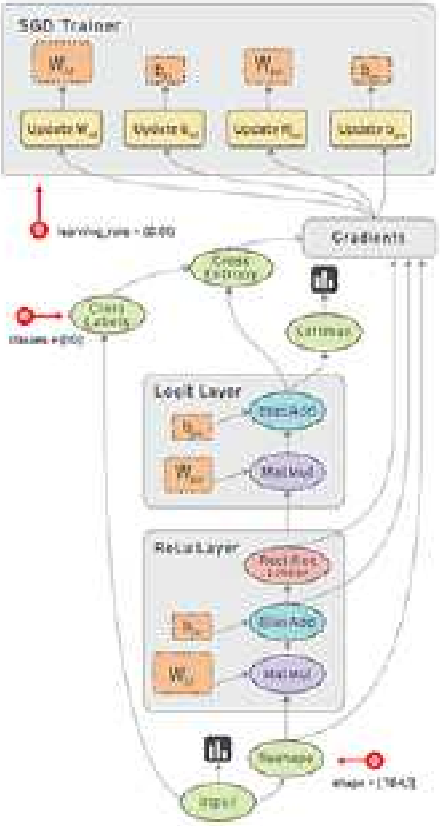


```
>>> y = tf.constant([False, False, True, True])
>>> z = tf.constant([False, True, False, True])
>>> tf.math.logical_xor(y, z)
<tf.Tensor: shape=(4,), dtype=bool, numpy=array([False,  True,  True, False])>
```

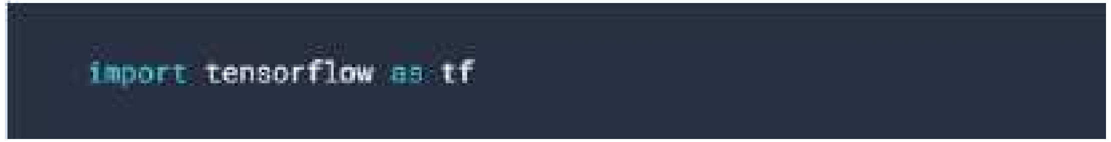
Logical XOR function.

Tensorflow2.0-Computational Graph



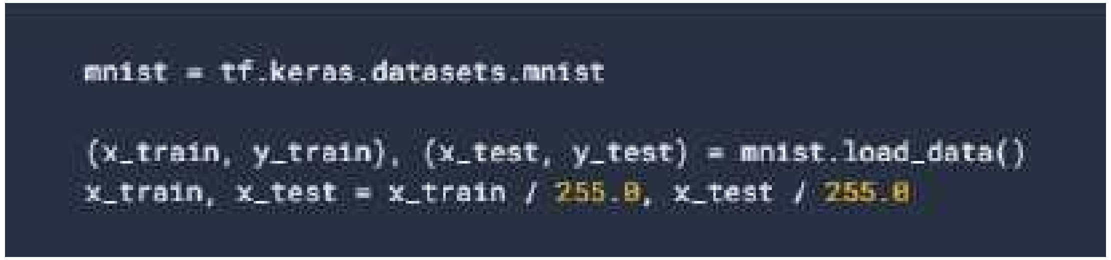


Tensorflow2.0-Example



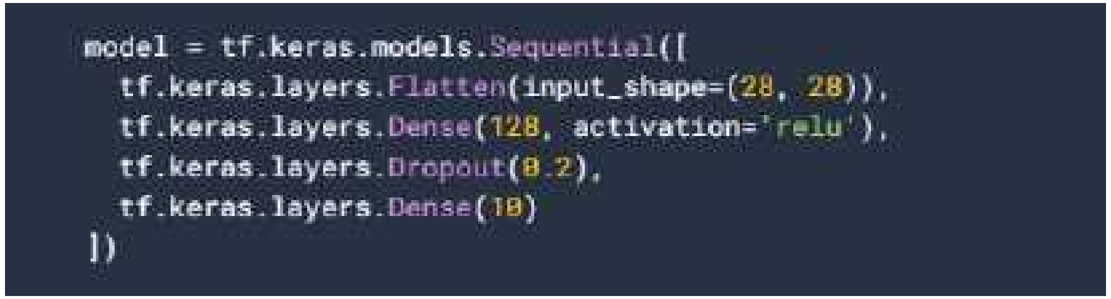
```
import tensorflow as tf
```

Load and prepare the MNIST dataset. Convert the samples from integers to floating-point numbers:



```
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

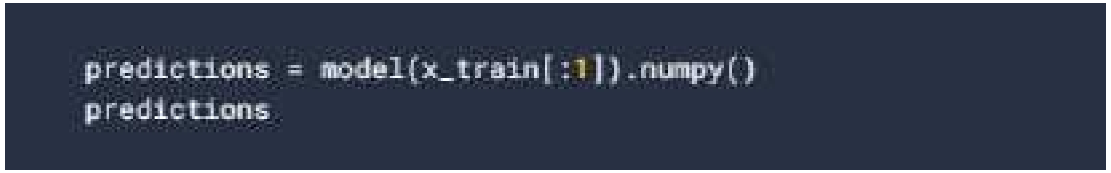
Build the tf.keras.Sequential model by stacking layers. Choose an optimizer and loss function for training:



```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
```

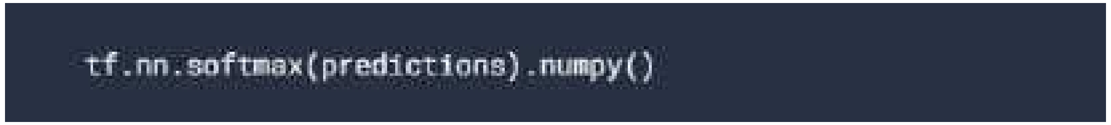
Tensorflow2.0-Example

For each example the model returns a vector of "logits" or "log-odds" scores, one for each class.



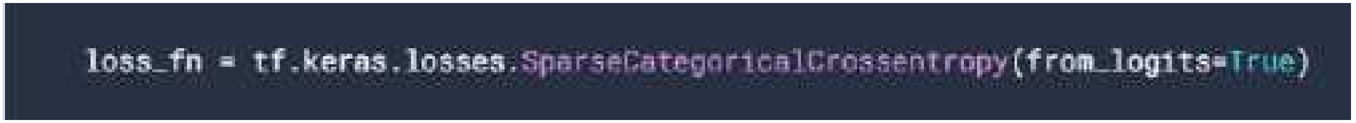
```
predictions = model(x_train[:1]).numpy()
predictions
```

The tf.nn.softmax function converts these logits to "probabilities" for each class:



```
tf.nn.softmax(predictions).numpy()
```

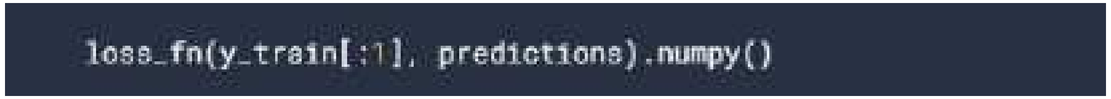
The losses.SparseCategoricalCrossentropy loss takes a vector of logits and a True index and returns a scalar loss for each example.



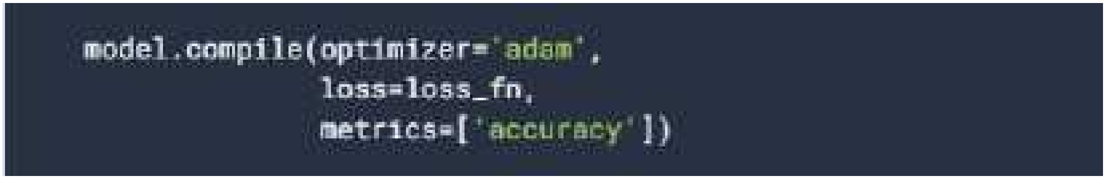
```
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

Tensorflow2.0-Example

This loss is equal to the negative log probability of the true class: It is zero if the model is sure of the correct class.

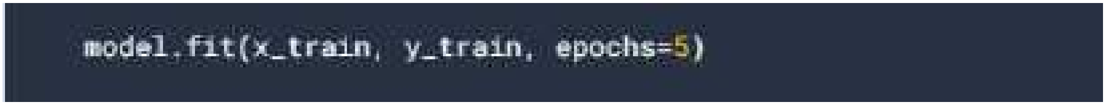


```
loss_fn(y_train[:1], predictions).numpy()
```



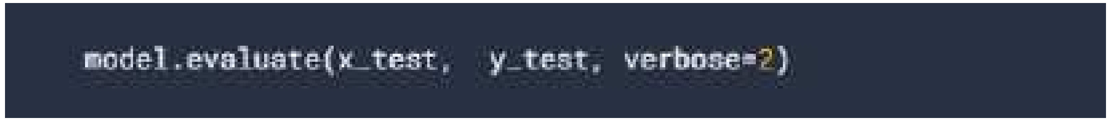
```
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])
```

The Model.fit method adjusts the model parameters to minimize the loss:



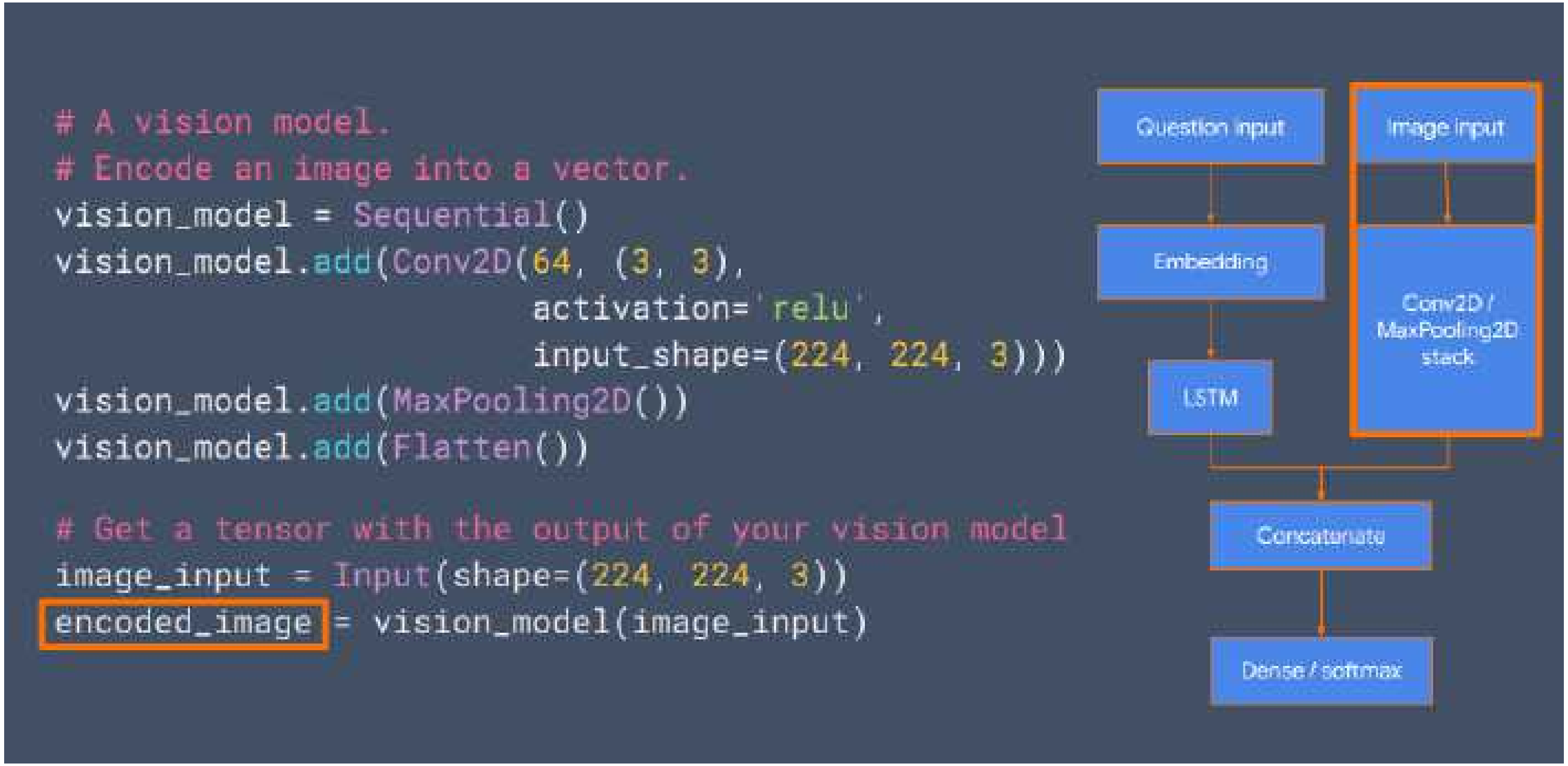
```
model.fit(x_train, y_train, epochs=5)
```

The Model.evaluate method checks the models performance, usually on a "Validation-set" or "Test-set".



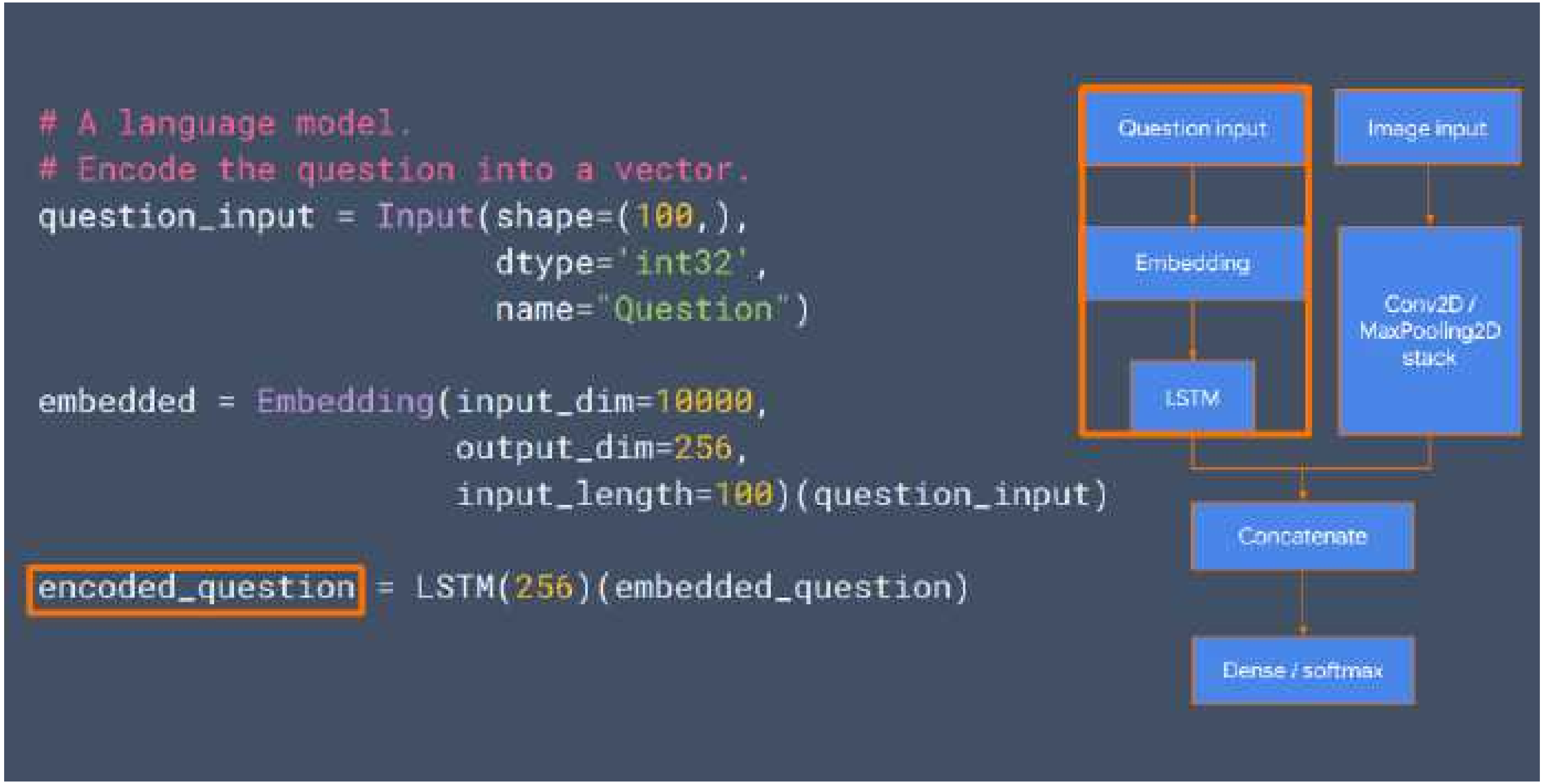
```
model.evaluate(x_test,  y_test, verbose=2)
```

Tensorflow2.0-Multi-input model



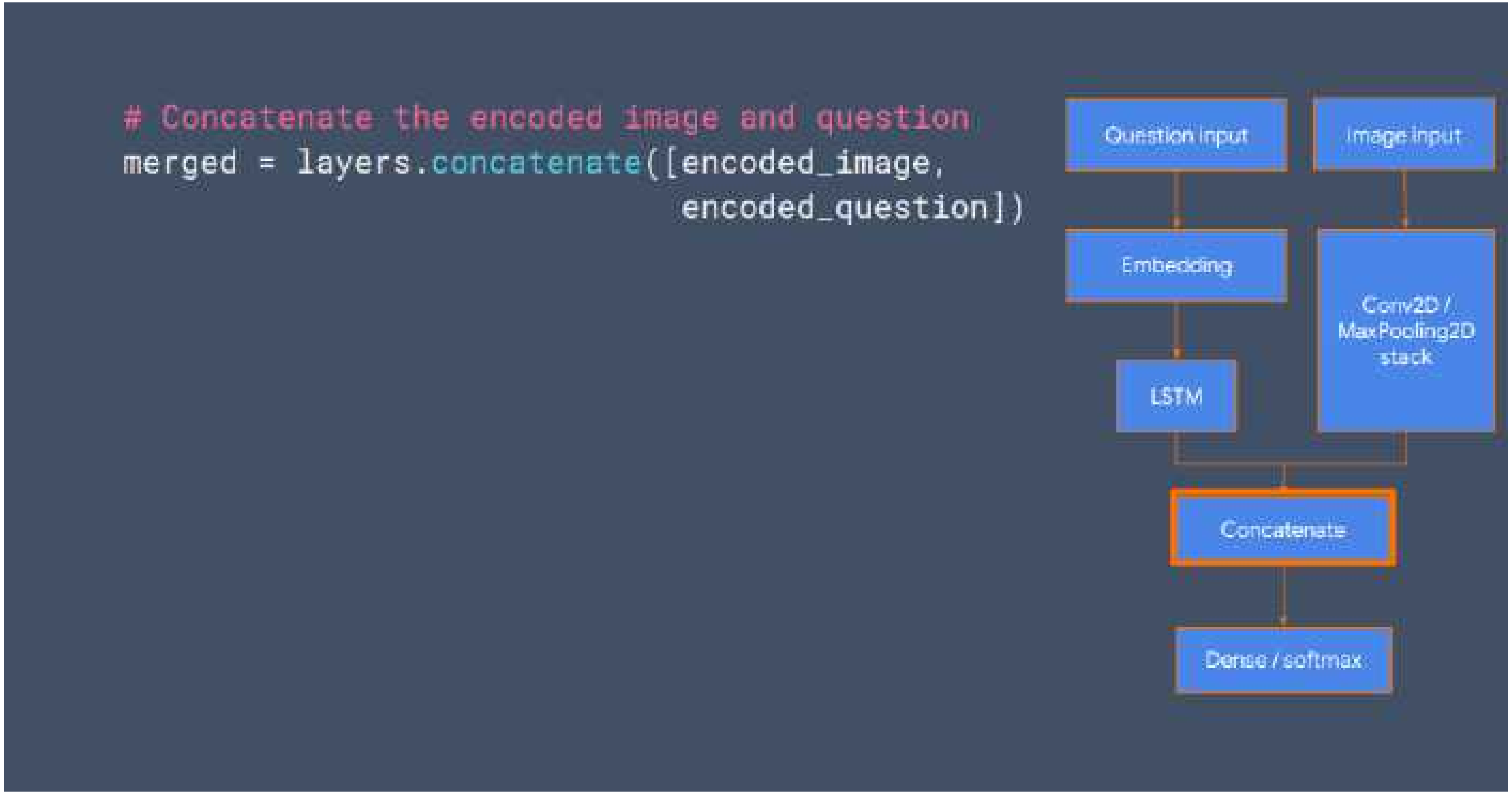
```
# A vision model.
# Encode an image into a vector.
vision_model = Sequential()
vision_model.add(Conv2D(64, (3, 3),
                        activation='relu',
                        input_shape=(224, 224, 3)))
vision_model.add(MaxPooling2D())
vision_model.add(Flatten())

# Get a tensor with the output of your vision model
image_input = Input(shape=(224, 224, 3))
encoded_image = vision_model(image_input)
```

Tensorflow2.0-Multi-input model



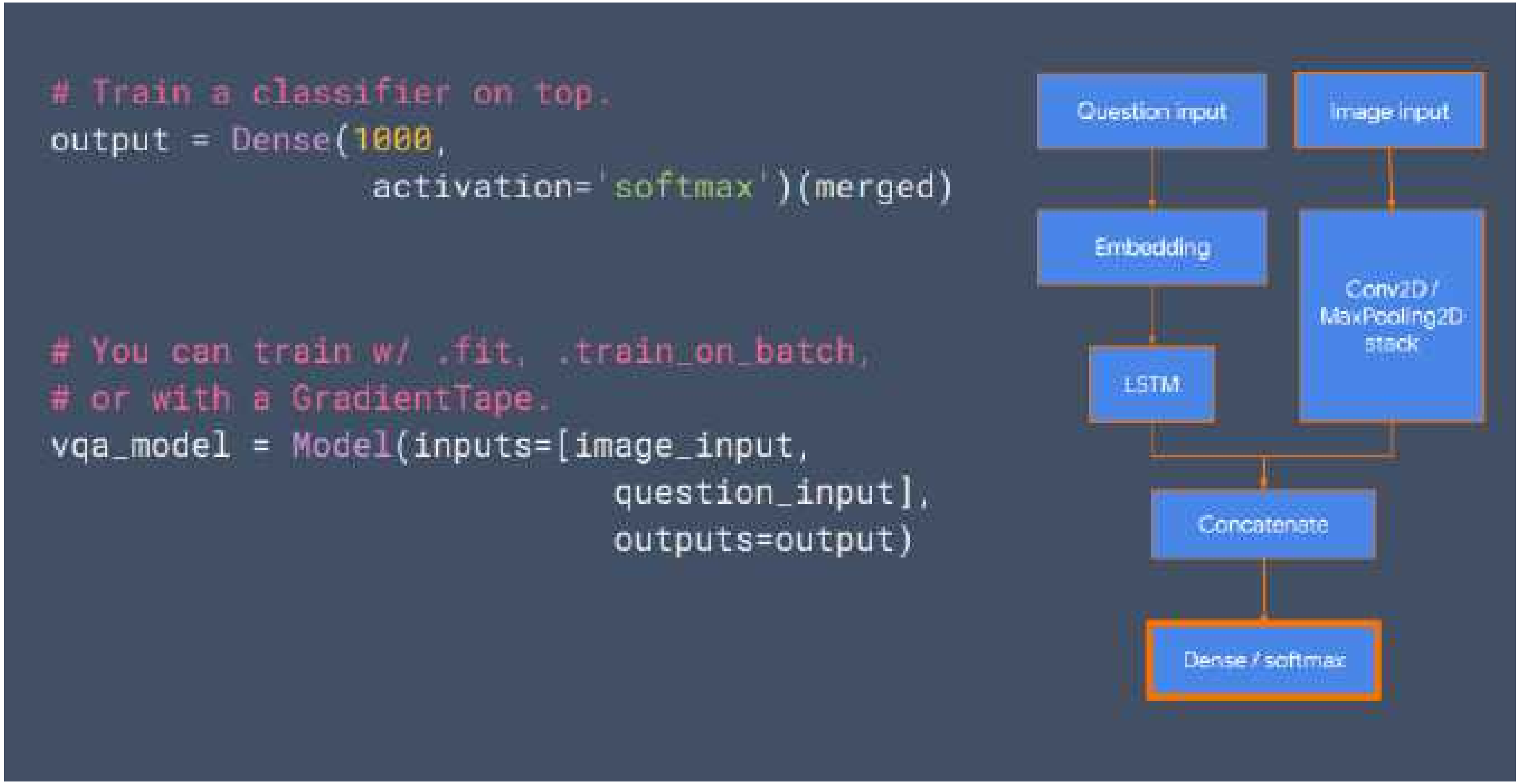
```
# A language model.
# Encode the question into a vector.
question_input = Input(shape=(100,),
                       dtype='int32',
                       name="Question")

embedded = Embedding(input_dim=10000,
                     output_dim=256,
                     input_length=100)(question_input)

encoded_question = LSTM(256)(embedded_question)
```

Tensorflow2.0-Multi-input model



```
# Concatenate the encoded image and question
merged = layers.concatenate([encoded_image,
                             encoded_question])
```

Tensorflow2.0-Multi-input model



```
# Train a classifier on top.
output = Dense(1000,
                activation='softmax')(merged)


# You can train w/ .fit, .train_on_batch,
# or with a GradientTape.
vqa_model = Model(inputs=[image_input,
                          question_input],
                          outputs=output)
```

Tensorflow2.0-Models

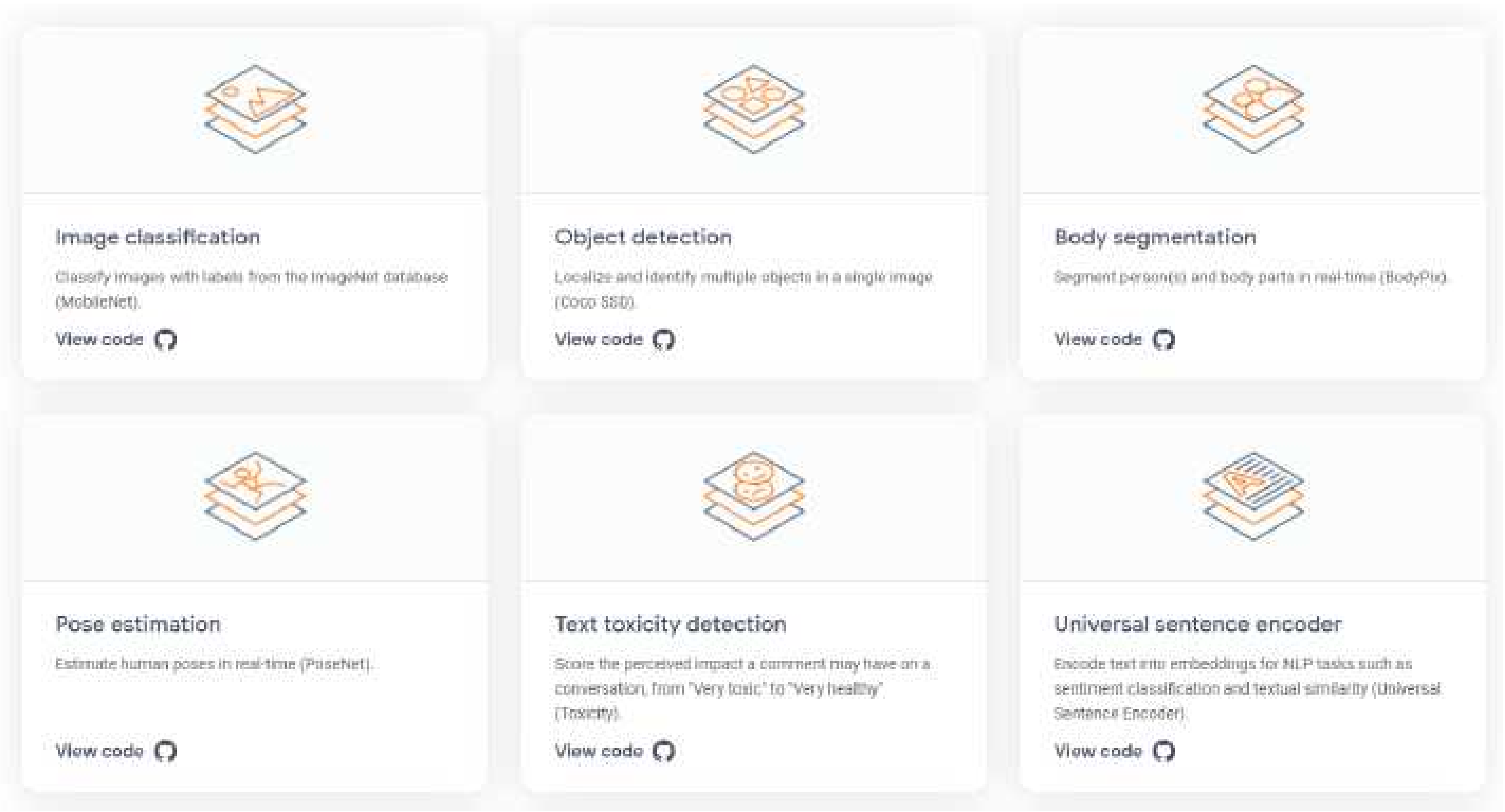


Image classification

Classify images with labels from the ImageNet database (MobileNet).

View code

Object detection

Localize and identify multiple objects in a single image (Coco SSD).

View code

Body segmentation

Segment person(s) and body parts in real-time (BodyPix).

View code

Pose estimation

Estimate human poses in real-time (PoseNet).

View code

Text toxicity detection

Score the perceived impact a comment may have on a conversation, from "Very toxic" to "Very healthy" (Toxicity).

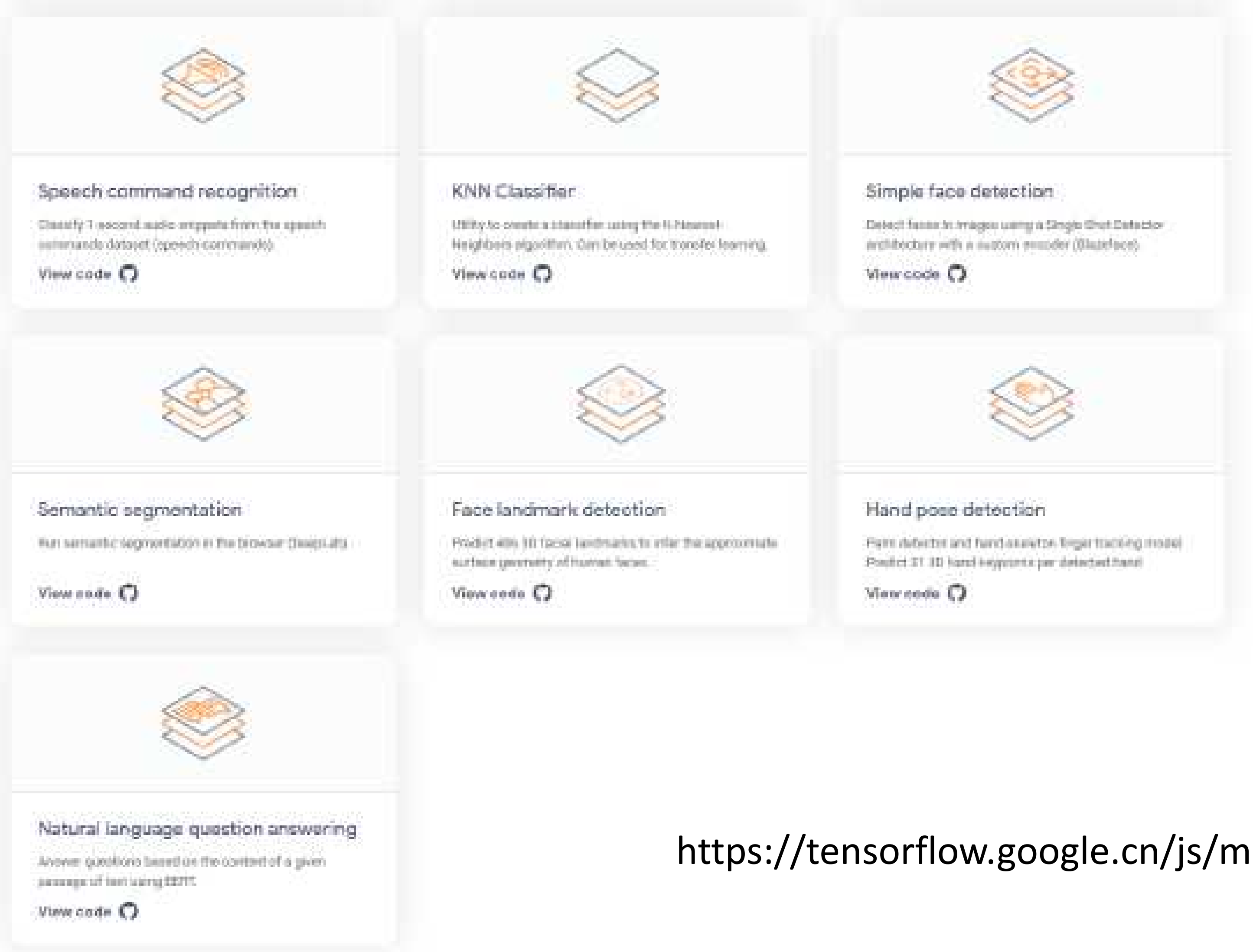
View code

Universal sentence encoder

Encode text into embeddings for NLP tasks such as sentiment classification and textual similarity (Universal Sentence Encoder).

View code

Tensorflow2.0-Models



Speech command recognition

Classify 1 second audio snippets from the speech commands dataset (speech-commands).

View code

KNN Classifier

Utility to create a classifier using the K-Nearest Neighbors algorithm. Can be used for transfer learning.

View code

Simple face detection

Detect faces in images using a Single Shot Detector architecture with a custom encoder (Blazeface).

View code

Semantic segmentation

Run semantic segmentation in the browser (DeepLab).

View code

Face landmark detection

Predict 486 3D facial landmarks to infer the approximate surface geometry of human faces.

View code

Hand pose detection

Palm detector and hand-skeleton finger tracking model. Predict 21 3D hand keypoints per detected hand.

View code

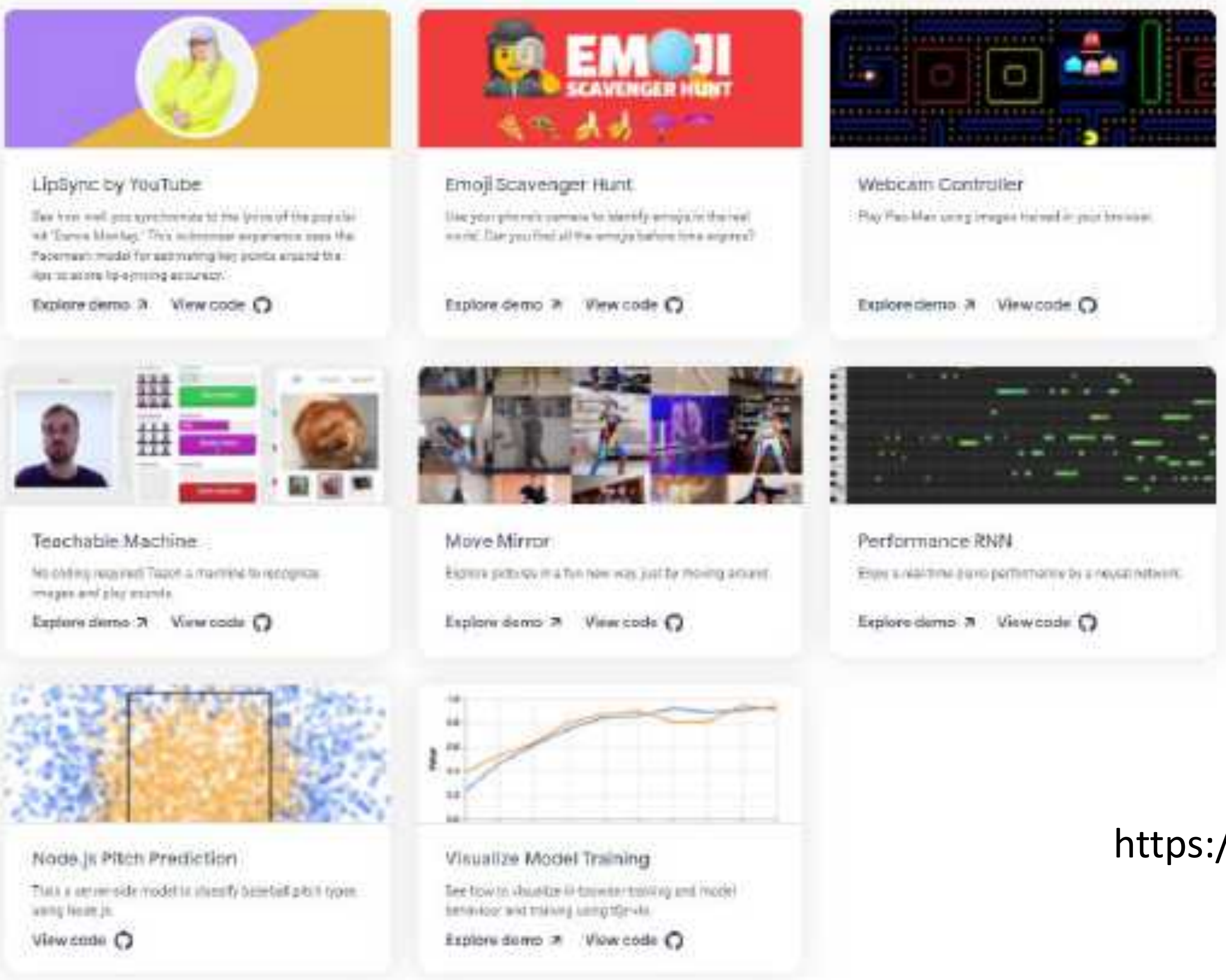
Natural language question answering

Answer questions based on the content of a given passage of text using BERT.

View code

https://tensorflow.google.cn/js/models

Tensorflow2.0-Demos



LipSync by YouTube

See how well you synchronize to the lyrics of the popular hit "Dance Monkey." This in-browser experience uses the Facemesh model for estimating key points around the lips to score lip-syncing accuracy.

Explore demo ↗ View code

Emoji Scavenger Hunt

Use your phone's camera to identify emojis in the real world. Can you find all the emojis before time expires?

Explore demo ↗ View code

Webcam Controller

Play Pac-Man using images trained in your browser.

Explore demo ↗ View code

Teachable Machine

No coding required! Teach a machine to recognize images and play sounds.

Explore demo ↗ View code

Move Mirror

Explore pictures in a fun new way, just by moving around.

Explore demo ↗ View code

Performance RNN

Enjoy a real-time piano performance by a neural network.

Explore demo ↗ View code

Node.js Pitch Prediction

This is a server-side model to classify baseball pitch types using Node.js.

View code

Visualize Model Training

See how to visualize in-browser training and model behavior and training using tfjs-vis.

Explore demo ↗ View code

https://tensorflow.google.cn/js/demos

CS 103 -06

Mid-Term Review and AI Platform Introduction

Jimmy Liu 刘江

TA： 章晓庆， 张洋， 肖尊杰， 杨冰

2020-10-23