علم الانظمة والبيانات

د. محمد الحموري

للطالبة المبدعة
فيحاء الحديدي

إرادة - ثقة - تغيير

Python Basics

Prof. Gheith Abandah

Reference

- Wes McKinney, **Python for Data Analysis**: Data Wrangling with Pandas, NumPy, and IPython, O'Reilly Media, 2nd Edition, 2018.
  - Material: https://github.com/wesm/pydata-book
- Vanderplas, Jacob T. **A Whirlwind Tour of Python**. O'Reilly Media, 2016. https://www.oreilly.com/programming/free/files/a-whirlwind-tour-of-python.pdf
  - https://github.com/jakevdp/WhirlwindTourOfPython/

Outline

- Quick Python Syntax
- Variables and Objects
- Operators
- Built-In Types: Simple Values
- Built-In Data Structures
- Control Flow
- Defining and Using Functions
- Objects and Classes
- Errors and Exceptions
- Iterators
- List Comprehensions
- Generators

Quick Python Syntax

- **Comments** are marked by **#**.
- **Quotation marks** (" ') can also be used to enter comments.
- Use **** to **extend a statement** on the next line.
- Semicolon **;** can optionally **terminate a statement**.

⇒ one line Comments

```
x += 2  # shorthand for x = x + 2
```

```
# Comments
"""
Multi-line comment often
used in documentation
"""

"Single-line Comment"
```

⇒ one line

⇒ multiple line Comments

```
In [2]: x = 1 + 2 + 3 + 4 +\
        5 + 6 + 7 + 8
```

اذا ال Expression تعدى ال line نستخدم ال [\] عشان اكمل ال line يلي بعده وتعتبر واحد Exp.

```
lower = []; upper = []
```

x = 10
y = 20
two line

≡

x = 10;
y = 20;
two line

⇒ ; ⇒ is not a must used in the python but Ex in the c++ is a must use

Quick Python Syntax

- In Python, **code blocks** are denoted by **indentation**.
- **Four spaces** are usually used.
- Which code snippet always prints **x**?

```
for i in range(10):
    if i < midpoint:
        lower.append(i)
    else:
        upper.append(i)
```

```
>>> if x < 4:              >>> if x < 4:
...     y = x * 2          ...     y = x * 2
...     print(x)           ... print(x)
```

this statement only

ب payhon لما نحط فراغ ؤه ال two statement تابعة لل IF

هو جملة ال (print) تابعة لل IF
ح يطبعلي ال (X) فقط إذا أقل من 4

هو ليس تابعة لل IF
ح يطبع (X) سواء أقل أو أكبر من 4

تابعة لل IF
بسبب وجود الفراغ من بداية الجملة

* ب payhon لا يوجد ال [؛]
Er) [؛] y = x * 2
[؛] print (x)

يكفي وضع indentation (فراغ)
يعني Tap

o.o ال [indentation] الفراغات [is important in the payhon]

نفس الشيء في ال loop
ح في while loop المقرر حق فراغات

Quick Python Syntax

- **Parentheses** are for:
    - **Grouping** ⇒ يعني بديا احسب الي في 4 قوس ما اضربها ب 2
    - **Calling**

```
In [5]: 2 * (3 + 4)
Out [5]: 14
```

```
In [6]: print('first value:', 1)
first value: 1
In [7]: print('second value:', 2)
second value: 2
```

⇒ بديا اقسم ال print لل two part

Outline

- Quick Python Syntax
- Variables and Objects
- Operators
- Built-In Types: Simple Values
- Built-In Data Structures
- Control Flow
- Defining and Using Functions
- Objects and Classes
- Errors and Exceptions
- Iterators
- List Comprehensions
- Generators

Variables and Objects

- Python **variables** are **pointers** to **objects**.

⇒ كل الـ Varible عبارة عن pointer to the Object

- **Variable names** can point to objects of **any type**.

```
x = 1          # x is an integer
x = 'hello'    # now x is a string
x = [1, 2, 3]  # now x is a list
```

* يعني الـ (X) ممكن تكون [int] بس احط فيها قيمة واحدة

* ممكن تكون [string] اذا حطيت فيها (hello)

* ممكن تكون [list] أو [array]
من list [Advance Array]

لقدر أحط فيها Int double char أي شيء بدك اياه

Ex) X = 20
type(X)
run → int

X = [5, 2, 'Ali']
type(X)
run → list

X = 20.5
type(X)
run → float

Ex) x = [1,2,5]
y = x
run → y = [1,2,5]
x += [4]
run → x = [1,2,5,4]
run → y = [1,2,5,4]
x = 'Ali'
run → y = [1,2,5,4]

Variables and Objects

type(x) → string

ال (y) ماتغيرت ! لانه حولنا x من list إلى string ح يحلى قيمة ال (y) نفسها

- If we have two variable names pointing to the same mutable object, then changing one will change the other as well!

```
In [2]: x = [1, 2, 3]
        y = x
```

ال y صارت نفس قيمة ال x

```
In [3]: print(y)

[1, 2, 3]
```

لانه list ← التعديل يلي صار بال (x) انتقل لل y

```
In [4]: x.append(4) # append 4 to the list pointed to by x
        print(y) # y's list is modified as well!

[1, 2, 3, 4]

In [5]: x = 'something else'
        print(y)   # y is unchanged

[1, 2, 3, 4]
```

ال (y) مابتوضعته (x) بتضل list

Variables and Objects

[لا تتغير]

- Numbers, strings, and other simple types are immutable.
  - int
  - double

ال list هي array مش (simple type) ← تتغير

```
In [6]: x = 10
        y = x
        x += 5  # add 5 to x's value, and assign it to x
        print("x =", x)
        print("y =", y)

x = 15
y = 10
```

⇒ ال (y) ماتغيرت !! ⇒ number

لكن ال [list] تتغير

Variables and Objects

- **Everything is an object**

```
In [7]:  x = 4
         type(x)
Out [7]: int
In [8]:  x = 'hello'
         type(x)
Out [8]: str
In [9]:  x = 3.14159
         type(x)
Out [9]: float
```

- Object have **attributes** and **methods** accessible through the dot syntax (.)

```
In [10]: L = [1, 2, 3]
         L.append(100)
         print(L)

[1, 2, 3, 100]
```

ال [L] لأنها object
ايالها (list) بقدر استخدم (dot)
مش simple type

* ال simple type ← number, String

* ال object ← list

Outline

- Quick Python Syntax
- Variables and Objects
- Operators
- Built-In Types: Simple Values
- Built-In Data Structures
- Control Flow
- Defining and Using Functions
- Objects and Classes
- Errors and Exceptions
- Iterators
- List Comprehensions
- Generators

Arithmetic Operators

Operator	Name	Description
a + b	Addition	Sum of a and b
a - b	Subtraction	Difference of a and b
a * b	Multiplication	Product of a and b
a / b	True division	Quotient of a and b
a // b	Floor division	Quotient of a and b, removing fractional parts
a % b	Modulus	Remainder after division of a by b
a ** b	Exponentiation	a raised to the power of b
-a	Negation	The negative of a
+a	Unary plus	a unchanged (rarely used)

```
>>> a = 5
>>> b = 3
>>> a / b
1.6666666666666667
>>> a // b
1
>>> a % b
2
```

(قسمة عادية)

أهمل النتيجة

Floor لـ

شيل الإشي بعد الفاصلة

Bitwise Operators

Operator	Name	Description
a & b	Bitwise AND	Bits defined in both a and b
a | b	Bitwise OR	Bits defined in a or b or both
a ^ b	Bitwise XOR	Bits defined in a or b but not both
a << b	Bit shift left	Shift bits of a left by b units
a >> b	Bit shift right	Shift bits of a right by b units
~a	Bitwise NOT	Bitwise negation of a

	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Weight

```
>>> a = 1
>>> b = 2
>>> print( a & b , a | b , a ^ b , b << a , b >> a , ~b )
0 3 3 4 1 -3
```

Comparison Operators

- Return Boolean values **True** or **False**

Operation	Description
a == b	a equal to b
a != b	a not equal to b
a < b	a less than b
a > b	a greater than b
a <= b	a less than or equal to b
a >= b	a greater than or equal to b

```
>>> a = 1
>>> b = 2
>>> print( a == b , a != b , a < b , a > b )
False True True False
```


Assignment Operators

- Assignment is evaluated from **left to right**.

[Right to the left]

3 2 1

```
>>> (i = (j = (k = 10)))
>>> print( i , j , k )
10 10 10
```

- There is an **augmented assignment** operator corresponding to each of the binary arithmetic and bitwise operators.

→ a=a+b → a=a-b ...

a += b	a -= b	a *= b	a /= b
a //= b	a %= b	a **= b	a &= b
a |= b	a ^= b	a <<= b	a >>= b

```
>>> a = 2
>>> b = 10
>>> b += a
>>> print( a , b )
2 12
```

Boolean Operators

T/F Same the C++

- The **Boolean operators** operate on Boolean values:
  - and
  - or
  - not
- Can be used to construct **complex comparisons**.

```
In [15]:  x = 4
          (x < 6) and (x > 2)

Out [15]: True

In [16]:  (x > 10) or (x % 2 == 0)

Out [16]: True

In [17]:  not (x < 6)

Out [17]: False
```


Identity and Membership Operators

Operator	Description
`a is b`	True if a and b are identical objects
`a is not b`	True if a and b are not identical objects
`a in b`	True if a is a member of b
`a not in b`	True if a is not a member of b

```
In [24]:  1 in [1, 2, 3]

Out [24]: True

In [25]:  2 not in [1, 2, 3]

Out [25]: False
```

```
In [19]:  a = [1, 2, 3]
          b = [1, 2, 3]

In [20]:  a == b

Out [20]: True

In [21]:  a is b

Out [21]: False

In [22]:  a is not b

Out [22]: True

In [23]:  a = [1, 2, 3]
          b = a  => the same
          a is b

Out [23]: True
```

Outline

- Quick Python Syntax
- Variables and Objects
- Operators
- Built-In Types: Simple Values
- Built-In Data Structures
- Control Flow
- Defining and Using Functions
- Objects and Classes
- Errors and Exceptions
- Iterators
- List Comprehensions
- Generators

Python Scalar Types

Simple type

Type	Example	Description
int	x = 1	Integers (i.e., whole numbers)
float	x = 1.0	Floating-point numbers (i.e., real numbers)
complex	x = 1 + 2j	Complex numbers (i.e., numbers with a real and imaginary part)
bool	x = True	Boolean: True/False values
str	x = 'abc'	String: characters or text
NoneType	x = None	Special object indicating nulls

null

```
>>> print( int('1') , float(1) , len(str(10)) )
1 1.0 2
```

Integers and Floats

- **Integers** are variable-precision, **no overflow** is possible.

```
>>> 2 ** 90      2^90 = [رح يطلع نتيجه]
1237940039285380274899124224
```

- The **floating-point** type can store fractional numbers. They can be defined either in standard **decimal notation** or in **exponential notation**.

```
In [5]: x = 0.000005
        y = 5e-6
        print(x == y)

True
```

```
In [6]: x = 1400000.00
        y = 1.4e6
        print(x == y)

True
```


Strings

- Strings in Python are created with **single** or **double quotes**.
- The built-in function **`len()`** returns the string length.
- Any character in the string can be accessed through its **index**.

```
>>> s1 = "Hi"
>>> s2 = 'Python'      String
>>> print( s1 + s2 , len( s2 ) , 3 * s1 , s2[0] )
Hi Python 6 Hi Hi Hi  P
```

None and Boolean

[return None] ما بترجع [Value] أبرجعلي Function أي

- **Functions that do not return** value return **None**.
- **None** variables are evaluated to **False**.

اذا حولنا ال none لل bool

in the paython [False = None] ∴ False ح يعطيني

- The **Boolean type** is a simple type with two possible values: **True** and **False**.
- Values are evaluated to **True** unless they are **None**, zero or empty.

```
>>> print( bool(1.5) , bool(0) , bool(None) , bool([]) )
True False False False
```

empty = false

True = zero لل none اي قيمة غير

Outline

- Quick Python Syntax
- Variables and Objects
- Operators
- Built-In Types: Simple Values
- Built-In Data Structures
- Control Flow
- Defining and Using Functions
- Objects and Classes
- Errors and Exceptions
- Iterators
- List Comprehensions
- Generators

Ex) dict حل " إذا المطلوب نص وبده تكرار كل كلمة " ⇒ who/school/m ⇒ Data structure أفضل

Ex

استخدم هو dict

(لأنه فيه ماكانية بين pair of data)

1) كلمة 1 'to': 1 2 ⇒ 'to': [1,2,3]

2) كلمة 2 'and': 1

3) كلمة 3 'to'

إذا إجا لي كلمة (to) كان مرة ، ما بخزنها بحط الـ (to) أول مرة وبصير (2)

(ممكن الـ value في dict تكون array)

الـ Collection يلي بخزن فيها Data مهمة hp

dict في* إذا حطينا الـ key تعد fix string التزامية والا تكون string

Built-In Data Structures

- There are four built in Python data structures.

Type Name	Example	Description
list	[1, 2, 3]	Ordered collection
tuple	(1, 2, 3)	Immutable ordered collection
dict	{'a':1, 'b':2, 'c':3}	Unordered (key,value) mapping
set	{1, 2, 3}	Unordered collection of unique values

list ⇒ عبارة عن array إلها (indeces)

بقدر المستخدم يليها القيم ⇒ شي يلي بضيفه نضاف آخر اشي

tuple: ما بقدر المستخدم يليها — بقدر اضيف عليها او اعدل عليها

وهي "mmutable" "قابلة للتعديل"

+ مش نفس الداتا فيها نفس النوع ممكن "(char + int) بنفس عملية الإدخال"

mix data type

dict: key "a" فيه قيمة واحدة — Key Value

بخزن فيها (pair of Key Value) OR pair of data

set: [بخزن فيها Uniq Value]

فيه Set بـ math ما بنزل

الـ (1) ليكون مكررمرتين !!

Ex) لو عندي (list) وبدي (Uniq Value) اسرع اشي اعمله انه احولها الـ Set

Ex) y = [1,2,5,4]

y.append(45)

run ⇒ y = [1,2,5,4,45]

الـ list عبارة عن Ordered Collection لأنه الرقم يلي بضيفه ح يجي بالنهاية

بـ list بتحفظ الـ order تاع ادخال القيم لـ list

Ex) y[0] = -15

run ⇒ y = [-15,2,5,4,100,45]

ح يحذف الـ 1 لأنه بدلته في -15

عشان هيك الـ list هي mmutable بقدر المستخدم القيم يلي فيها 6 مش زي موجود في tuple !

Lists

بحفظ الـ order تاع الـ element حسب ادخالها في الـ list

- List are **ordered** and **mutable**.
- A list can hold objects of **any type**.
- Python uses **zero-based** indexing.
- Elements at the **end** of the list can be accessed with negative numbers, starting from **-1**.

ح يضيفهم في البداية

```
>>> L = [2, 3, 5, 7]
>>> L.append(11)
>>> print( len(L) , L[0] , L[4] )
5 2 11
>>> L = [15, 16] + L
>>> print( L )
[15, 16, 2, 3, 5, 7, 11]
>>> L.sort()
>>> print( L )
[2, 3, 5, 7, 11, 15, 16]
>>> L = [1, 'two', 3.14, [0, 3, 5]]
>>> L[3]
[0, 3, 5]
>>> L[-2]
3.14
```

⇒ ح يضيفها في الآخر — الطول = 5 — 2 — آخر قيمة لأنه index بيبدأ من zero = 11

0	1	2	3	4	5
2	3	5	7	11	
-5	-4	-3	-2	-1	

2 = [index -5] / [right] لما تبدأ من ناحية الـ [index-1]

11

Ex) Student_tuple = ('Amanda', 'Blue', [98, 75, 87])

Student_tuple [2] [0] = 100 العنصر الثالث عبارة عن list

لو العنصر (3) من الـ tuple
العنصر الأول من list
نقدر اعدل اذا بلي داخل tuple list

run → Student_tuple = ('Ahmad', 'Blue', [100, 75, 87])

Lists

stname, steye, stgrade = Student_tuple ⇒ (unpacking) وزعت الـ tuple بلي فيه (three small object)

Amanda حطلي Blue [98, 75, 87]

run → St name = Amand
run → St eye = Blue

- **Slicing** is a means of accessing multiple values in sub-lists.

  [start : end+1 : inc]

- **Negative step** reverses the list.
- Both indexing and slicing can be used **to set elements** as well as access them.

```
      0  1  2  3  4
>>> L = [2, 3, 5, 7, 11]
>>> L[:]   ⇒  [ list ] بيعطيني كل الـ
[2, 3, 5, 7, 11]
>>> L[:3]  ⇒  بلش من البداية لـ (2)
[2, 3, 5]
>>> L[2:]  ⇒  بلش من (2) للنهاية
[5, 7, 11]
>>> L[1:4] ⇒  بلش من (1) لـ (3)
[3, 5, 7]
>>> L[::2] ⇒  بلش من البداية كل مرة +2
[2, 5, 11]
>>> L[::-1] ⇒ بلش من الآخر للأول
[11, 7, 5, 3, 2]      ... -3 -2 -1
>>> L[0] = 100
>>> L[1:3] = [20, 30]
>>> L
[100, 20, 30, 7, 11]
```


Ex) a_list = [ ] مفتاح لضيف ارقام من 1 لـ 5 [6-1]
#-1 [Final-1] Range
for number in range(1,6):
    a_list += [number] ⇒ a list بكل مرة number بيدخل فيها ⇒ a list [0] = 1, a list [1] = 2 ... 5

run → a_list = [1,2,3,4,5]

لو كان ⇒ for number in range (1,10,2):
    a_list += [number]   +2 (1+2=3, 3+2=5, 5+2=7, 7+2=9)

run → a_list = [1,3,5,7,9]

Ex) t = "moh", 500, 30
type(t)
run → tuple
t[0] = Ali
→ type error 'tuple' object does not support item assignment

Tuples

ما بقدر اعدل على القيم الموجودة فيه بقدر ازيد عليه

- Tuples are **similar to lists**, but are $\boxed{\text{immutable.}}$
- Can be defined with or without parentheses **( )**.

t += (50, 'Ali')
run → t = ('moh', 500, 30, 50, 'Ali')

- **Functions return** multiple values as **tuples**.

بقدر ازيد على الـ tuple اما اعدل عليه لا !

Ex)
index 0 1 2 3 4 5 6
numbers = [2,3,5,7,11,13,17]
numbers [2:6]   6-1=5
→ [5,7,11,13]

الـ Slicing انا بقسم الـ list لأجزاء list

numbers [0:1] 0 (1-1=0)
→ [2]   من index = 0 لـ index = 0

numbers [0:2]
→ [2,3]   2-1=1

```
>>> t = (1, 2, 3)
>>> t = 1, 2, 3
>>> print( t[2], len(t) )
3 3
>>> x = 0.25
>>> x.as_integer_ratio()
(1, 4)
>>> numerator, denominator = x.as_integer_ratio()
>>> print( numerator, denominator )
1 4
```


اذا بدي الـ 13 رقم الـ index = 5
عشان الشرط 6 بدي 6-1 = 5
numbers [:6]
run → [2,3,5,7,11,13]

لو بقل
⇒ [:] ⇒ [رح يعطيني كل الأرقام]
⇒ [: : 2] ⇒ [2, 5, 11, 17] رح يبدأ من الأول وبكل مرة يقفز مرتين (+2)
⇒ [: : -1] ⇒ بلش من النهاية

Dictionaries

- Dictionaries are flexible **mappings** of **keys** to **values**.
- They can be created via a **comma-separated list** of **key:value** pairs within **curly braces**.

```
>>> d = {'Name':'Sami', "Weight":75}
>>> d['Length'] = 1.75
>>> d
{'Name': 'Sami', 'Weight': 75, 'Length': 1.75}
>>> d['Name']
'Sami'
```


Sets

- Sets are **unordered** collections of **unique** items.
- They are defined using curly brackets **{ }**.
- Set operations include **union**, **intersection**, **difference** and **symmetric difference**.

لا يتكرر ال element ال Set

```
>>> primes = {2, 3, 5, 7}
>>> odds = {1, 3, 5, 7, 9}
>>> primes | odds     # Union
{1, 2, 3, 5, 7, 9}
>>> primes & odds     # Intersection
{3, 5, 7}
>>> primes - odds     # Differences
{2}
>>> primes ^ odds     # Symmetric difference
{1, 2, 9}
```

Outline

- Quick Python Syntax
- Variables and Objects
- Operators
- Built-In Types: Simple Values
- Built-In Data Structures
- Control Flow
- Defining and Using Functions
- Objects and Classes
- Errors and Exceptions
- Iterators
- List Comprehensions
- Generators

Conditional Statements: `if`, `elif`, and `else`

- **`if`** statements in Python have optional **`elif`** and **`else`** parts.

[الفراغات مهمة] !!

```
In [1]: x = -15

        if x == 0:
            print(x, "is zero")
        elif x > 0:
            print(x, "is positive")
        elif x < 0:
            print(x, "is negative")
        else:
            print(x, "is unlike anything I've ever seen...")

-15 is negative
```

elseIf ⇐ elif

: بكتب else بالآخر

for Loops

- The **for** loop is repeated for each index returned by the **iterator** after **in**.

```
In [2]: for N in [2, 3, 5, 7]:   ⇒ element/element كل شي بالليست
            print(N, end=' ') # print all on same line

2 3 5 7
```

- The **range()** object is very useful in for loops.

[0-9]

```
In [3]: for i in range(10):
            print(i, end=' ')

0 1 2 3 4 5 6 7 8 9
```


for Loops

- The **range(start, end+1, inc)** has default zero start and unit increment.

```
In [4]: # range from 5 to 10
        list(range(5, 10))

Out [4]: [5, 6, 7, 8, 9]
```

```
In [5]: # range from 0 to 10 by 2
        list(range(0, 10, 2))

Out [5]: [0, 2, 4, 6, 8]
```

while Loops

- The **while** loop iterates as long as the condition is met.

(عامل دايمي للاقتراس)

```
In [6]: i = 0
        while i < 10:
            print(i, end=' ')
            i += 1

0 1 2 3 4 5 6 7 8 9
```


[loop إذا توجدت بـ اي loop تطلع من]

[skip the current iteration ويكمل ال Iteration يلي بعده]

break and continue: Fine-Tuning Your Loops

- The **continue** statement skips the remainder of the current loop, and goes to the next iteration.

```
In [7]: for n in range(20):
            # check if n is even
            if n % 2 == 0:
                continue
            print(n, end=' ')

1 3 5 7 9 11 13 15 17 19
```

Prints odd numbers

Defining Functions

مشان أعرف Function

- Functions are defined with the **def** statement.
- The following function returns a list of the first N Fibonacci numbers.

```
In [4]: def fibonacci(N):
            L = []
            a, b = 0, 1
            while len(L) < N:
                a, b = b, a + b
                L.append(a)
            return L
```

L = [] ⇒ عرفت list فاضية

a, b = 0, 1 ⇒ a=0 / b=1

[N=10]

a, b = b, a + b ⇒ a=b / b=a+b

L.append(a) ⇐ list L ⇐ append لعين نجمّع فيها

- Calling it:

بيستثني اول عددين

```
In [5]:  fibonacci(10)

Out [5]: [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

وبعمل عند مجموع العددين يلي قبل

1+1=2
1+2=3
2+3=5
...

Default Argument Values

- You can have **default values** for arguments.

```
def fibonacci(N, a=0, b=1):
    L = []
    while len(L) < N:
        a, b = b, a + b
        L.append(a)
    return L
```

- It can be called with our without the optional args.

```
fibonacci(10)

[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

```
fibonacci(10, 0, 2)

[2, 2, 4, 6, 10, 16, 26, 42, 68, 110]
```

3 *(IPython Session)* Create a dictionary named `states` that maps three state abbreviations to their state names, then display the dictionary.
Answer:

```
In [1]: states = {'VT': 'Vermont', 'NH': 'New Hampshire',
   ...:           'MA': 'Massachusetts'}
   ...:

In [2]: states
Out[2]: {'VT': 'Vermont', 'NH': 'New Hampshire', 'MA': 'Massachusetts'}
```

6.2.2 Iterating through a Dictionary

The following dictionary maps month-name strings to `int` values representing the numbers of days in the corresponding month. Note that *multiple* keys can have the *same* value:

```
In [1]: days_per_month = {'January': 31, 'February': 28, 'March': 31}
```

اضافة اشي جديد ⇒ days-per-month ["Jan"] = 30

```
In [2]: days_per_month
```

بيعدل الموجود ⇒ days-per-month ["January"] = 30

```
Out[2]: {'January': 31, 'February': 28, 'March': 31}
```

+ 'Jan' = 30] (الحالة الاضافة)

Again, the dictionary's string representation shows the key–value pairs in their insertion order, but this is not guaranteed because dictionaries are *unordered*. We'll show how to process keys in *sorted* order later in this chapter.

The following `for` statement iterates through `days_per_month`'s key–value pairs. Dictionary method **`items`** returns each key–value pair as a tuple, which we unpack into `month` and `days`:

.items: Pair of data (Value / Key) ترجع

.items اعمال dict

```
In [3]: for month, days in days_per_month.items():
   ...:     print(f'{month} has {days} days')
   ...:
January has 31 days
February has 28 days
March has 31 days
```

إجباري

Key Value

قيم ال month

البيانات بتخزن ال record month و days [record : dict]

بتخزن ال month في Key

وال days في Value

[has, days : static words]

Self Check

1 *(Fill-In)* Dictionary method ________ returns each key–value pair as a tuple.
Answer: `items`.

6.2.3 Basic Dictionary Operations

For this section, let's begin by creating and displaying the dictionary `roman_numerals`. We intentionally provide the incorrect value `100` for the key `'X'`, which we'll correct shortly:

```
In [1]: roman_numerals = {'I': 1, 'II': 2, 'III': 3, 'V': 5, 'X': 100}

In [2]: roman_numerals
Out[2]: {'I': 1, 'II': 2, 'III': 3, 'V': 5, 'X': 100}
```

Accessing the Value Associated with a Key

Let's get the value associated with the key `'V'`:

```
In [3]: roman_numerals['V']
Out[3]: 5
```

Updating the Value of an Existing Key–Value Pair

You can update a key's associated value in an assignment statement, which we do here to replace the incorrect value associated with the key 'X':

```
In [4]: roman_numerals['X'] = 10

In [5]: roman_numerals
Out[5]: {'I': 1, 'II': 2, 'III': 3, 'V': 5, 'X': 10}
```

Adding a New Key–Value Pair

Assigning a value to a nonexistent key inserts the key–value pair in the dictionary:

```
In [6]: roman_numerals['L'] = 50

In [7]: roman_numerals
Out[7]: {'I': 1, 'II': 2, 'III': 3, 'V': 5, 'X': 10, 'L': 50}
```

String keys are case sensitive. Assigning to a nonexistent key inserts a new key–value pair. This may be what you intend, or it could be a logic error.

اذا بدي احذف اشي معين من dict

Removing a Key–Value Pair

You can delete a key–value pair from a dictionary with the del statement:

① del [اسم الـ dict][key بدي ازيلها]

```
In [8]: del roman_numerals['III']
```

اسم الـ dict — key

⇒ تلقائياً لما احذف الـ key رح يحذف الـ value

```
In [9]: roman_numerals
Out[9]: {'I': 1, 'II': 2, 'V': 5, 'X': 10, 'L': 50}
```

OR

② pop

You also can remove a key–value pair with the dictionary method **pop**, which returns the value for the removed key:

```
In [10]: roman_numerals.pop('X')
Out[10]: 10
```

→ [بتحدد الشي بدك تمسحه key]

```
In [11]: roman_numerals
Out[11]: {'I': 1, 'II': 2, 'V': 5, 'L': 50}
```

Attempting to Access a Nonexistent Key

Accessing a nonexistent key results in a KeyError:

```
In [12]: roman_numerals['III']
-------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-12-ccd50c7f0c8b> in <module>()
----> 1 roman_numerals['III']

KeyError: 'III'
```

You can prevent this error by using dictionary method **get**, which normally returns its argument's corresponding value. If that key is not found, get returns None. IPython does not display anything when None is returned in snippet [13]. If you specify a second argument to get, it returns that value if the key is not found:

```
In [13]: roman_numerals.get('III')

In [14]: roman_numerals.get('III', 'III not in dictionary')
Out[14]: 'III not in dictionary'

In [15]: roman_numerals.get('V')
Out[15]: 5
```

✓ Self Check

1 *(True/False)* The `==` comparison evaluates to `True` only if both dictionaries have the same key–value pairs in the same order.

Answer: False. The `==` comparison evaluates to `True` if both dictionaries have the same key–value pairs, regardless of their order.

6.2.6 Example: Dictionary of Student Grades

The script in Fig. 6.1 represents an instructor's grade book as a dictionary that maps each student's name (a string) to a list of integers containing that student's grades on three exams. In each iteration of the loop that displays the data (lines 13–17), we unpack a key–value pair into the variables `name` and `grades` containing one student's name and the corresponding list of three grades. Line 14 uses built-in function `sum` to total a given student's grades, then line 15 calculates and displays that student's average by dividing `total` by the number of grades for that student (`len(grades)`). Lines 16–17 keep track of the total of all four students' grades and the number of grades for all the students, respectively. Line 19 prints the class average of all the students' grades on all the exams.

```
1  # fig06_01.py
2  """Using a dictionary to represent an instructor's grade book."""
3  grade_book = {
4      'Susan': [92, 85, 100],
5      'Eduardo': [83, 95, 79],
6      'Azizi': [91, 89, 82],
7      'Pantipa': [97, 91, 92]
8  }
9
10 all_grades_total = 0
11 all_grades_count = 0
12
13 for name, grades in grade_book.items():
14     total = sum(grades)
15     print(f'Average for {name} is {total/len(grades):.2f}')
16     all_grades_total += total
17     all_grades_count += len(grades)
18
19 print(f"Class's average is: {all_grades_total / all_grades_count:.2f}")
```

الـ Value ليست قيمة واحدة بل list

بدي أحسب Avg لكل Key

عرفنا two variable

.item => برجع الـ Keys لحال والـ list لحال

بخزن فيه كل مجموعة
Ex [92.85.100]

اذا بدي اطلع double number
[.2f] معناه بدي العدد (٠) منزلتين بعد الفاصلة

اشي واحد بدل الجمع

```
Average for Susan is 92.33
Average for Eduardo is 85.67
Average for Azizi is 87.33
Average for Pantipa is 93.33
Class's average is: 89.67
```

Fig. 6.1 | Instructor's gradebook dictionary.

6.2.7 Example: Word Counts[2]

The script in Fig. 6.2 builds a dictionary to count the number of occurrences of each word in a string. Lines 4–5 create a string text that we'll break into words—a process known as **tokenizing a string.** Python automatically concatenates strings separated by whitespace in parentheses. Line 7 creates an empty dictionary. The dictionary's keys will be the unique words, and its values will be integer counts of how many times each word appears in text.

```
1  # fig06_02.py
2  """Tokenizing a string and counting unique words."""
3
4  text = ('this is sample text with several words '
5          'this is more sample text with some different words')
6
7  word_counts = {}
8
9  # count occurrences of each unique word
10 for word in text.split():
11     if word in word_counts:
12         word_counts[word] += 1  # update existing key-value pair
13     else:
14         word_counts[word] = 1  # insert new key-value pair
15
16 print(f'{"WORD":<12}COUNT')
17
18 for word, count in sorted(word_counts.items()):
19     print(f'{word:<12}{count}')
20
21 print('\nNumber of unique words:', len(word_counts))
```

```
WORD        COUNT
different   1
is          2
more        1
sample      2
several     1
some        1
text        2
this        2
with        2
words       2
Number of unique words: 10
```

Fig. 6.2 | Tokenizing a string and producing word counts.

Line 10 tokenizes text by calling string method **split**, which separates the words using the method's delimiter string argument. If you do not provide an argument, split uses a space. The method returns a list of tokens (that is, the words in text). Lines 10–14

2. Techniques like word frequency counting are often used to analyze published works. For example, some people believe that the works of William Shakespeare actually might have been written by Sir Francis Bacon, Christopher Marlowe or others. Comparing the word frequencies of their works with those of Shakespeare can reveal writing-style similarities. We'll look at other document-analysis techniques in the "Natural Language Processing (NLP)" chapter.

```
In [1]: months = {'January': 1, 'February': 2, 'March': 3}

In [2]: months2 = {number: name for name, number in months.items()}

In [3]: months2
Out[3]: {1: 'January', 2: 'February', 3: 'March'}
```

Curly braces delimit a *dictionary comprehension*, and the expression to the left of the `for` clause specifies a key–value pair of the form *key*: *value*. The comprehension iterates through `months.items()`, unpacking each key–value pair tuple into the variables `name` and `number`. The expression `number: name` reverses the key and value, so the new dictionary maps the month numbers to the month names.

What if `months` contained *duplicate* values? As these become the keys in `months2`, attempting to insert a *duplicate* key simply updates the existing key's value. So if `'February'` and `'March'` both mapped to 2 originally, the preceding code would have produced

```
{1: 'January', 2: 'March'}
```

A dictionary comprehension also can map a dictionary's values to new values. The following comprehension converts a dictionary of names and lists of grades into a dictionary of names and grade-point averages. The variables `k` and `v` commonly mean *key* and *value*:

```
In [4]: grades = {'Sue': [98, 87, 94], 'Bob': [84, 95, 91]}

In [5]: grades2 = {k: sum(v) / len(v) for k, v in grades.items()}

In [6]: grades2
Out[6]: {'Sue': 93.0, 'Bob': 90.0}
```

The comprehension unpacks each tuple returned by `grades.items()` into `k` (the name) and `v` (the list of grades). Then, the comprehension creates a new key–value pair with the key `k` and the value of `sum(v) / len(v)`, which averages the list's elements.

✓ Self Check

1 *(IPython Session)* Use a dictionary comprehension to create a dictionary of the numbers 1–5 mapped to their cubes:

Answer:

```
In [1]: {number: number ** 3 for number in range(1, 6)}
Out[1]: {1: 1, 2: 8, 3: 27, 4: 64, 5: 125}
```

6.3 Sets

A set is an unordered collection of *unique* values. Sets may contain only immutable objects, like strings, `int`s, `float`s and tuples that contain only immutable elements. Though sets are iterable, they are not sequences and do not support indexing and slicing with square brackets, `[]`. Dictionaries also do not support slicing.

Creating a Set with Curly Braces

The following code creates a set of strings named `colors`:

```
In [1]: colors = {'red', 'orange', 'yellow', 'green', 'red', 'blue'}
```

~~Cout~~ Colors? ⇒ data stru. بعرف شو ⇒ الجواب set

ليش set مش dict ؟! لأنه مافي Key

```
In [2]: colors
Out[2]: {'blue', 'green', 'orange', 'red', 'yellow'}
```

set ما بنفع للـ list ، بنفع للـ ⇐ Colors += 'black' ⇒ set ما بقدر اضيف بهاي الحركة ⓧ

Colors | "black" ⇒ set بهاي الطريقة بنفع للـ ⇐ (ما بعدل مباشرة)

لأنه هي set مش list

newSet = colors | {'black'} ⇒ [set عشان اضيف عـ / save اعمل]

Notice that the duplicate string `'red'` was ignored (without causing an error). An important use of sets is **duplicate elimination**, which is automatic when creating a set. Also, the resulting set's values are *not* displayed in the same order as they were listed in snippet `[1]`. Though the color names are displayed in sorted order, sets are *unordered*. You should not write code that depends on the order of their elements.

Determining a Set's Length

You can determine the number of items in a set with the built-in `len` function:

```
In [3]: len(colors)
Out[3]: 5
```

Checking Whether a Value Is in a Set

You can check whether a set contains a particular value using the `in` and `not in` operators:

```
In [4]: 'red' in colors
Out[4]: True

In [5]: 'purple' in colors
Out[5]: False

In [6]: 'purple' not in colors
Out[6]: True
```

Iterating Through a Set

Sets are iterable, so you can process each set element with a `for` loop:

```
In [7]: for color in colors:
   ...:     print(color.upper(), end=' ')
   ...:
RED GREEN YELLOW BLUE ORANGE
```

Sets are *unordered*, so there's no significance to the iteration order.

Creating a Set with the Built-In set Function

You can create a set from another collection of values by using the built-in **set** function—here we create a list that contains several duplicate integer values and use that list as set's argument:

```
In [8]: numbers = list(range(10)) + list(range(5))

In [9]: numbers
Out[9]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4]

In [10]: set(numbers)
Out[10]: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

من ٠-٩ من ٠-٤

فيه تكرار بين [٠-٩]، [٠-٤]
عشان اشيل التكرار
نعمل Set
للـ numbers

If you need to create an empty set, you must use the set function with empty parentheses, rather than empty braces, `{}`, which represent an empty dictionary:

```
In [11]: set()
Out[11]: set()
```

Python displays an empty set as `set()` to avoid confusion with Python's string representation of an empty dictionary (`{}`).

Hashemite University

Computer Engineering Department

Data Science

Array-Oriented Programming with NumPy and Pandas

Dr. Mohammad Al-hammouri

NumPy (Numerical Python)

- High-performance, richly functional *n*-dimensional array type called **ndarray**
- Open-source libraries that the Anaconda Python distribution installs.
- Operations on arrays are up to two orders of magnitude faster than those on lists
- import numpy as np] ⇒ include "library"
- numbers = np.array([2, 3, 5, 7, 11]) ⇒ عرفت array اسمها numbers من النوع np موجود فيها الـ element من [2,3,...,11]

دخلت list وحولتها لـ np array

object

```
In [3]: type(numbers)   (النوع)
Out[3]: numpy.ndarray

In [4]: numbers   (المحتوى) ج
Out[4]: array([ 2,  3,  5,  7, 11])
```

NumPy (Array attributes)

(two dim. array)

- integers = np.array([[1, 2, 3], [4, 5, 6]])

two row
1 2 3
4 5 6

- floats = np.array([0.0, 0.1, 0.2, 0.3, 0.4])

برجع النوع — برجع dimin. — كم عدد الاسطر والاعمدة

- (.dtype), (.ndim), (.shape), (.size)

كم element فيها

- Iterating Through a Multidimensional

كيف اعمل (iterate) داخل (2D) array !?

```
for row in integers:
    for column in row:
        print(column, end=' ')
    print()
```

⇒ 1 2 3
4 5 6

Filling arrays with Specific Values

ال NumPy بقدر اعبيها ب zeros ب ones ب قيمة معينة

- np.zeros(5)
- np.ones((2, 4), dtype=int)
- np.full((3, 5), 13)
- np.arange(5), np.arange(5,10), np.arange(10, 1, -2)
- np.arange(1, 21).reshape(4, 5), np.arange(1, 100001).reshape(4, 25000)

List vs. array Performance: Introducing %timeit

عملت مقارنة بين الـ list والـ Numpy
من خلال الـ [%timeit]

List:

```
In [1]: import random

In [2]: %timeit rolls_list = \
   ...:     [random.randrange(1, 7) for i in range(0, 6_000_000)]
6.29 s ± 119 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

مدة وقت

NumPy Array ⇒ فيه الـ Performance اسرع بكتير من الـ list

```
In [3]: import numpy as np

In [4]: %timeit rolls_array = np.random.randint(1, 7, 6_000_000)
72.4 ms ± 635 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

مدة وقت

الفرق في الوقت جدا مهم لما يكون هنا Data كبيرة

5

NumPy Calculation Methods

- Grades = np.array([[87, 96, 70], [100, 87, 90],
- [94, 77, 90], [100, 81, 82]])
- Grades. (sum(), min(), mean() …)
- numbers = np.arange(1, 10)
- numbers * 2, numbers += 10, numbers >=6

number
1
1
1
5
5
5
6
6

number == 3]

تعتبر الطريقة لـ array
لما نوخذ جزء من array

- **Slicing**:

Grades[0, 1], Grades[0:2], Grades[[1, 3]], Grades[:, 0], Grades[:, 1:3]

6

Shallow Copies(views), Deep Copies

Shallow Copies(views)

- numbers = np.arange(1, 8)
- numbers2 = numbers
- *numbers2* array is a view from *numbers*
- *numbers2* views the *same* data as *numbers*

Deep Copies:

numbers = np.arange(1, 20)

[numbers2 = numbers.copy()]

Change on numbers will not affect numbers2

7

رح نتعرف على الـ Numpy (more advance) (more powerful) الـ

Intro to Data Science: pandas Series and DataFrames

عبارة عن array — (د انا ما نفس النوع) → متجانس

- NumPy's array is optimized for homogeneous numeric data that's accessed via integer indices. Ex) Grades[:, 0:2] [integer indices] لنبحث الـ لحتى اعمل Access
- Data science presents unique demands for which more customized data structures are required. Ex) (String) ← اوقات بدي indices مختلفة بهاي الحالة الـ indices تكون String not numeric
- Big data applications must support:
  - mixed data types, ← اوقات الداتا يلي بدي اياها مش (homogeneous) مش (numeric data)
  - customized indexing, A['name']
  - missing data, ⇒ يعني الـ array بقول انا تكون فاضية
  - data that's not structured consistently ⇒ (الداتا تكون مخربطة)
- Pandas is the most popular library for dealing with such data.

الـ map تكون مش مرتبة مش منظمة بدي (library) تقدر من هاي الداتا، افضل خيار لكم ايجاد

8

① one dim. array ② two dim array

Pandas Series and DataFrames

- Pandas uses Series for one-dimensional collections and DataFrames for two-dimensional collections.
- Creating a Series with Default Indices:

```
import pandas as pd

grades = pd.Series([87, 100, 94])
```

```
In [3]: grades
Out[3]:
index 0     87
index 1    100
index 2     94
dtype: int64
```


Pandas Series

- Creating a **Series** with Custom Indices

grades = pd.Series([87, 100, 94], index=['Wally', 'Eva', 'Sam'])

Ex) grades ['Eva'] = 100

- Dictionary Initializers:
- grades = pd.Series({'Wally': 87, 'Eva': 100, 'Sam': 94})

Pandas DataFrames

- A **DataFrame** is an enhanced two-dimensional array
- Creating DataFrames:

```
In [1]: import pandas as pd

In [2]: grades_dict = {'Wally': [87, 96, 70], 'Eva': [100, 87, 90],
   ...:                'Sam': [94, 77, 90], 'Katie': [100, 81, 82],
   ...:                'Bob': [83, 65, 85]}
   ...:

In [3]: grades = pd.DataFrame(grades_dict)

In [4]: grades
Out[4]:
   Wally  Eva  Sam  Katie  Bob
0     87  100   94    100   83
1     96   87   77     81   65
2     70   90   90     82   85
```


Pandas DataFrames

- Accessing column

```
In [6]: grades
Out[6]:
       Wally  Eva  Sam  Katie  Bob
Test1     87  100   94    100   83
Test2     96   87   77     81   65
Test3     70   90   90     82   85


In [7]: grades['Eva']
Out[7]:
Test1    100
Test2     87
Test3     90
Name: Eva, dtype: int64
```

Pandas Data frame

- Selecting Rows via the **loc** and **iloc** Attributes
- We can access a row by its label via the DataFrame's **loc** attribute
- We can access rows by integer zero-based indices using the **iloc** attribute

[10]: {-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

[ ]:

[ ]:

[11]:
```
numbers.add(20)
```

[23]:
```
numbers
```

[23]: {1, 2, 3, 4, 5, 15, 20}

[24]:
```
numbers.add(10)
```

[25]:
```
numbers
```

[25]: {1, 2, 3, 4, 5, 10, 15, 20}

[13]:
```
numbers
```

[13]: {1, 2, 3, 4, 5, 15, 20}

[9]:
```
# March , 5, NumPy Array

import numpy as np
```

[7]:
```
integers = np.array([[1, 2, 3], [4, 5, 6]])
```

[10]:
```
integers
```

لو تعديل ⇒ integers[0]

out: array([1,2,3])

integers[1]

out: array([4,5,6])

[10]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

⇒ أول سطر

⇒ ثاني سطر

[2D array]

integers[1][0] ⇒ ح يرجع الـ element الموجود من السطر الثاني في أول واحد

out: 4

[52]:
```
integers.dtype
```

⇒ نوع الداتا الموجودة داخل الـ array

[52]: dtype('int32')

[11]:
```
integers.ndim
```

لو تعديل ⇒ numbers = np.arange(1,8) (1,7)

→ Range(1 8-1)

out: array([1,2,3,4,5,6,7])

[11]: 2

[16]:
```
aa = np.array([[1, 2, 3], [4, 5, 6]])
```

[17]:
```
aa[0]
```

[17]: array([1, 2, 3])

[18]:
```
aa[1]
```

[18]:
```
array([4, 5, 6])
```

[49]:
```
floats = np.array([0.0, 0.1, 0.2, 0.3, 0.4])
floats
```

[49]:
```
array([0. , 0.1, 0.2, 0.3, 0.4])
```

[57]:
```
floats.ndim
```
⇒ float الـ dim ال

[57]:
```
1
```
⇒ one d array

[51]:
```
floats.dtype
```
← نوع الداتا الموجودة في array

[51]:
```
dtype('float64')
```

[ ]:

[12]:
```
aa = np.array([[1, 2, 3,50,30], [4, 5, 6,-8,8],[7, 5, 30,10,8]])
```
⇒ 5 اعمدة / 3 اسطر

[54]:
```
aa
```

[54]:
```
array([[ 1,  2,  3, 50, 30],
       [ 4,  5,  6, -8,  8],
       [ 7,  5, 30, 10,  8]])
```

[13]:
```
aa.ndim
```

[13]:
```
2
```

[14]:
```
aa.shape
```
⇒ (بطلع كم عدد الاسطر والاعمدة)

[14]:
```
(3, 5)
```
(الاعمدة) rows ← ← colums (الاسطر)

[58]:
```
aa.size
```
⇒ كم element فيها ⇒ 3x5=15

[58]:
```
15
```

[22]:
```
y = np.zeros((2, 3, 4))
y
```

[22]:
```
array([[[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]],

       [[0., 0., 0., 0.],
```

```
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]]])
```

```
[23]: aa
```

```
[23]: array([[ 1,  2,  3, 50, 30],
             [ 4,  5,  6, -8,  8],
             [ 7,  5, 30, 10,  8]])
```

```
[25]: for i in aa:
          for j in i:
              print(j, end=' ')
          print()
```

row

Colums

رح تلف على الاسطر

rows

Colums "elemnt"

رح تلف على العامود داخل كل سطر

لما اخلص طباعة كل ال rows المطلوبة print

```
1 2 3 50 30
4 5 6 -8 8
7 5 30 10 8
```

```
[4]: y
```

```
[4]: array([[[0., 0., 0., 0.],
             [0., 0., 0., 0.],
             [0., 0., 0., 0.]],

            [[0., 0., 0., 0.],
             [0., 0., 0., 0.],
             [0., 0., 0., 0.]]])
```

```
[ ]:
```

```
[60]: np.zeros(5)
```

⇒ رح ترجع numpy array مكونة من element(5) كلهم zero

```
[60]: array([0., 0., 0., 0., 0.])
```

```
[61]: np.ones(30)
```

30 element فيها ones كلها ال array

```
[61]: array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
             1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```
[63]: np.full((3, 5), 8)
```

⇒ بدي (3) السطر (5) الاعمدة

كلهم ب 8

```
[63]: array([[8, 8, 8, 8, 8],
             [8, 8, 8, 8, 8],
             [8, 8, 8, 8, 8]])
```

```
[26]: np.arange(5,20)
```

20-1=19

تعديل ⇒ np.arange(5)

Rang 0-4

out: array([0,1,2,3,4])

```
[26]: array([ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

```
[67]: np.arange(10, 1, -2)
```

⇒ تبدأ من (10) بكل مرة (-2)

تتوقف عند (2) لأنه ما بزبط تنزل لل (1)

```
[67]: array([10,  8,  6,  4,  2])
```

21-1 = 20 م

```
[28]: np.arange(1, 21).reshape(4, 5)
```

⇒ تعطيني ارقام من 1 لل [21-1=20] وترتيبهم ب (4) أسطر و (5) أعمدة

```
[28]: array([[ 1,  2,  3,  4,  5],
             [ 6,  7,  8,  9, 10],
             [11, 12, 13, 14, 15],
             [16, 17, 18, 19, 20]])
```

⚠ لو حطينا ال 23 بدل ال (4) اسطر و (5) اعمدة

تعطي error ! لانه ما في تناسق بين الاسطر، الاعمدة، وال element

(✓) 4×5 = 20 اما 4×5 = 21 (X)

```
[71]: np.arange(1, 100001).reshape(4, 25000)
```

بدل: تكون ال size تبع ال array مناسب مع reshape

```
[71]: array([[     1,      2,      3, ...,  24998,  24999,  25000],
             [ 25001,  25002,  25003, ...,  49998,  49999,  50000],
             [ 50001,  50002,  50003, ...,  74998,  74999,  75000],
             [ 75001,  75002,  75003, ...,  99998,  99999, 100000]])
```

```
[31]: Grades = np.array([[87, 96, 70], [100, 87, 90],
      [94, 77, 90], [100, 81, 82]])
```

```
[32]: Grades
```

```
[32]: array([[ 87,  96,  70],
             [100,  87,  90],
             [ 94,  77,  90],
             [100,  81,  82]])
```

```
[33]: Grades.sum()
```

كذلك ⇒ Grades.max()

out: 100

Grades.min()

out: 70

Grades.mean() → avg

87.83333

```
[33]: 1054
```

```
[34]: Grades2= Grades-5
```

```
[40]: Grades
```

لو كتبنا ⇒ Grades2 ⇒ out: نقص (5) من كل element

[82, 91, 65]
[95, .., ..]
[.., .., ..]
[.., .., ..]

```
[40]: array([[ 87,  96,  70],
             [100,  87,  90],
             [ 94,  77,  90],
             [100,  81,  82]])
```

87, 96, 70
100, 87, 90

يعرض اول سطرين

2-1=1 م

بدي من كل السطر، العامود من ال 3-1=2

نغير ال Column الثاني والثالث

```
[ ]: Grades[0, 1], Grades[0:2], Grades[[1, 3]], Grades[:, 0], Grades[:, 1:3]
```

out: 96 "اول سطر، اول اشي"

ال (0) يعني بديا السطر (1) العامود (2)

اما ال (:) بديا من ← الى

بدي كل ال rows بس نفوت اول Colums

[87, 100, 94, 100]

(زي كانه العامود الاول)

لو [0 :]

out: [87, 96, 70]

(زي كانه السطر الاول)

```
[41]: Grades
      Grades[:,0]
```

```
[41]: array([ 87, 100,  94, 100])
```

```
[42]: Grades[:, 1:3]
```

```
[42]: array([[96, 70],
             [87, 90],
             [77, 90],
             [81, 82]])
```

```
[90]: Grades[0:3]
```

```
[90]: array([[ 87,  96,  70],
             [100,  87,  90],
             [ 94,  77,  90]])
```

```
[89]: Grades[[1,3]]
```

```
[89]: array([[100,  87,  90],
             [100,  81,  82]])
```

```
[91]: #elements of the first column
      Grades[:, 0]
```

```
[91]: array([ 87, 100,  94, 100])
```

```
[93]: #elements of the first row
      Grades[0,:]
```

```
[93]: array([87, 96, 70])
```

```
[94]: Grades[:, 1:3]
```

```
[94]: array([[96, 70],
             [87, 90],
             [77, 90],
             [81, 82]])
```

```
[95]: Grades[:,0:2]
```

out ⇒ the first two Colum

```
[95]: array([[ 87,  96],
             [100,  87],
             [ 94,  77],
             [100,  81]])
```

```
[43]: numbers = np.arange(1, 8)
```

```
[44]: numbers
```

[44]: array([1, 2, 3, 4, 5, 6, 7])

```
[45]: #numbers2 is a views for array numbers
      numbers2 = numbers
```

⇒ Copy (numbers2 رح توخذ نفس قيم الـ number)
يسمى (Shallow views)

```
[46]: numbers2
```

[46]: array([1, 2, 3, 4, 5, 6, 7])

التعديل على القديم رح ينعكس عليها
يعني زدت على number (+10)
الـ (+10) برضه رح يروح للـ numbers2

```
[47]: numbers +=10
```

```
[48]: numbers
```

لذلك يسمى "Shallow copy" أي تعديل على numbers رح يتحول للـ numbers2

[48]: array([11, 12, 13, 14, 15, 16, 17])

← أوقات بدي اعمل نسخة (copy object) وعشان احفظ الـ object القديم ، والـ object الجديد المعدل عليه زي ما بدي بدون ما يأثر على الأصلي للـ object من خلال (deepcopy)

```
[49]: numbers2
```

[49]: array([11, 12, 13, 14, 15, 16, 17])

```
[96]: #deep copy
      numbers = np.arange(1, 20)
      numbers2 = numbers.copy()
```

⇒ بيعطي نسخة مفصولة عن numbers

```
[97]: numbers
```

[97]: array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19])

```
[99]: numbers += 20
      numbers
```

[99]: array([41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,
       58, 59])

```
[100]: numbers2
```

⇒ [ما تأثر مع التغير تبع الـ number]

[100]: array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19])

```
[ ]: numbers2.des
```

```
[64]: import pandas as pd
```

```
[65]: grades = pd.Series([87, 100, 94])
```

```
[66]: grades
```

```
[66]: 0      87
      1     100
      2      94
      dtype: int64
```

```
[67]: grades = pd.Series([87, 100, 94], index=['Wally', 'Eva', 'Sam'])
```

⑴

```
[68]: grades
```

```
[68]: Wally      87
      Eva       100
      Sam        94
      dtype: int64
```

لو(تعديل) ⇒ grades['Eva'] ←
out: 100

طريقتين لانشاء Series
⑴ , ⑵

```
[70]: grades2 = pd.Series({'Wally': 87, 'Eva': 100, 'Sam': 94})
      grades2
```

⑵

```
[70]: Wally      87
      Eva       100
      Sam        94
      dtype: int64
```

```
[71]: grades2[1]
```

OR grades2['Eva']

```
[71]: 100
```

```
[18]: grades
```

```
[18]: Wally      87
      Eva       100
      Sam        94
      dtype: int64
```

```
[21]: grades[2]
```

```
[21]: 94
```

```
[78]: grades = pd.Series([87, 100, 94])

      grades = grades.append(pd.Series([91]))

      grades = grades.reset_index(drop=True)
```

هو بتعيد ترتيب ال index) 6 ما بين two value الهم
Same
index

```
C:\Users\user\AppData\Local\Temp\ipykernel_17352\3702451305.py:3: FutureWarning:
The series.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  grades = grades.append(pd.Series([91]))
```

```
[79]: grades
```

```
[79]: 0     87
      1    100
      2     94
      3     91
      dtype: int64
```

```
[ ]:
```

```
[73]: grades.min()
```

```
[73]: 87
```

```
[15]: grades.describe()
```

بحدد القيم الموجودة في Series

```
[15]: count      3.000000
      mean      93.666667
      std        6.506407
      min       87.000000
      25%       90.500000
      50%       94.000000
      75%       97.000000
      max      100.000000
      dtype: float64
```

Avg (mean)

```
[ ]:
```

```
[75]: pd.Series(12, range(10))
```

انشاء Series ⇐ بدي اخزن 12 في كل ال cell

كل ال indexes تبدأ من 0 لـ 9

```
[75]: 0    12
      1    12
      2    12
      3    12
      4    12
      5    12
      6    12
      7    12
      8    12
      9    12
      dtype: int64
```

```
[16]: grades = pd.Series([87, 100, 94], index=['Wally', 'Eva', 'Sam'])
```

```
[17]: grades
```

```
[17]: Wally     87
      Eva      100
      Sam       94
      dtype: int64
```

```
[19]: grades['Eva']
```

```
[19]: 100
```

```
[43]: grades = pd.Series({'Wally': 87, 'Eva': 100, 'Sam': 94})
```

```
[44]: grades
```

```
[44]: Wally     87
      Eva      100
      Sam       94
      dtype: int64
```

```
[46]: import pandas as pd
      grades_dict = {'Wally': [87, 96, 70], 'Eva': [100, 87, 90],
      'Sam': [94, 77, 90], 'Katie': [100, 81, 82],
      'Bob': [83, 65, 85]}
```

[كل Key مقابله list]

```
[51]: grades_dict
```

```
[51]: {'Wally': [87, 96, 70],
       'Eva': [100, 87, 90],
       'Sam': [94, 77, 90],
       'Katie': [100, 81, 82],
       'Bob': [83, 65, 85]}
```

دالة ال (data frame) نقدر نحول ال (dict) لل (data frame)

```
[52]: grades = pd.DataFrame(grades_dict)
      grades
```

Fun. — اسم ال dict

```
[52]:    Wally  Eva  Sam  Katie  Bob
      0     87  100   94    100   83
      1     96   87   77     81   65
      2     70   90   90     82   85
```

حولت كل Key لل Colum

كل row الها index

(اسم ال dataframe) ← اسم Colum

```
[58]: grades['Wally']
```

⇒ (Colum معين) لل Access اعمل

```
[58]: 0    87
      1    96
      2    70
      Name: Wally, dtype: int64
```

```
[57]: grades.iloc[1]
```

⇒ (row) اقرأ

← يجيب السطر ال [1 = index]

```
[57]: Wally     96
      Eva       87
      Sam       77
      Katie     81
      Bob       65
      Name: 1, dtype: int64
```

m(string)

```
[87]: grades2 = pd.DataFrame(grades_dict, index=['Test1', 'Test2', 'Test3'])
      grades2
```

(data frame) اسمها

```
[87]:        Wally  Eva  Sam  Katie  Bob
      Test1     87  100   94    100   83
      Test2     96   87   77     81   65
      Test3     70   90   90     82   85
```

```
[88]: grades2['Wally']
```

```
[88]: Test1     87
      Test2     96
      Test3     70
      Name: Wally, dtype: int64
```

اذا بدي معلومات معين Column (اسم العامود)

اذا بدي معلومات السطر الاول ⇒ grades2.iloc[0] ⇒ ما اشتغلت صح زي قبل

out: 87

∴ مش عن طريقها (iloc) بحل عن طريق loc[Test1]

out: Wally 87
Eva 100
Sam 94
Katie 100
Bob 83

```
[ ]:
```

```
[62]: grades.index = ['r1', 'r2', 'r3']
```

```
[63]: grades
```

```
[63]:     Wally  Eva  Sam  Katie  Bob
      r1     87  100   94    100   83
      r2     96   87   77     81   65
      r3     70   90   90     82   85
```

```
[29]: grades['Eva']
```

```
[29]: 0    100
      1     87
      2     90
      Name: Eva, dtype: int64
```

```
[30]: grades.Eva
```

```
[30]: 0    100
      1     87
      2     90
      Name: Eva, dtype: int64
```

[89]:
```
grades = pd.DataFrame(grades_dict)
grades
```

[89]:

	Wally	Eva	Sam	Katie	Bob
0	87	100	94	100	83
1	96	87	77	81	65
2	70	90	90	82	85

[90]:
```
grades.index = ['Test1', 'Test2', 'Test3']
```
⇒ نقدر انعدل مسار ال index بعد انشاءه

" نقدر انعدل ال label الخاصة في row بعد انشاء ال grades "

[91]:
```
grades
```

[91]:

	Wally	Eva	Sam	Katie	Bob
Test1	87	100	94	100	83
Test2	96	87	77	81	65
Test3	70	90	90	82	85

[92]:
```
grades.loc['Test1']
```
⇒ بعطي row معلومات

[92]:
```
Wally     87
Eva      100
Sam       94
Katie    100
Bob       83
Name: Test1, dtype: int64
```

[93]:
```
grades.iloc[1]
```

[93]:
```
Wally    96
Eva      87
Sam      77
Katie    81
Bob      65
Name: Test2, dtype: int64
```

[102]:
```
grades.iloc[0:3]
```

[102]:

	Wally	Eva	Sam	Katie	Bob
Test1	87	100	94	100	83
Test2	96	87	77	81	65
Test3	70	90	90	82	85

[39]:
```
grades['Eva']
```

تعديل grades.iloc [1:2]

رقم 2-1=1

← بطلع record واحد : out

[1:3]

← بطلع two record

[0:3]

← بطلعهم كلهم

[39]:
```
Test1    100
Test2     87
Test3     90
```

```
Name: Eva, dtype: int64
```

```
[ ]:
```

```
[41]: grades.iloc[0:2]
```

⇒ اول سطرين

[41]:

	Wally	Eva	Sam	Katie	Bob
Test1	87	100	94	100	83
Test2	96	87	77	81	65

```
[107]: grades.loc[['Test1', 'Test3']]
```

معلومات Test1 و Test3

انتبه الـ (,) بتجيب من هون

[107]:

	Wally	Eva	Sam	Katie	Bob
Test1	87	100	94	100	83
Test3	70	90	90	82	85

```
[ ]:
```

Hashemite University
Computer Engineering
Data Science Course

Introduction to Machine Learning

Dr. Mohammad Al-hammouri

1

2

Outline

- What Is Machine Learning?
- Supervised learning
- Unsupervised learning
- Semi supervised learning
- Confusion matrix

What Is Machine Learning?

- Machine Learning is the science (and art) of programming computers so they can learn from data.

ال Computer لوحده انسان ما بفكر، هو فقط انت تتبعت Input وهو يطلع output
بدنا نحاول انه ال Computer يكون اله قرار لشيء معين، يتعلم ويعطي قرار

Here is a slightly more general definition:
[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.
—Arthur Samuel, 1959

Ex لعني انت حاكيله اذا اجاك A التصنيف بكون A class
اذا اجاك B التصنيف بكون Class 2
اذا اجاك A+B التصنيف بكون Class c
[اما اذا اجاك اشي لا ال Computer مارح يعرف يعمل اشي]

الخطية [Exper.] وخبرته على task معين من خلال ال [T+E] يلي ادرب عليها

And a more engineering-oriented one:
A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

يعني اذا اجا اشي جديد غير A/B مارح يقرر

ال Comp. يوصل لل performance الي اذا performance تحسنت

كل من الخطوة الاولى [I'am explicitly programming] يعني فعليا انا حكيلو اذا اجاك كذا اعمل كذا

ليس [Explicitly programmed] ما بقدر يقرر كحال ال Computer اذا اجاه اشي جديد لانه انت مش مبرمجه الاول A و B

احنا بدنا ال Computer يكون [Explicitly programmed] بدنا كاله يقرر اذا اجاه اشي جديد من يلي تعلمه لقدر يقرر

ال machine learning ال Computer ما بكون [Explicitly programmed] هو قادر اذا اجاه [new case] بقدر يقرر، حتى لو انت مش حاكيلو هيا
[احلى ال Computer هيا انه يفكر]

مصنالة ال Computer قادر تعلم "ما بعض" على نفس المحتوى "يلي كان عليه"
ال Exp. تزيد وال pref. تزيد
"كل ما زادت" ال data set وكانت صحيحة كل ما زاد ال perf. "زاد"

The Data Science Cycle

[اهم نقطة في ال cycle]

[1] تحضير الداتا

1. **Capture:** Data Acquisition, Data Entry, Data Extraction. *This stage involves gathering raw data.*

[2] اجهز الداتا "اعمل" "Clen" ; [اي string attribute] حل تحويله لـ م

2. **Prepare:** Data Preparation, Data Cleaning and Data Pre-processing. *This stage taking the raw data and putting it in a form that can be used.*

لحفظ المشاكل انه في داتا فاضية

ال machine learning ما بتقبل string بدي number

[[3] الاجراءات ومكن استفيد من الداتا اسئلة study يلي لازم ادرسها على الداتا]

3. **Process:** Data mining, Clustering/Classification, Data Modeling. Data scientists take the prepared data and examine its patterns, ranges, and biases to determine how useful it will be in predictive analysis.

اذا مشكلة التوقع للسعر [Regression Analysis]

اذا المشكلة بدها توقع ; [4] تحديد ال Analysis

4. **Analyze:** Exploratory, Predictive Analysis, Regression, Text Mining, Qualitative Analysis. *Here is the main step of the lifecycle.*
5. **Communicate:** Data Reporting, Decision Making.

[5] يعني انت بنيت ال model تاعك وال AI model، بنجير التطبيق ال model في ارض الواقع
بحيث تعمل testing بتفت داتا جديدة وتشوف اذا اشتغل ال model تاعك صح اولا

Machine Learning Example?

* اوقات لبعض ال email بتحول ال spam وكانه بكون في [model AI] موجود ف email ف كاله اذا شك فيه Email
انه منه مصدر غير موثوق بيوديه لل spam

- Your spam filter is a Machine Learning program that can learn to flag spam given examples of spam emails (e.g., flagged by users) and examples of regular (non-spam, also called "ham") emails

← الصنف ال [email] لكاله متر كاشات تشوفه انا هو [spam] او [not

- The examples that the system uses to learn are called the **training set.**

كمال ال Dataset بلي اندرب عليها ال model التمي [traning set] اما بلي بنحل عليها test ال model التمي [testing set]

- In this case, the task **T** is to flag spam for new emails, the experience **E** is the training data, and the performance measure **P** needs to be defined; for example, you can use the ratio of correctly classified emails.

← بلي بدرب عليه ال model [Expr.]

لتحدد اذا بتطلع صح او لا

5

Ex) Data = 100k → 70% training ⇒ بدرب ال model عليه وبكون فعتا، الخوارزميات
→ 30% testing
بعد ما تكون بنعمل ال test على 30% بتشوف هل تدريبك بلي انت دربته كويس او لا

← بالنهاية انت بتحسب ال [accuracy] كل ما كانت عالية كلما ال model اشتغال بشكل صحيح

بتحسب ال accuracy عل
99% = Ex ⇐ training
30% = Ex ⇐ testing
⇐ ال model بيس ليس overfit model

learning :-

(1) [Supervised learning] ⇒ يعني الداتا بلي بتشتغل عليها Data [is labeled] موجود في كل record
[each record is labeled with the target value]

Supervised learning: Algorithms which learn from a training set of labeled examples to generalize to the set of all possible inputs.

(1) تصنيف الداتا لل (two class) (1) or more two class (2)

- A typical supervised learning task is **classification**. The spam filter is a good example of this: it is trained with many example emails along with their class (spam or ham), and it must learn how to classify new emails

التوقع سعر البيت ليس classification ضد Regression

- Another typical task is to $\boxed{\text{predict}}$ a target numeric value, such as the price of a car, given a set of features (mileage, age, brand, etc.) called predictors. This sort of task is called (2) **regression**. To train the system, you need to give it many examples of cars, including both their predictors and their labels (i.e., their prices).

6

هو بجيك [new instance] وبتصنفها

Supervised learning (classification)

[كل record ليها label]

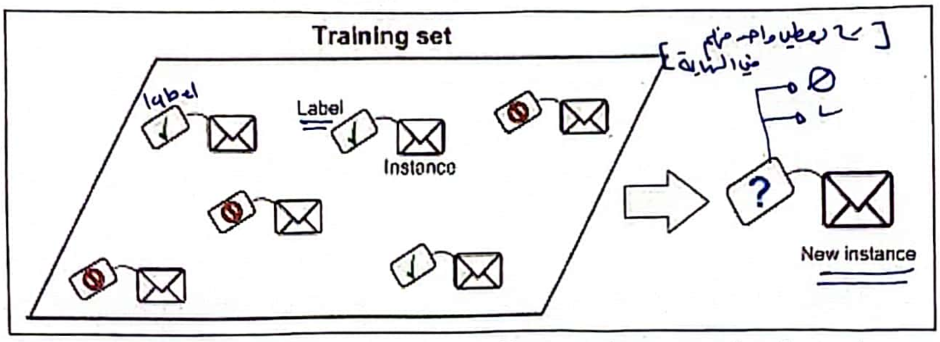


Figure 1-5. A labeled training set for supervised learning (e.g., spam classification)

هو بتوخذ كل الـ [feature] وبترسم علاقتها مع [target value]

Supervised learning (Regression)

هو بترسم العلاقة بين الـ x والـ y لعدين لتشوف القيمة يلي بدها اياها وبتعمل اسقاط على المنحنى وبتشوف الـ value

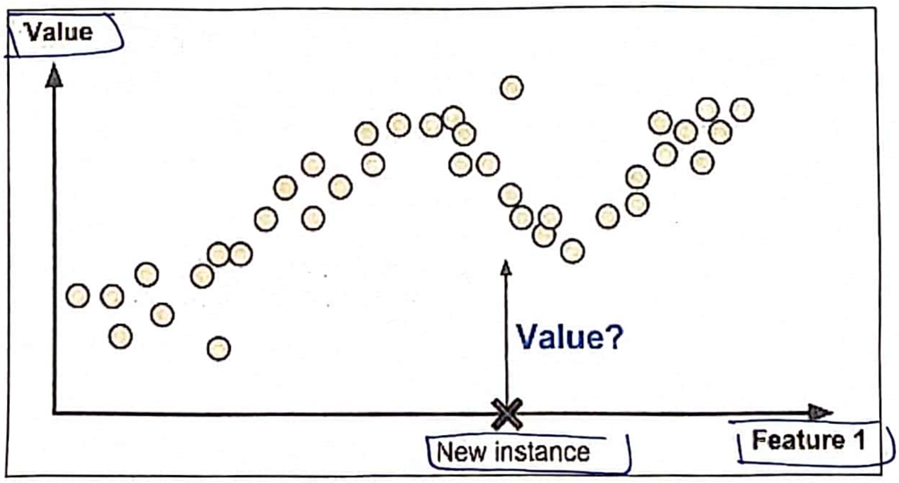


Figure 1-6. Regression

Supervised Learning

Here are some of the most important supervised learning algorithms:

-Nearest NeighborsLinear Regression

1 • Logistic Regression
2 • Naive Bayes
3 • Support Vector Machines (SVMs)
4 • Decision Trees and Random Forests
5 • Neural networks

لتتشغل على مستوى اعلى

[learning more complicated]

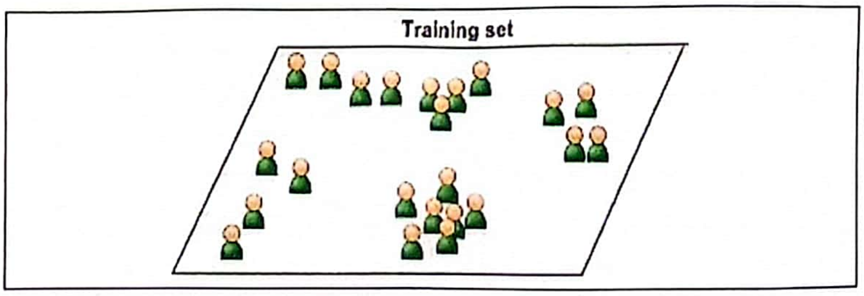
Accuracy تكون اقل

اذا الـ label مش موجود

2 [Unsupervised learning] [Data unlabeled]

Algorithms that learn from a training set of unlabeled examples. Used to explore data according to some statistical, geometric or similarity criterion. (The system tries to learn without a teacher) [Data عشوائية فيها feature فقط]

بتشوف كل مجموعة قريبة من بعض بتعمل [Clustering]



الـ model في un-super بيدرس الداتا ويجمع كل مجموعة من الداتا في [Clustering] بحيث في مرحلة الـ testing لما نعطيه (new data) ما بيعمل (classification) بيوزعها حسب الـ [Clustering] المناسب إليها

Figure 1-7. An unlabeled training set for unsupervised learning

شي مفيد لأنه الداتا مش دائمًا فيها label وانت بدل لتتفيرها من خلال الـ [Clustering]

[Unsupervised Learning]

Here are some of the most important Unsupervised learning algorithms:

1) • Clustering
  —[K-Means]

2) • (Anomaly) detection and novelty detection
الي فيهم مشكلة
  —One-class SVM
  —Isolation Forest

3) • Visualization and dimensionality reduction
  —Principal Component Analysis (PCA) ⇒ نجزئ ال clustering ، بدرس ال feature

بجمع كل مجموعة ، بالي الها علاقة مع بعض
بدمجهم في feature موحدة

لو احنا مش كل ال [feature] محتاجين ، عشان نضيع وقت ال model
عليهم ، فـ بدمج مجموعة مع بعض

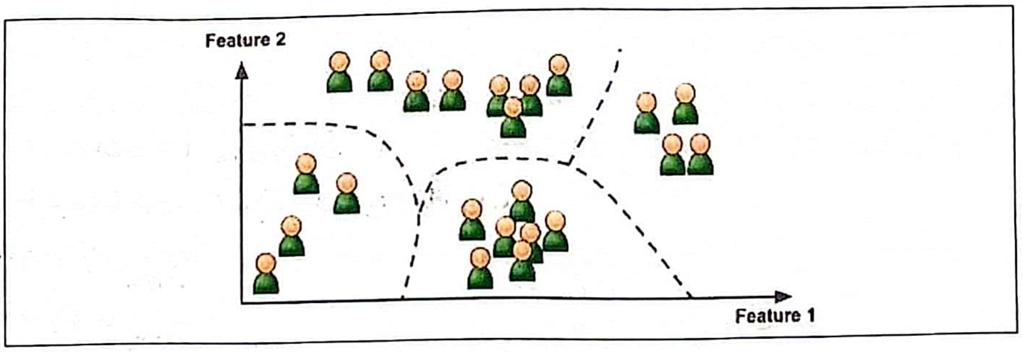
Ex) 100 data set — (1) ⇒ 100 بدل ما نشتغل على ⇒ ادمج (1) بسوا من 100 ⇒ (1) يعني بدل (100) اشتغل على (1)
(10) ⇒ ادمج (10) بسوا
⋮

هدف ال PCA : نقلل عدد ال [feature] ونقلل ال training time

Unsu. لأنه ما بتعتمد على (target value) "label data" هي لتشوف ال feature

Ex) تصنيف الزوار على الموقع تبعك

Clustering



• Let say you have a lot of data about your blog's visitors.
• You may want to run a clustering algorithm to try to detect groups of similar visitors.
• For example, it might notice that 40% of your visitors are males who love comic books and generally read your blog in the evening

Unsuper. احد تطبيقات ال

Visualization And [Dimensionality Reduction] PCA

- The goal is to simplify the data without losing too much information.

بتشتغل انه بشوف بين ال feature "العلاقة"

- One way to do this is to merge several [correlated features] into one

بحل [overfit] يعني ال model بعمل على ال testing بعكس ال training "بعطيني result عالية"
انا لما اعطيه عالـ [new dataset] بتقل ال accuracy

- For example, a car's mileage may be very correlated with its age, so the dimensionality reduction algorithm will merge them into one feature that

- This is called feature extraction.

لما اشيل ال (Correlated feature) ال model بشتغل بشكل افضل

[Anomaly detection]

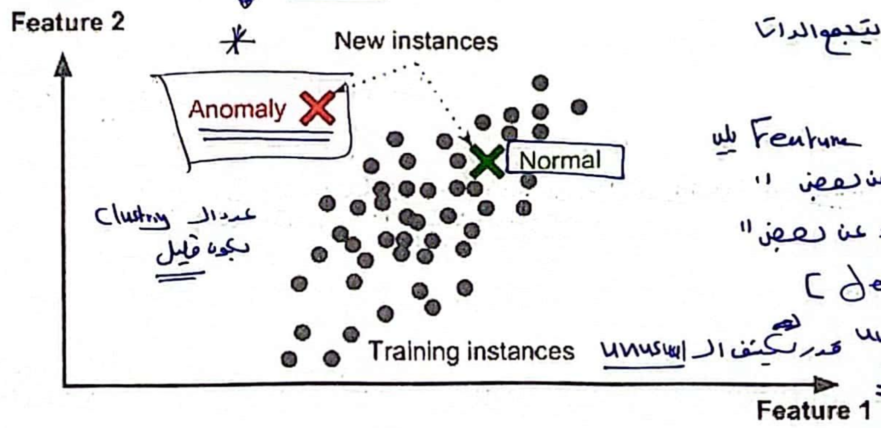
- Unsupervised task used for detection the unusual (anomaly) data

- For example, detecting unusual credit card transactions to prevent fraud, catching manufacturing defects

- The system is shown mostly normal instances during training, so it learns to recognize them and when it sees a new instance it can tell whether it looks normal or not.

Anomaly detection

"احتمال ١" fraud انه تكون most probably



Clustering عدد ال بكون قليل

unusual بقدر يكتشف ال

ال model لما يتجمعوا الداتا بفرزلك اياها

← ال "normal" يلي Feature "قراب من بعض"

← Anomaly "بعاد عن بعض"

بعطلك [detect]

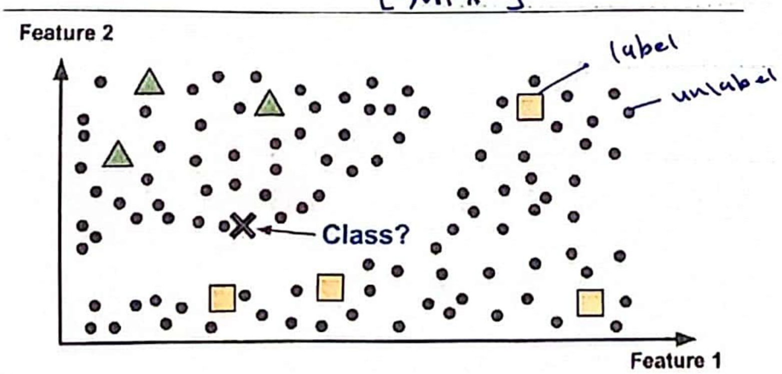
هاد unsup. task

[Semi supervised learning]

[a lot of unlabel data and little label data] لما عادة بكون

- Some algorithms can deal with partially labeled training data, usually a lot of unlabeled data and a little bit of labeled data.
- This is called semi supervised learning

[Mix]



tips

March 14, 2024

اول اشي import عشان احمل ال csv file عليه panda dataframe

```
[14]: import pandas as pd  1
      import numpy as np  2
      import matplotlib.pyplot as plt  3

      import seaborn as sns  4
```

عشان تخزنهم [as two dim. array]

* كل library بنعملها اختصار عشان اقدر اتعامل معها بشكل اسهل.

الهدف انه اعرف واتعرف كيف اعمل upload لل [csv file] فيه داتا على ال paython وكيف اعمل [visualize data] الموجودة في File يعني اني ارسم الداتا "as a graph"

```
[15]: df= pd.read_csv('tips.csv')
```

اسم بخزنه فيه dataframe

ما في داعي اكتبنا extension لما قبله، لانه موجود بال Home لـ ليس لحط المسار ال file نفس ال (folder) يلي اشغل عليه

```
[16]: df
```

[16]:

	total_bill	tip	sex	smoker	day	time	size
0 اول record	16.99	1.01	Female	No	Sun	Dinner	2
1 ثاني record	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
..	…	…	…	…	…	…	…
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

[244 rows x 7 columns]

⇒ df ⇒ عبارة عن 244 record [0 → 243] و 7 "column" او بنقول لها Features الموجودين عندي

شرح ال Feature:
* هذه الداتا بتحتوي معلومات عن مطعم بتعرضلي ال tip المدفوع في transaction عندي ⇐ 244 [transaction] tip "البقشيش"

احنا بدنا نعمل (visualize) لل data "نرسمها" ازا ليش لما نرسم الداتا بنعملها تصور اكثر من الارقام

```
[5]: df.describe()
```

عدد ال record الموجود في dataframe = 244

[5]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
AVG mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

[17]:
```
a=df.columns
```

[18]:
```
a
```

[18]:
```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'],
      dtype='object')
```

[7]:
```
df.info()
```
] صفة الـ Comm. ⇒ تبحثلي اذا في لبعض الـ Features فيه Null Value
وايش نوع كل Features

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    object
 3   smoker      244 non-null    object
 4   day         244 non-null    object
 5   time        244 non-null    object
 6   size        244 non-null    int64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

# of record "total" = 244
total bill Non null = 244] ⇒ No Null Value في total bill يعني اصلا attribute

object ⇒ means "String"

معناه انه [numeric feature مش string]
float int

لو فيه size كان فيه [230]
لقيت انه الـ size attribute فيه 230 [Not null] و 14 [Null] ح اتعامل معهم بطريقة مختلفة

[19]:
```
df.tail()
```
⇒ [[5 record] بطلع أول] ≠ head
[آخر كم record]

[19]:

	total_bill	tip	sex	smoker	day	time	size
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

آخر (5) record

[9]:
```
df.isnull().sum()
```
⇒ [اذا فيه عندك null value او لا وبتعطيك عددهم]

[9]:
```
total_bill    0
tip           0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
```
⇒ لأنه مافيه اي attribute فيها null فـ zero

[10]:
```
df.nunique()
```

لتفيد منها [string attribute] كم قيمة
منها Category لما تكون String

```
[10]: total_bill    229
      tip           123
      sex             2
      smoker          2
      day             4
      time            2
      size            6
      dtype: int64
```

Sex اما من ← sex

الطالب (2) يعني ← smoker

two يبنشي Category

"male/famale"

day ← بدرس الـ(4) ايام

Numeric Category الـ ماح تفيدني ؟ كل رقم number كالعطلين ارقام عالية

```
[8]: df['time'].unique()
```

df['day'] unique()

```
[8]: array(['Dinner', 'Lunch'], dtype=object)
```

→ out array(['femal', 'Male'])

```
[9]: df['day'].unique()
```

ده من (string Feature) ولازم اعرف الـ Category الموجودين في الـ Feature

```
[9]: array(['Sun', 'Sat', 'Thur', 'Fri'], dtype=object)
```

```
[21]: df['size'].unique()
```

```
[21]: array([2, 3, 4, 1, 6, 5], dtype=int64)
```

⇒ max = 6 [عدد الاشخاص على الطاولة]
min = 1

```
[57]: df.describe()
```

[57]:

	total_bill	tip	size	tip_pct
count	244.000000	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672	0.202123
std	8.902412	1.383638	0.951100	0.163385
min	3.070000	1.000000	1.000000	0.036955
25%	13.347500	2.000000	2.000000	0.148274
50%	17.795000	2.900000	2.000000	0.183110
75%	24.127500	3.562500	3.000000	0.236821
max	50.810000	10.000000	6.000000	2.452381

للـ Feature يلي اسمها [day] للـ [dataframe] يلي اسمها df

ارسم Count plot — الاسم — data frame

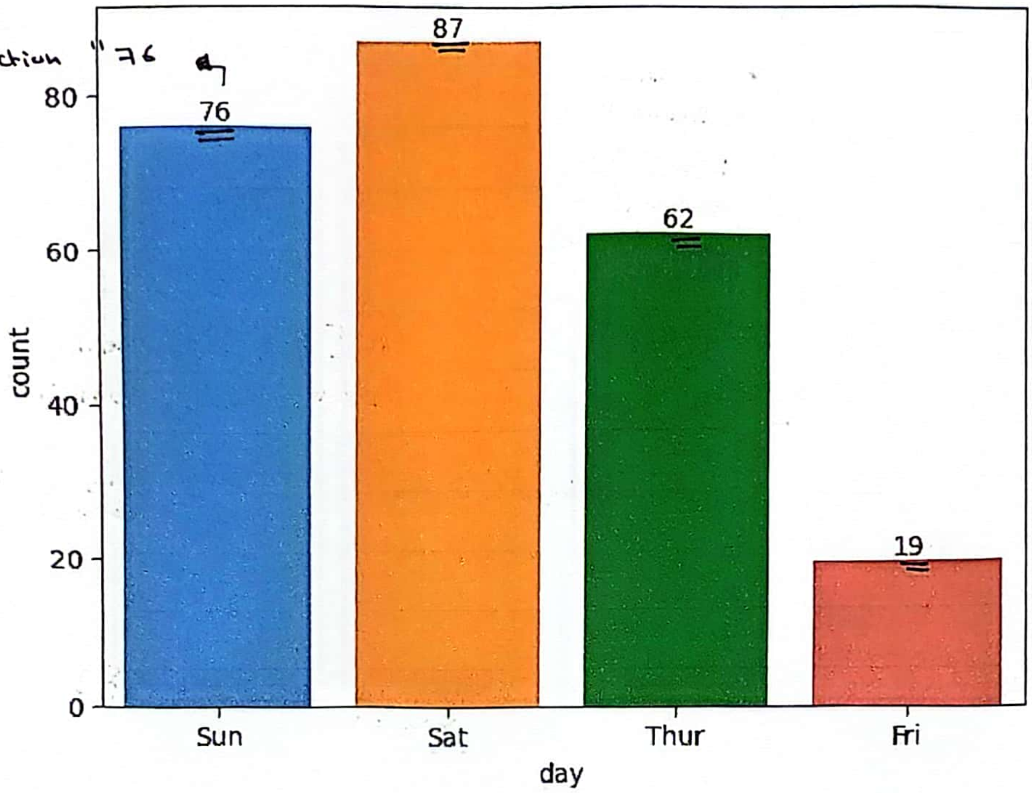
```
[22]: ax=sns.countplot(x='day', data=df)
      ax.bar_label(ax.containers[0])
```

بدي ارسم الـ Feature يلي اسمها day "استخدم لكل يوم" من الـ لـ ايام ، كم عدد الـ [transaction] من كل يوم

[عشان يحط الرقم على كل bar]

```
[22]: [Text(0, 0, '76'), Text(0, 0, '87'), Text(0, 0, '62'), Text(0, 0, '19')]
```

"transaction" 76



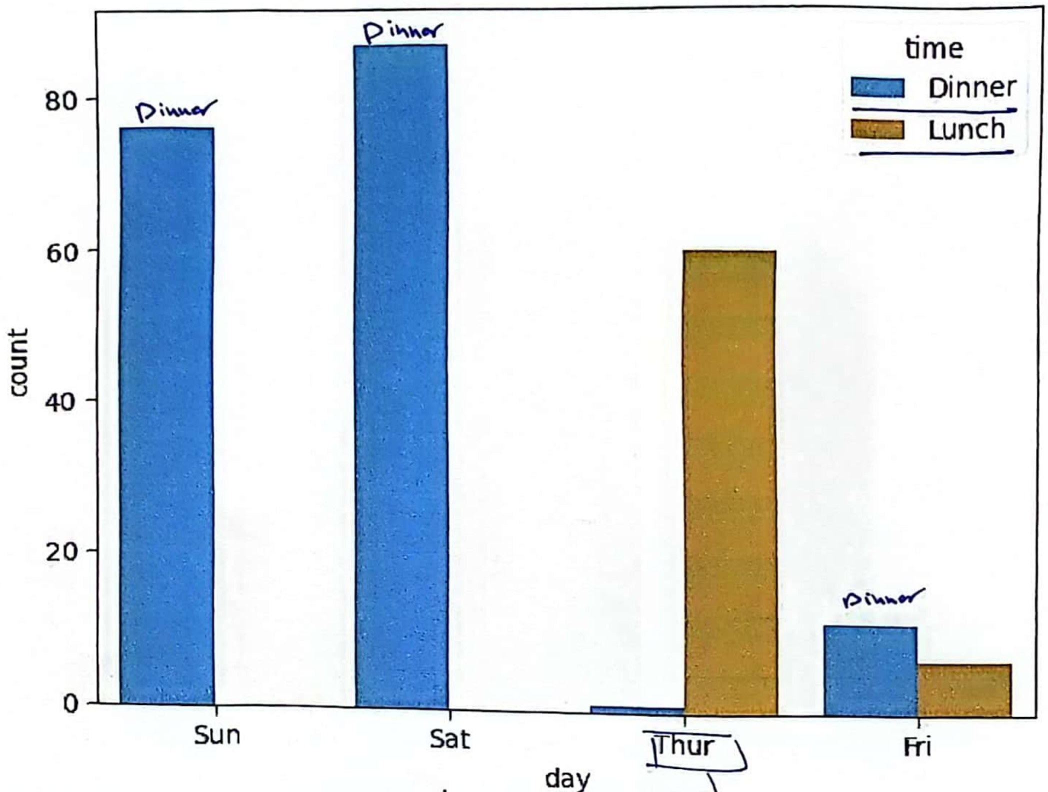
[بتستفيد من الرسمة كقراءة اولية ، انه اغلب الناس يلي بزوروا المطعم ، بين Weekend "Sun , Sat"
بالتالي صاحب المطعم رح يهتم في نهاية الاسبوع اكثر من الايام الثانية]

```
[23]: sns.countplot(x='day', data=df, hue='time')
```

كم من خلالنا ارسم [two feature] ، الاول الـ day والثاني الـ time

```
[23]: <AxesSubplot:xlabel='day', ylabel='count'>
```

لو بدي اشوف هاي الارقام مثلا Sun مين اجا اكثر ، على وقت الغداء او مش وقت الغداء



لو لاحظنا صادمين في على الأغلب مارح حدا يجي على العشاء بما أنه تاني يوم دوام

اما بيجوا على lunch ا أكثر 6 الناس الموجودين من العمل ع يومهم

:. هاي القراءة بتعطيني انطباع عن [transaction]

```
[48]: sns.countplot(x='day', data=df, hue='smoker')
```

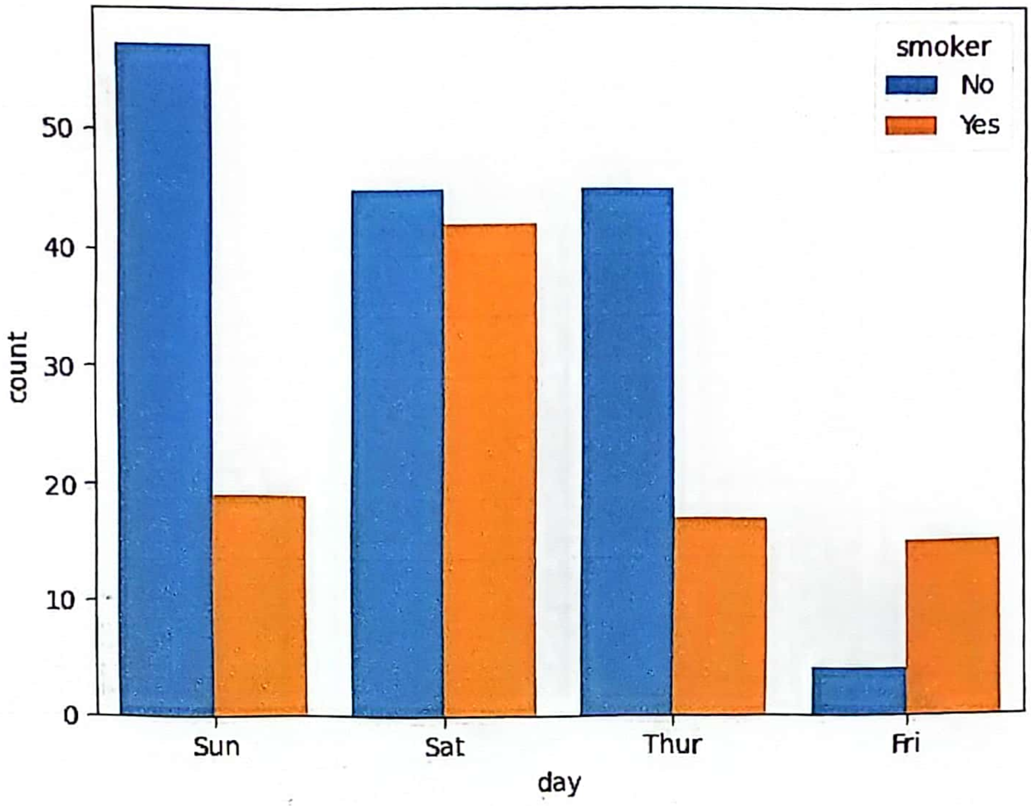
لو السؤال هنا برسم الـ [Feature] الثانية مع [Feature] الأول

```
[48]: <AxesSubplot:xlabel='day', ylabel='count'>
```

"العنصر الأساسي عندي "day

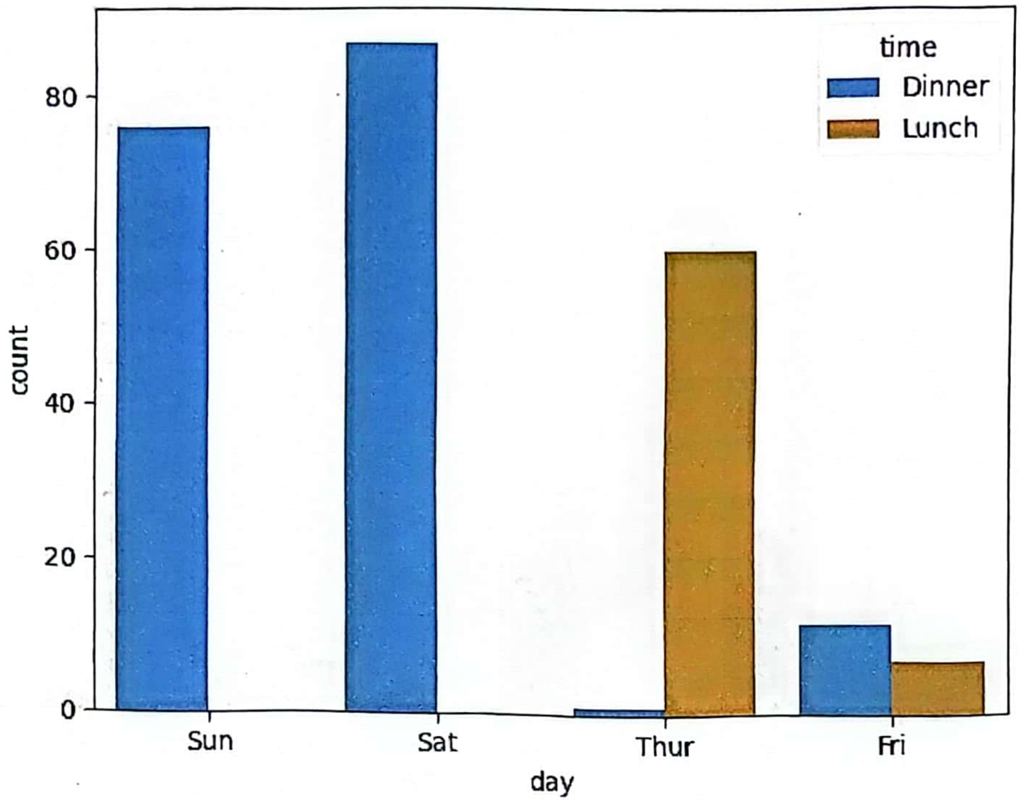
hue "Feature 2"

day "x Axis"




```
[49]: sns.countplot(x='day', data=df, hue='time')
```

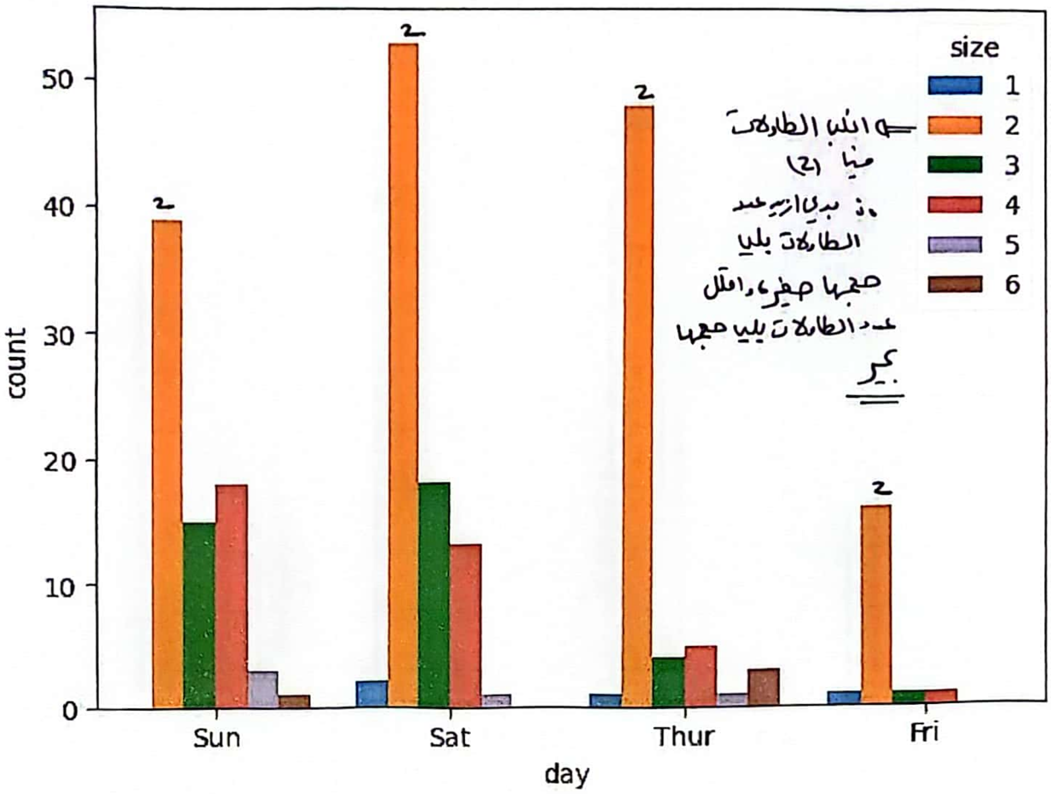
```
[49]: <AxesSubplot:xlabel='day', ylabel='count'>
```



(عدد الاشخاص يلي بيجو)

```
[50]: sns.countplot(x='day', data=df, hue='size')
```

```
[50]: <AxesSubplot:xlabel='day', ylabel='count'>
```



[25]:
```
s = df['day'].value_counts()
s
```

⇒ Category لشوف كل
وكم مرة اتكرر record لا

[25]:
```
Sat     87
Sun     76
Thur    62
Fri     19
Name: day, dtype: int64
```

[32]:
```
s = df['day'].value_counts()
plt.pie(s,autopct=(%5).1f')
plt.legend(labels = s.index ,loc="best")
plt.show()
```

⇒ [يلي انا برسم الـ pie]

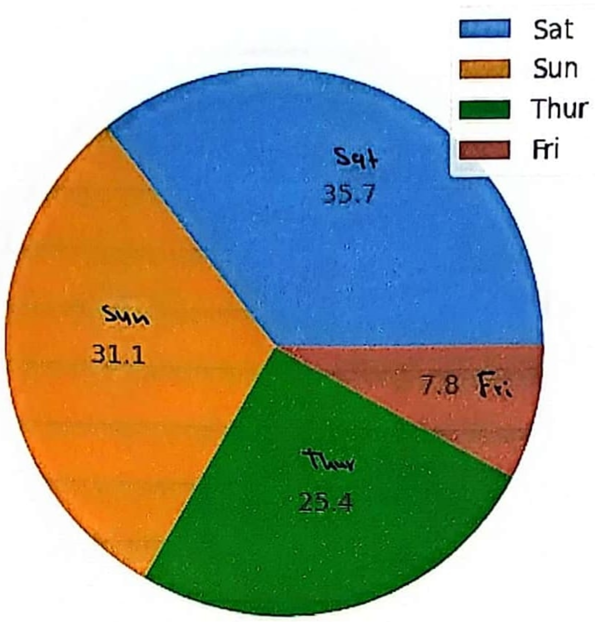
(عدد المنازل العشرية)

(لتقريبهم او بيعرضهم لنا الوسط)

المربع يلي فيه Sat Sun

ارسم عدد الزوار (percentage) حسب الـ day

[Count plot] مش دائماً برسم [pie plot] هكذا برسم

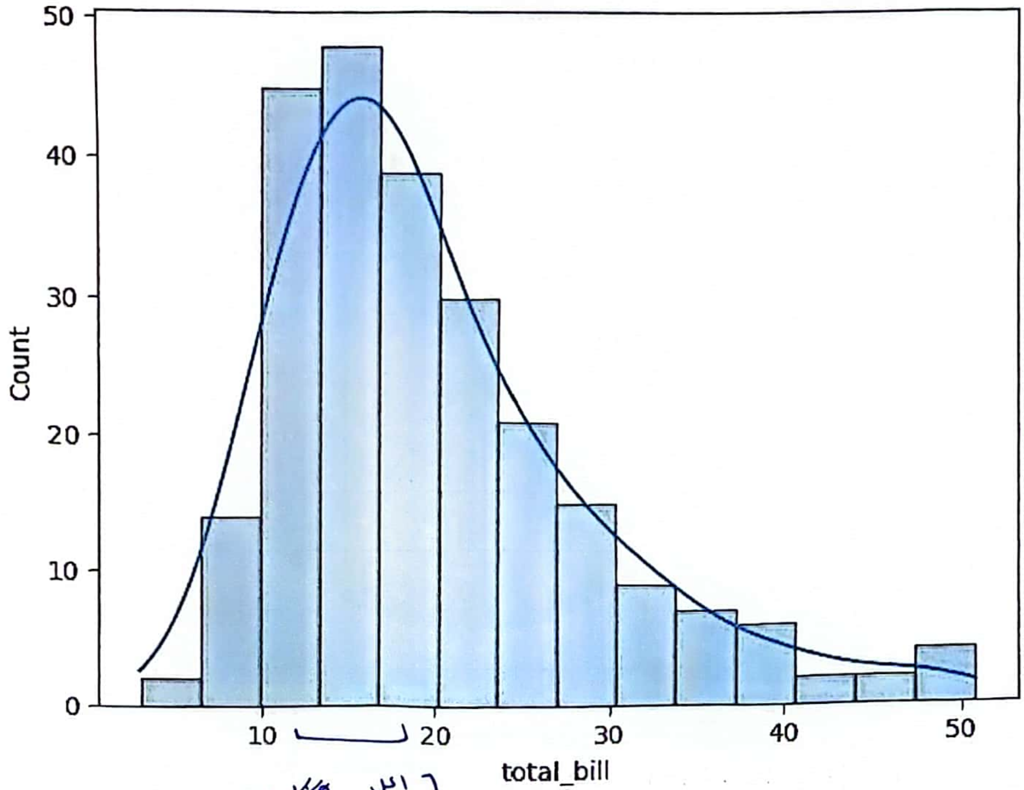


Seaborn التابع لل
[histogram عشان ارسم الـ]

```
[55]: sns.histplot(df['total_bill'], kde=True)
```

[عشان ارسم الخط الازرق]

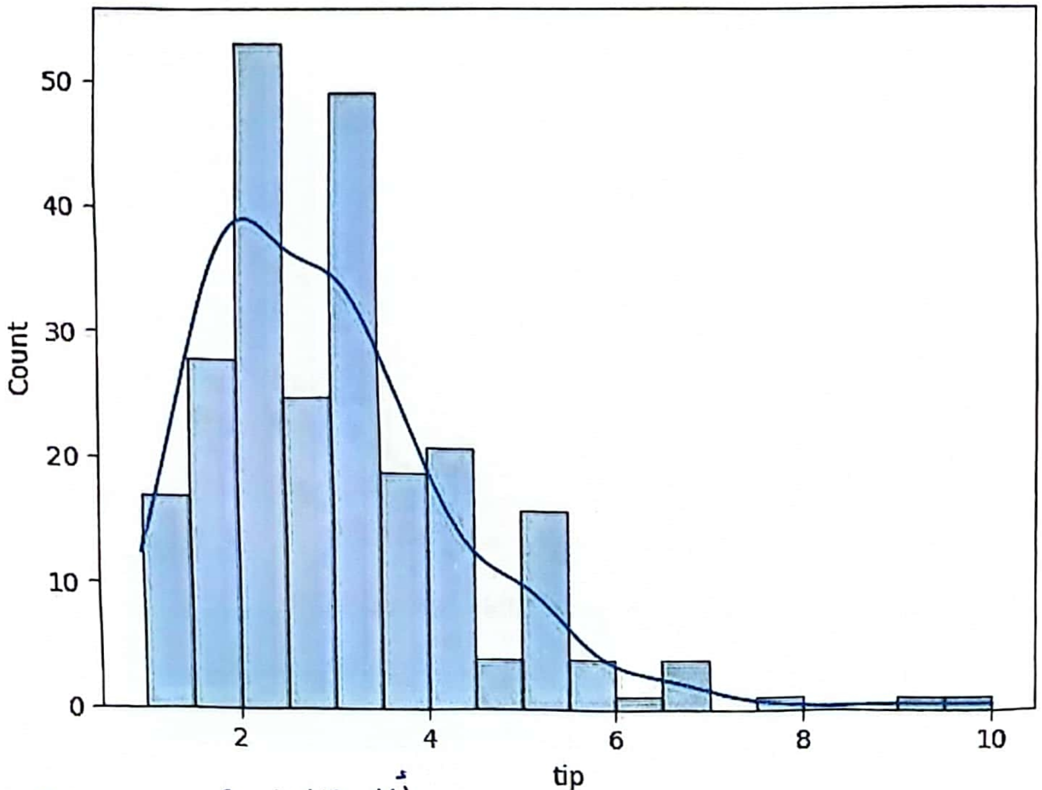
```
[55]: <AxesSubplot:xlabel='total_bill', ylabel='Count'>
```



[[10-20] بين ال ~~tip~~ total bill ال]

```
[34]: sns.histplot(df['tip'], kde=True)
```

```
[34]: <AxesSubplot:xlabel='tip', ylabel='Count'>
```

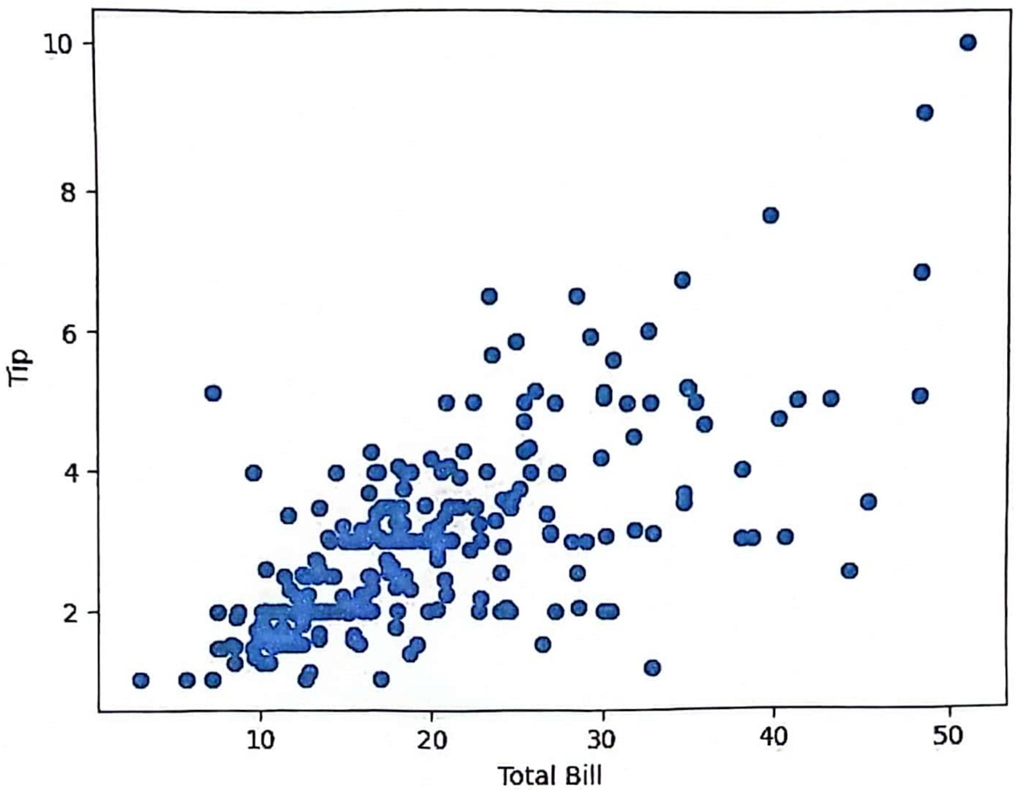


أكثر tip بين [1.0 والـ 3.2]

Scatter : [نقاط]

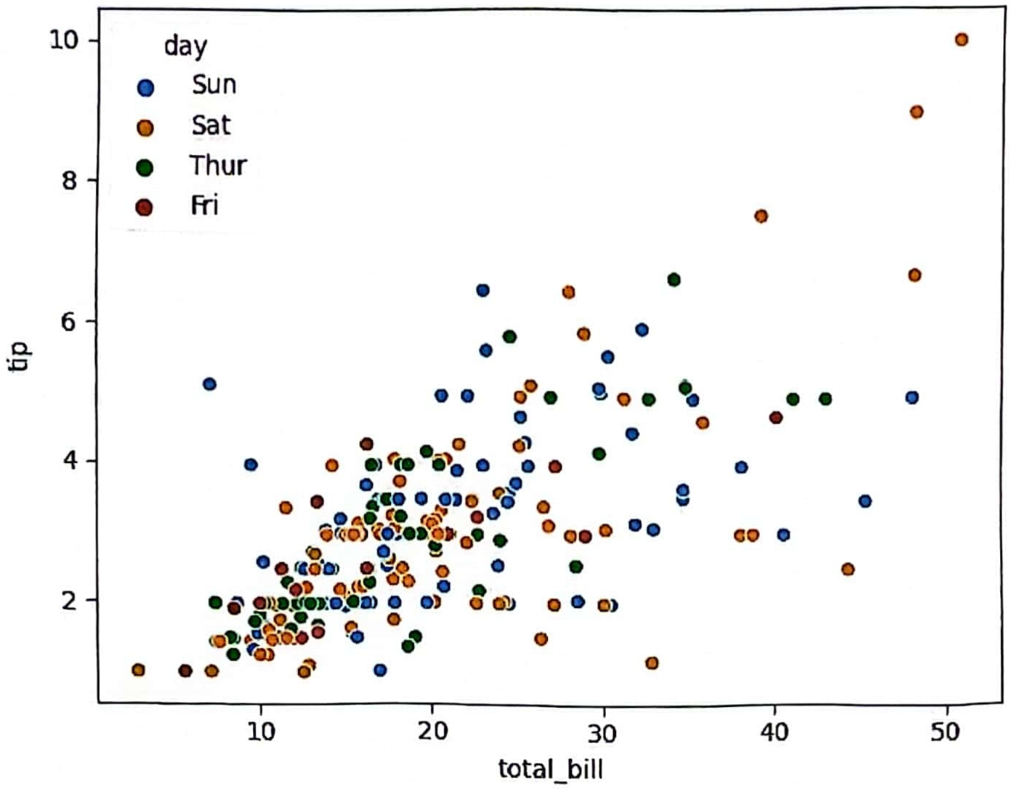
يرسم العلاقة بين total والـ tip

```
[36]: plt.scatter(df['total_bill'],df['tip'])
      plt.xlabel('Total Bill')  # Label for the x-axis
      plt.ylabel('Tip')  # Label for the y-axis
      plt.show()
```

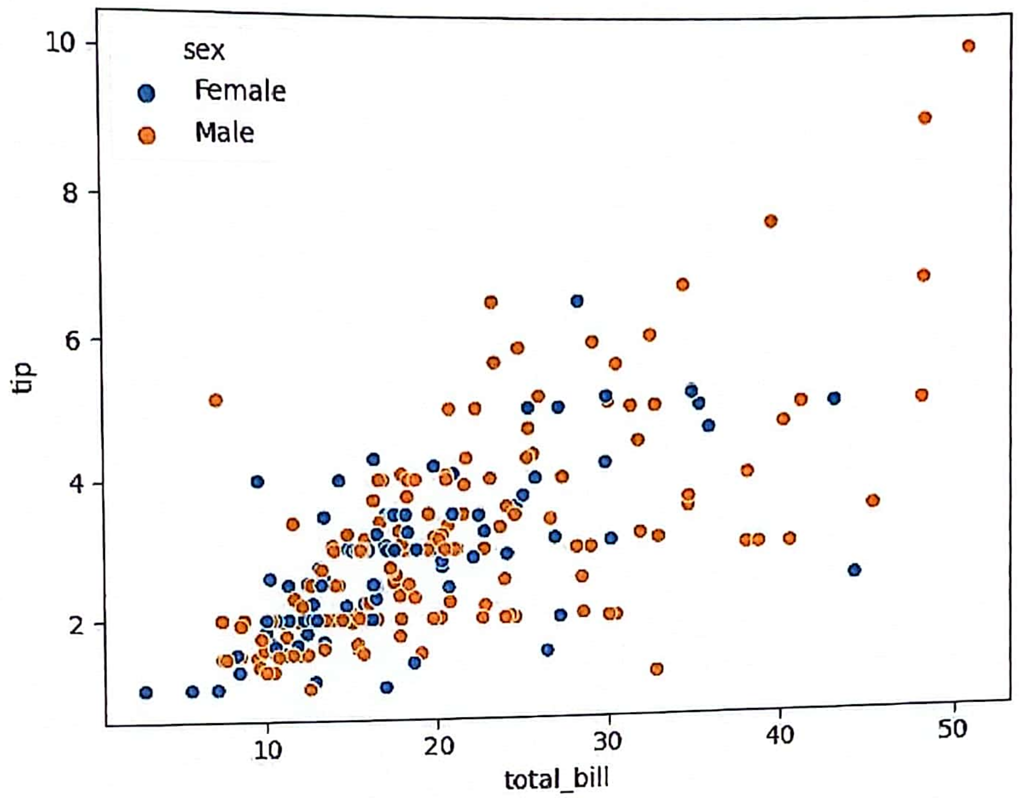


[21]:
```
sns.scatterplot(x='total_bill',y='tip',data=df,hue='day')
plt.show()
```

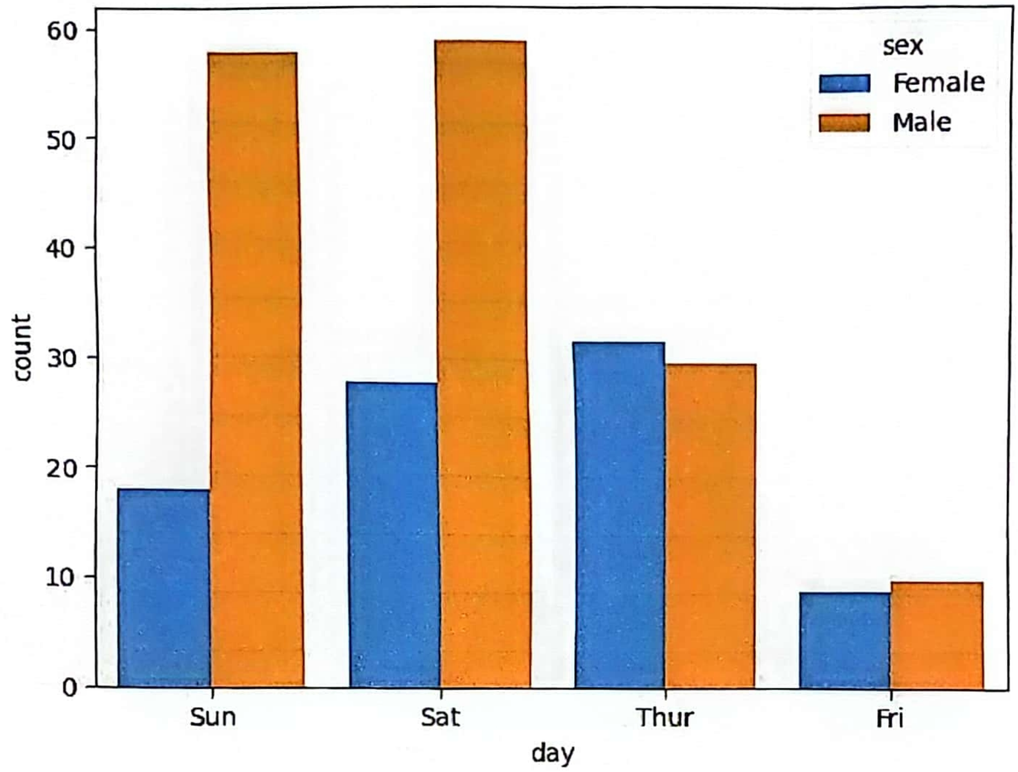
[العلاقة بين ال total وال tip وال day]



[22]:
```
sns.scatterplot(x='total_bill',y='tip',data=df,hue='sex')
plt.show()
```
(male, female)



[23]:
```
sns.countplot(x='day',data=df,hue='sex')
plt.show()
```

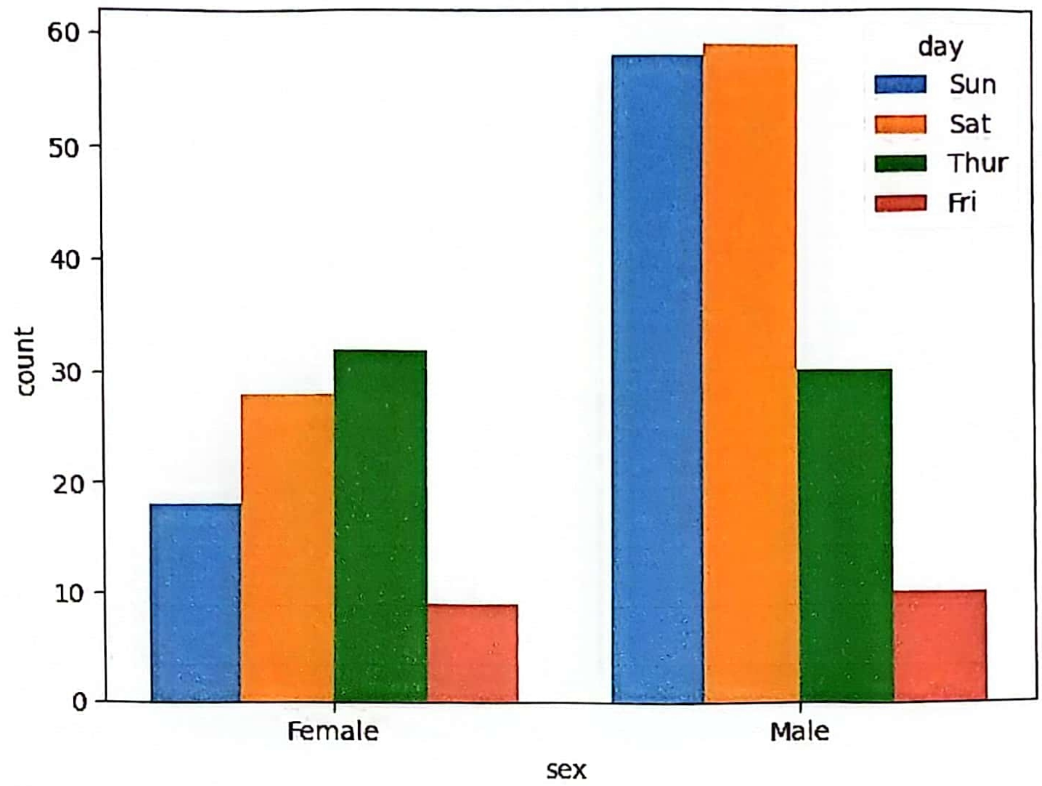


[58]:
```
df.head()
```

[58]:

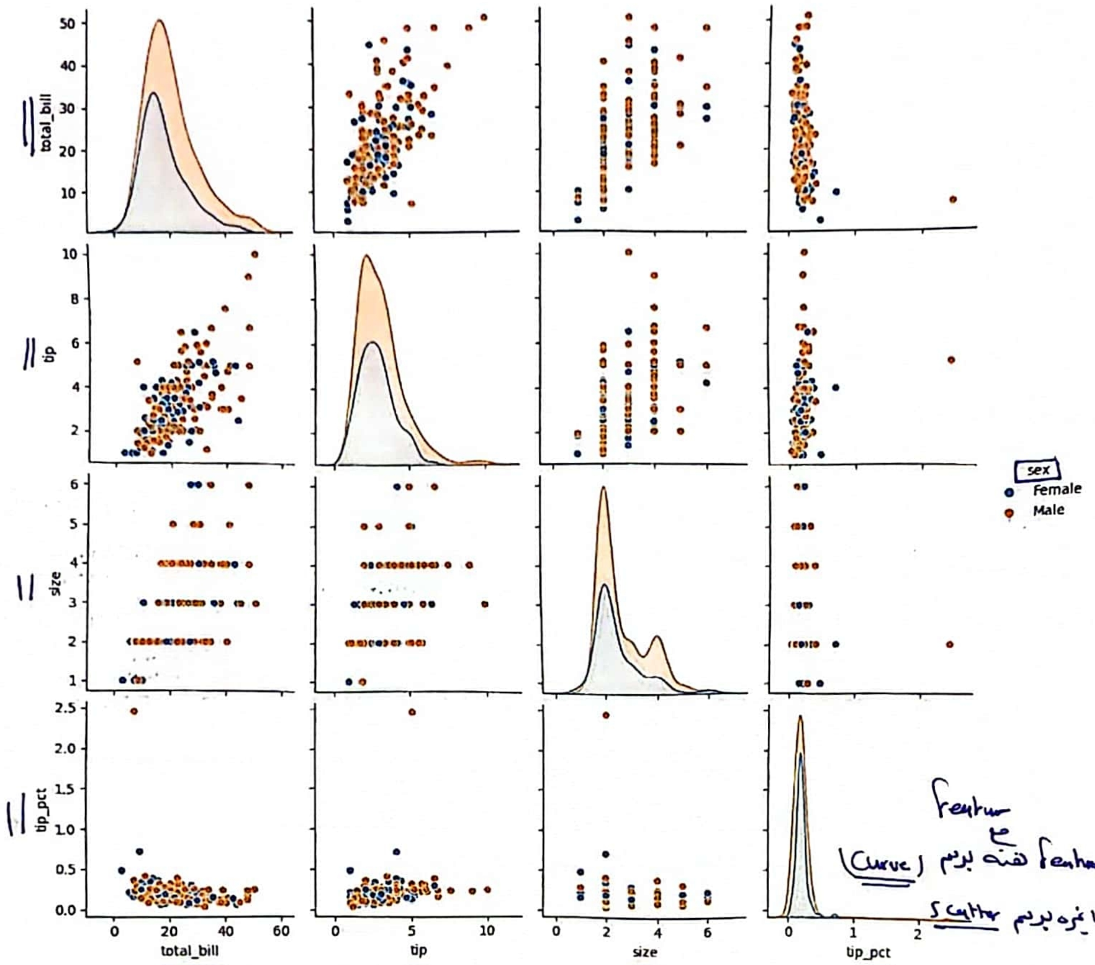
	total_bill	tip	sex	smoker	day	time	size	tip_pct
0	16.99	1.01	Female	No	Sun	Dinner	2	0.063204
1	10.34	1.66	Male	No	Sun	Dinner	3	0.191244
2	21.01	3.50	Male	No	Sun	Dinner	3	0.199886
3	23.68	3.31	Male	No	Sun	Dinner	2	0.162494
4	24.59	3.61	Female	No	Sun	Dinner	4	0.172069

[24]:
```
sns.countplot(x='sex',data=df,hue='day')
plt.show()
```



[59]:
```python
sns.pairplot(data=df,hue='sex')
plt.show()
```

العلاقة الكل مع الكل
[عمودة اكبر]
ورابطا منهم ال Sex



feature مع feature فقنه برسم (curve)
اما غيره برسم scatter

نقدر اضيف feature جديد

```
[37]: df['tip_pct'] = df['tip'] / (df['total_bill'] - df['tip'])
```

⇒ انشأت new column في الـ df اسمه [tip-pct] عبارة عن Expr.

```
[27]: df
```

[عادةً column ما بجيني في decision الـ age : لستبدله بـ العمر الفعلي]

[27]:

	total_bill	tip	sex	smoker	day	time	size	tip_pct
0	16.99	1.01	Female	No	Sun	Dinner	2	0.063204
1	10.34	1.66	Male	No	Sun	Dinner	3	0.191244
2	21.01	3.50	Male	No	Sun	Dinner	3	0.199886
3	23.68	3.31	Male	No	Sun	Dinner	2	0.162494
4	24.59	3.61	Female	No	Sun	Dinner	4	0.172069
..	...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3	0.256166
240	27.18	2.00	Female	Yes	Sat	Dinner	2	0.079428
241	22.67	2.00	Male	Yes	Sat	Dinner	2	0.096759
242	17.82	1.75	Male	No	Sat	Dinner	2	0.108899
243	18.78	3.00	Female	No	Thur	Dinner	2	0.190114

df['Age'] = [ ]
المعادلة كيف يحسبو

⇐ احياناً ليصير عندي new feature
OR
New derived feature
لحالها [derived] من الـ feature يلي اصلاً موجودة عندك

```
[244 rows x 8 columns]
```

[60]:
```
df.describe()
```

[60]:

	total_bill	tip	size	tip_pct
count	244.000000	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672	0.202123
std	8.902412	1.383638	0.951100	0.163385
min	3.070000	1.000000	1.000000	0.036955
25%	13.347500	2.000000	2.000000	0.148274
50%	17.795000	2.900000	2.000000	0.183110
75%	24.127500	3.562500	3.000000	0.236821
max	50.810000	10.000000	6.000000	2.452381

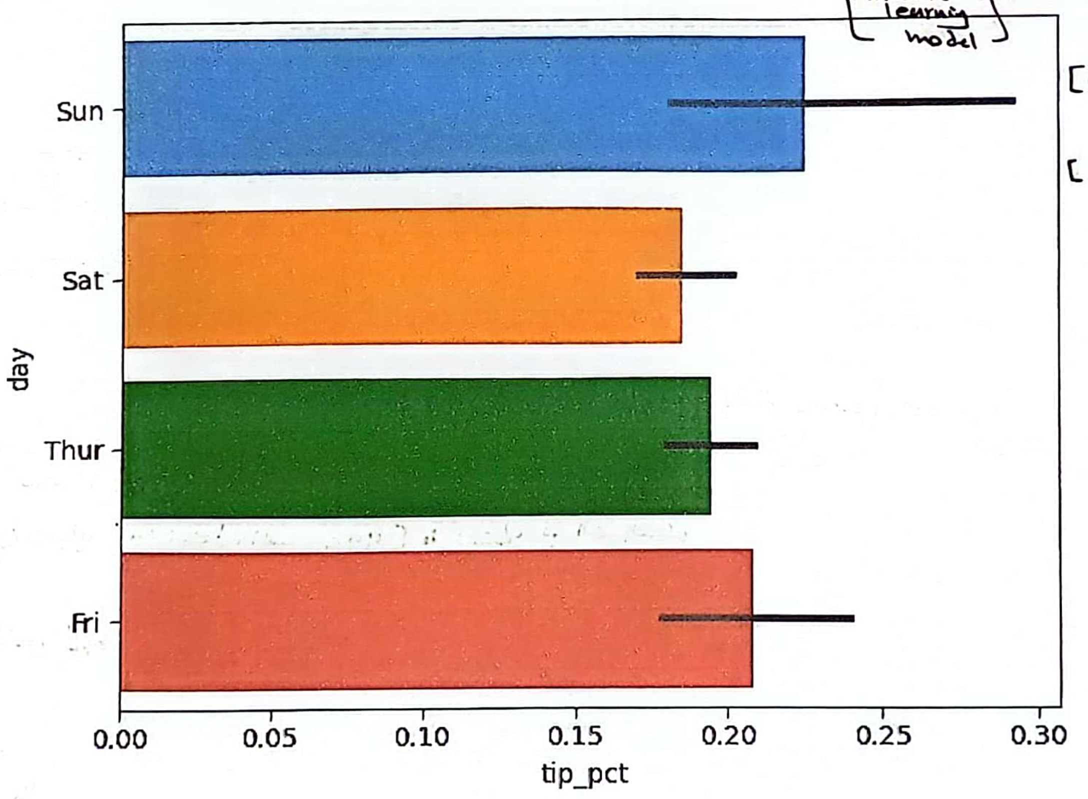
يرسم بالعرض مش بالطول

[33]:
```
sns.barplot(x='tip_pct', y='day', data=df, orient='h')
```

(horizontal)

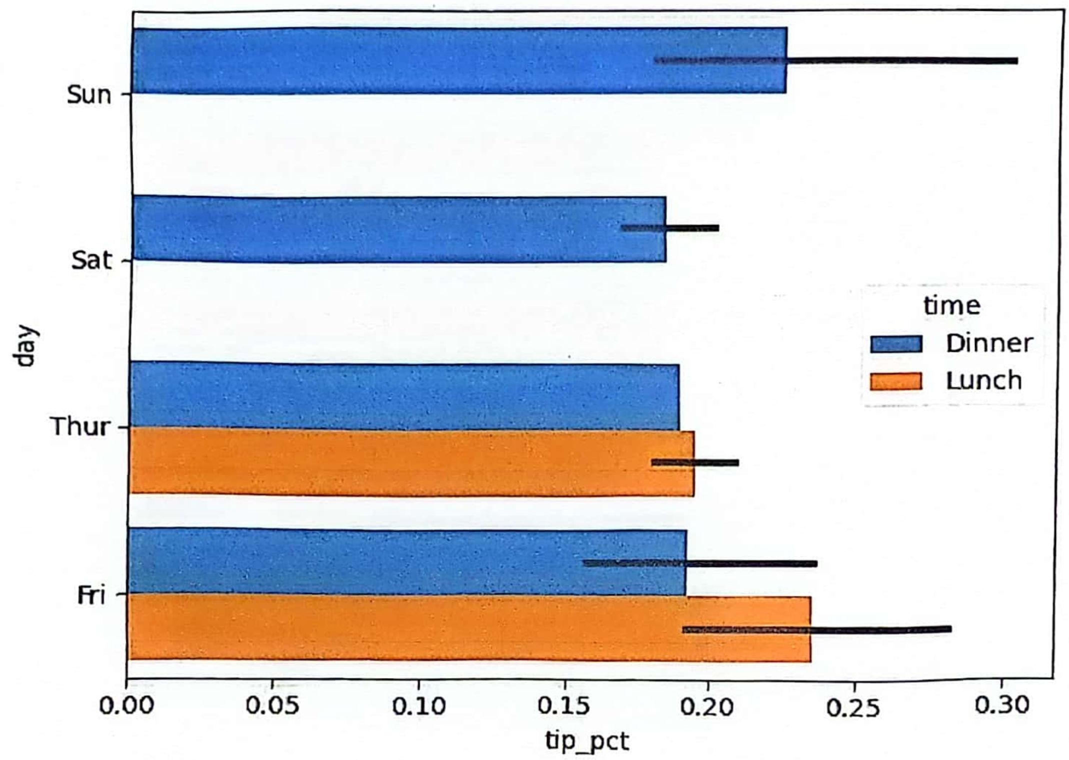
بوجد اكثر [flexibility]
عشان ارسم واعمل [visualize data]
وهادا شي بفيدني في فهم الداتا
وتحليلها قبل ما ارسلها لـ [machine learning model]
قبل ما اعمل [classification]
وقبل ما اعمل [Regression]

[33]: <AxesSubplot:xlabel='tip_pct', ylabel='day'>



[36]:
```
sns.barplot(x='tip_pct', y='day', hue='time', data=df, orient='h')
```

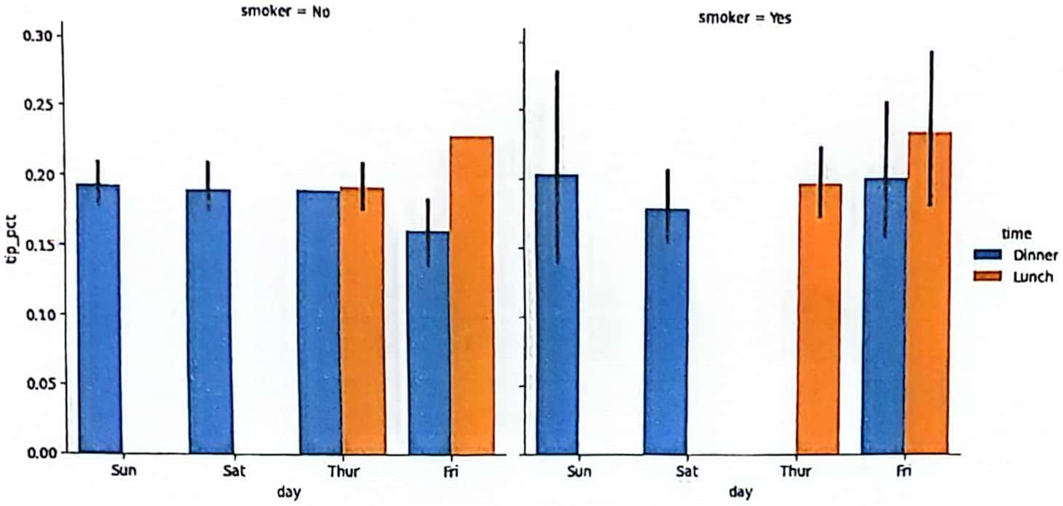
[36]: <AxesSubplot:xlabel='tip_pct', ylabel='day'>



[62]:
```
sns.factorplot(x='day', y='tip_pct', hue='time', col='smoker',kind='bar',␣
    ↪data=df[df.tip_pct < 1])
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\categorical.py:3717:
UserWarning: The `factorplot` function has been renamed to `catplot`. The
original name will be removed in a future release. Please update your code. Note
that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in
`catplot`.
  warnings.warn(msg)
```

[62]: <seaborn.axisgrid.FacetGrid at 0x1f68ae64910>



[41]:
```
df
```

[41]:

	total_bill	tip	sex	smoker	day	time	size	tip_pct
0	16.99	1.01	Female	No	Sun	Dinner	2	0.063204
1	10.34	1.66	Male	No	Sun	Dinner	3	0.191244
2	21.01	3.50	Male	No	Sun	Dinner	3	0.199886
3	23.68	3.31	Male	No	Sun	Dinner	2	0.162494
4	24.59	3.61	Female	No	Sun	Dinner	4	0.172069
..	...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3	0.256166
240	27.18	2.00	Female	Yes	Sat	Dinner	2	0.079428
241	22.67	2.00	Male	Yes	Sat	Dinner	2	0.096759
242	17.82	1.75	Male	No	Sat	Dinner	2	0.108899
243	18.78	3.00	Female	No	Thur	Dinner	2	0.190114

[244 rows x 8 columns]

[42]:
```
df.describe()
```

[42]:

	total_bill	tip	size	tip_pct
count	244.000000	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672	0.202123
std	8.902412	1.383638	0.951100	0.163385
min	3.070000	1.000000	1.000000	0.036955
25%	13.347500	2.000000	2.000000	0.148274
50%	17.795000	2.900000	2.000000	0.183110
75%	24.127500	3.562500	3.000000	0.236821
max	50.810000	10.000000	6.000000	2.452381

[ ]:

Testing and Validating

- know how well a model will generalize to new cases is to actually try
it out on new cases.

- A better option is to split your data into two sets: the training set and the test set.
You train your model using the training set, and you test it using the test set.

- Evaluating your model on the test set, you get an estimate of model accuracy. This value tells you how well your model will perform on instances it has never seen before

Testing and Validating

It is common to use 80% of the data for training and hold out 20% for testing. However, this depends on the size of the dataset: if it contains 10 million instances, then holding out 1% means your test set will contain 100,000 instances: that's probably more than enough to get a good estimate of the generalization error.

مش شرط دائماً
آخذ 20%
اذا حجم الداتا كبير
ممكن آخذ 1%
او 2%

CamScanner الممسوحة ضوئيا بـ

Measuring performance (Confusion matrix)

[بتقسم مين الصح ومين الغلط]

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

ال Actual No + Predicted No
يسمى "TN" (True negative)

positive class: yes مع مرضه انه
negative class: No

ال Actual yes + Predicted yes
يسمى "Tp" (True positive)

المشكلة مقصود الـ
① من الكمية 60 عنده negative class

ال Actual No "negative" وانا عملت predicted yes "positive"
يسمى "Fp" False positive
لانه التوقع غلط

اما لو عمل Actual yes "positive" وانا عملت predicted No "negative"
يسمى "FN" False negative

② من الكمية 105 عنده positive class

(توقع الـ model انه (55) الرقم negative) (توقع الرقم (positive))

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known

[بعرف الـ 60 والـ 105 من الـ [data set] نفسها
التوقع مش لازم يكون قد الـ 60 والـ 105 ! الـ model متوقع الـ 55 والـ 110

Confusion matrix (Example)

* بـ اي classification problem عندي الـ class اسمه yes "T" ، الـ class اسمه No "F"

There are two possible predicted classes: "**yes**" and "**no**".
If we were predicting the presence of a disease.
for example, "**yes**" would **mean they have the disease**,
and "no" would mean they don't have the disease.

(أي انه يوجد 165 مريض نفحص بوجود المرض او لعدم وجوده)

The classifier made a total of 165 predictions
(e.g., 165 patients were being tested for the presence of that disease).

Confusion matrix (Example)

21

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	
	No	yes	

* out of the 110 predicted yes
كان منهم "100" "Correctly predicted"
فـ الـ TP "10"

يعني توقعت 110 تا لحين

Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times. In reality, 105 patients in the sample have the disease, and 60 patients do not.

yes الي هو class لل positive الـ TP 6 فـ الـ ☑ 100 ← Class yes لل
TP=100 طرحت من 110 بيصير 10
فـ الـ $\boxed{10 = FP}$

* out of the 55 no prediction يعني توقعت انه (55) الرقم (Negative) OR no ← 50 منهم (Correctly predictive) صحاح الـ $\boxed{50 = TN}$ والـ $\boxed{5 = FN}$

حسب كل الحالات ولهتم 6 اول اشي بحدد الـ TP والـ TN

الـ TP بيرجع لل توقعات الـ yes "110" بيكون منهم 100 صحيحات TP والباقي لل FP
لعدين بيرجع لل Negative حالي 55 ← 50 منهم صح فـ 50 = TN
5 لل FN

Confusion matrix (Example)

22

- **True positives (TP):** These are cases in which we predicted yes , and they do have the disease.

* طلع فرض انه الحالة اول الـ cases
No disease OR disease
[مريض او لا]

- **True negatives (TN):** We predicted no, and they don't have the disease.

- **False positives (FP):** We predicted yes, but they don't actually have the disease.

- **False negatives (FN):** We predicted no, but they actually do have the disease.

[الـ 4 Cases ب اي Classification problem لازم ادرسهم]

Confusion matrix (Example)

23

[نقسم التوقع الصح / كل التوقعات]

Accuracy: Overall, how often is the classifier correct?
(TP+TN)/total = (100+50)/165 = 0.91

كل التوقعات

ش نتعامل للجمع TN و TP

يعني نسبة ال error ⇐ **Misclassification** Rate: Overall, how often is it wrong?

(1 - Accuracy) OR ⇐ (FP+FN)/total = (10+5)/165 = 0.09
equivalent to 1 minus Accuracy
also known as "Error Rate"

* ال Confusion matrix استخدمناها في حالة ال classification وفي حالة ال [regression] مافيها [TN , TP]

ال [regression] مفهومها prediction "توقع" فالتوقع اما صح أو خطأ، نستخدم ال (RMSE error)

Ex) Feature [Temp.] prediction

X_1	50	49
X_2	25	26
X_3	43	42.999

Target Value "leabel" Actual ← $h(x^i)$ prediction value

[X, y is a vector of all Feature value]

درجة الحرارة الفعلية في data set — توقع ال model

Root Mean Square Error (RMSE)

24

بكرر من ال 1 و m ، m معناها لكل ال row الموجودة عندي في (Data set) لكل record

(y : عدد ال (Column))

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{\boxed{m}} \sum_{i=1}^{\boxed{m}} \left(\overbrace{h\left(\mathbf{x}^{(i)}\right)}^{\text{Fun. "model"}} - y^{(i)}\right)^{2}}$$

عدد ال row

(نتطرح ال prediction ناقص ال Actual)

لو نجيبنا ال Feature يلي عندي في (model)

Vector prediction — Actual Value

لحساب ال error مابنربط ال Confusion matrix لانه هو اما zero او one "0,1"

يعني 50 و 49 رح يكون التوقع خطأ !

و 42.999 و 43 رح يكون خطأ !

من (Conf. matrix) مش منطق نستخدم 66% طريقة اخرى لحساب ال error

m is the number of instances in the dataset

For example, if you are evaluating the RMSE on a validation set of 2,000 districts, then m = 2,000. (record) OR row

لو يعني عندي 2000 (record) من data set

في المثال ال label هو ال Target Value Temp

$y^{(i)}$, $\mathbf{x}^{(i)}$ is a vector of all the feature values (excluding the label) of the i_{th} instance in the dataset, and $y^{(i)}$ is its label (the desired output value for that instance). يلي اشتغل عليها

يلي بتجزل record عن التاني

* في ال classification شادد label هل الشي موجود ام مش موجود، اما في regression توقع "Temp" من هذا المثال

Root Mean Square Error (RMSE)

25

$$x^{(1)} = \begin{pmatrix} -118.29 \\ 33.91 \\ 1,416 \\ 38,372 \end{pmatrix} \begin{matrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{matrix}$$

Ex

$\Rightarrow h(x_i)$ (prediction y) بطلع ال

and:

record الأول label ال $\boxed{y^{(1)} = 156,400}$ $\Rightarrow$ Actual price يسمى لكل record

(Actual ال y ال)

" اذا كان التوقع عبارة عن سعر بيت "

h(x^i) is your system's prediction function

For example, if your system predicts that the median housing price in the first district is \$158,400, then $\hat{y}_{(1)} = h(\mathbf{x}_{(1)}) = 158,400$. The prediction error for this district is $\hat{y}_{(1)} - y_{(1)} = 2,000$.

اذا بديا اجيب ال prediction price !?

(طرحنا ال predi من ال Actual)

y predicted

اسمه ال model هو (h) بستقبل ال $(x) \Leftarrow h(x^i)$

بطلع ال (prediction (y))

بحسب ال (RMSE)

بنقص ال predi. من ال Actual وداخل تربيع

$$\frac{(\text{pred.} - \text{Actual})^2 + (\text{row}2) + (\text{row}3) + \cdots}{\text{record عدد ال}}$$

$\Rightarrow$ بحسب ال error

دبعطيني أدق شي ال error

اما في Conf. matrix بطلعوا كله غلط

*الفرق بين ال prediction وال Actual هي يلي بتعطيني شو ال (Accuracy) تاعت ال model

Hashemite University

Computer Engineering Department

Data Science

Data Cleaning and Preparation

(Chp7, Text 3)

Dr. Mohammad Al-hammouri

Data Cleaning and Preparation

1. Handling Missing Data (اعالجها) — اي داتا موجودة (Null) لازم نتعامل معها بالطريقة الصحيحة ⇒ ما بنربط الـ Table فيه Null ع شان ح يعطي result غلط ليعرف طرق للتعامل معه
2. Filtering Missing Data → اعرف مواقع الـ Missing Data وعددها
3. Data Transformation
   - Removing duplicates ⇒ عادي اذا يكون في two row (ـ التكرار افقي) / * اذا في اكثر من row مكرر وبالكامل لازم احذف واحد منهم
   - Replacing values (بمعنى عادي Column)
4. Detecting and Filtering Outliers → مثلاً عندك Column اسمه Age ويتدرس على صف الـ خامس (3) مش منطقي !! ⇒ (ابدلو لتكون خطأ) outliers
5. Handling Text and Categorical Attributes

2

Handling Missing Data

null ← نفترض missing Data اذا موجود في (Data Set)

- Pandas uses the floating-point value NaN (Not a Number) to represent missing data.

(non a number)

- string_data = pd.Series(['aardvark', 'artichoke', np.nan, 'avocado'])

لما نبين انو value null يعني بشيك على كل (entry) ويحكي "F" not null "T" is null

```
In [11]: string_data
Out[11]:
0     aardvark
1    artichoke
2          NaN
3      avocado
dtype: object
```

```
In [12]: string_data.isnull()
Out[12]:
0    False
1    False
2     True   => is null
3    False   -> np.nan => not null
dtype: bool
```

```
In [13]: string_data[0] = None

In [14]: string_data.isnull()
Out[14]:
0     True   — null
1    False
2     True   — null
3    False
dtype: bool
```


Handling Missing Data

- Filtering Out Missing Data (dropna())

(لعرض استخدم NA بدل non)

(np.nan) بنستخدمها بدل

```
In [15]: from numpy import nan as NA   -> numpy

In [16]: data = pd.Series([1, NA, 3.5, NA, 7])

In [17]: data.dropna()
Out[17]:
0    1.0
2    3.5
4    7.0
dtype: float64
```

مهم بعمل drop لل (null Value)

(بعطي data of الـ data not null)

```
In [18]: data[data.notnull()]
Out[18]:
0    1.0
2    3.5
4    7.0
dtype: float64
```

ترجع indexes يلي ما فيها not null ترجع

3-data-cleaning-1

March 23, 2024

[1]:
```
import pandas as pd
import numpy as np
```

هنا mix بين null value و non و none

[2]:
```
string_data = pd.Series(['aardvark', 'artichoke', np.nan, 'avocado'])
```

[3]:
```
string_data.isnull()
```

[3]:
```
0    False
1    False
2     True
3    False
dtype: bool
```

[4]:
```
string_data[0] = None
```

[6]:
```
#Filtering Out Missing Data
from numpy import nan as NA
data = pd.Series([1, NA, 3.5, NA, 7])
```

[7]:
```
data=data.dropna()
```

⇒ (أي row فيه null بتحذفه)

بتشيل الـ row كامل

[8]:
```
data
```

[8]:
```
0    1.0
2    3.5
4    7.0
dtype: float64
```

[15]:
```
data = pd.DataFrame([[1., 6.5, 3.], [1., NA, NA],
....: [NA, NA, NA], [NA, 6.5, 3.]])
```

[10]:
```
data
```

⇒ // رجعت زي ما هي

هاي الـ Comm. ما تعدل مباشرة عل الداتا

[10]:
```
     0    1    2
0  1.0  6.5  3.0
1  1.0  NaN  NaN
2  NaN  NaN  NaN
```

" عملت drop ، لكن ما عدلت عل الـ data "
بتعدل عليها اذا عملت مساواة (=)

```
3  NaN  6.5  3.0
```

```
[11]: data.dropna()
```

```
[11]:      0    1    2
       0  1.0  6.5  3.0
```

```
[13]: data = data.dropna()
```

⇒ (بعدّل عـ الداتا بطريقة مباشرة)

```
[16]: data
```

```
[16]:      0    1    2
       0  1.0  6.5  3.0
       1  1.0  NaN  NaN
       2  NaN  NaN  NaN
       3  NaN  6.5  3.0
```

```
[10]: cleaned = data.dropna()
```

(الداتا ما بتأثر بطريقة مباشرة عليها)
خزنت فيها cleaned

```
[11]: cleaned
```

```
[11]:      0    1    2
       0  1.0  6.5  3.0
```

```
[11]: data
```

```
[11]:      0    1    2
       0  1.0  6.5  3.0
       1  1.0  NaN  NaN
       2  NaN  NaN  NaN
       3  NaN  6.5  3.0
```

```
[12]: cleaned
```

```
[12]:      0    1    2
       0  1.0  6.5  3.0
```

```
[14]: data.dropna(how='all')
```

⇒ (Delete) بحذف الـ row فقط اذا
كان كل الـ row عبارة عن null

```
[14]:      0    1    2
       0  1.0  6.5  3.0
       1  1.0  NaN  NaN
       3  NaN  6.5  3.0
```

⇒ ما حذفهم لأنه مش كل
الـ row عبارة عن null

حذف (index 2)

```
[14]: data
```

[14]:

	0	1	2
0	1.0	6.5	3.0
1	1.0	NaN	NaN
2	NaN	NaN	NaN
3	NaN	6.5	3.0

[19]:
```
data[4] = NA
```
⇒ معناه ضيف (new Column) رقمه (4) مبدئياً كلهم null

[20]:
```
data
```

[20]:

	0	1	2	4
0	1.0	6.5	NaN	NaN
1	1.0	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	6.5	NaN	NaN

* بتحذف ال Column اذا كلهم null

(4,2) ح ينحذف لانه التنين null

* اذا بدي اعمل dropna على مستوى ال Column

[21]:
```
#delete column if all are NA
data.dropna(axis=1, how='all')
```

* لقصدينا ال (y-axis) يعني ال Column

[21]:

	0	1
0	1.0	6.5
1	1.0	NaN
2	NaN	NaN
3	NaN	6.5

[17]:
```
data
```

[17]:

	0	1	2	4
0	1.0	6.5	3.0	NaN
1	1.0	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	6.5	3.0	NaN

[24]:
```
#The np.random.randn function in NumPy generates samples from the standard␣
↪normal distribution,
#This means that most of the generated numbers will fall within the range of␣
↪about -3 to 3-3 to 3

df = pd.DataFrame(np.random.randn(7, 3))
```
⇒ بدي اعمل data Fram الي بالحجم

normal distribution بحد حولين ال Zero

[25]:
```
np.random.randn(7, 3)
```
⇒ (Random Variable Value)

ح نعطي ارقام من -3 لل 3

[25]:
```
array([[-0.67356087, -0.45357337,  1.42600949],
       [ 1.02914203,  1.6100844 , -0.04983307],
       [ 0.29948237, -0.64751821, -0.86883525],
       [-1.41497484,  0.30871947,  0.54809519],
       [ 0.8556187 , -2.03673211, -0.83981317],
```
⇒ numpy ⇒ ما تبين Index

(7,3) العدد ⇒ هنا 7 اسطر 4 اعمدة

اي numpy ⇒ عشان احول لل data Fram بحط

dataFram(array)

بتحط عبر داتا فريم ، افضل من numpy ، لانه بترجم الاسطر والاعمدة ليصير اسهل

```
       [-1.45358381, -0.34964666,  0.1109247 ],
       [ 0.77201501, -0.63450747, -0.1268022 ]])
```

[21]:
```
df
```
(data fram)

[21]:

	0	1	2
0	0.280182	-0.076201	-1.960355
1	1.075388	-0.895660	0.386111
2	-2.366018	0.304589	0.002855
3	1.582293	-0.185548	-0.620989
4	0.593570	-0.331516	0.515896
5	0.035016	0.623496	2.146304
6	1.148683	0.182433	-1.010698

Column (1) ← اول (3) اسطر من العامود الاول

[26]:
```
df.iloc[:4, 1] = NA
```
→ Access لتغير من خلاله لعدد لبعض الـ rows الموجودة في dataframe

[23]:
```
df
```

[23]:

	0	1	2
0	0.280182	NaN	-1.960355
1	1.075388	NaN	0.386111
2	-2.366018	NaN	0.002855
3	1.582293	NaN	-0.620989
4	0.593570	-0.331516	0.515896
5	0.035016	0.623496	2.146304
6	1.148683	0.182433	-1.010698

[27]:
```
df.iloc[:2, 2] = NA
```
(العامود الثاني اول سطرين (0,1))

[28]:
```
df
```

[28]:

	0	1	2
0	0.731182	NaN	NaN
1	1.010105	NaN	NaN
2	-0.068527	NaN	0.401530
3	-0.292148	NaN	-1.658488
4	-0.142982	-0.036343	-1.212064
5	-1.034044	0.046531	0.908259
6	2.244516	0.069813	0.047924

[9]:
```
df.dropna()
```
⇒ (بتشيل اي سطر فيه الـ null)

حذف بغض النظر عدد الـ null مرة او اكثر

[9]:

	0	1	2
4	1.294844	-0.704270	-0.980718
5	-0.336609	0.470196	0.988733
6	0.319031	-1.390572	0.983515

[29]:
```
df
```

[29]:
```
          0         1         2
0  0.731182       NaN       NaN
1  1.010105       NaN       NaN
2 -0.068527       NaN  0.401530
3 -0.292148       NaN -1.658488
4 -0.142982 -0.036343 -1.212064
5 -1.034044  0.046531  0.908259
6  2.244516  0.069813  0.047924
```

[27]:
```
# thresh=2 means, the rows needs at least two not null values to survive
df.dropna(thresh=2)
```

⇒ بتحذف ال row اذا كان فيه two null

[27]:
```
          0         1         2
2 -2.366018       NaN  0.002855
3  1.582293       NaN -0.620989
4  0.593570 -0.331516  0.515896
5  0.035016  0.623496  2.146304
6  1.148683  0.182433 -1.010698
```

[10]:
```
df
```

[10]:
```
          0         1         2
0  1.215950       NaN       NaN
1  0.540135       NaN       NaN
2  0.829622       NaN -0.784088
3  0.703976       NaN  2.101510
4  1.294844 -0.704270 -0.980718
5 -0.336609  0.470196  0.988733
6  0.319031 -1.390572  0.983515
```

[32]:
```
df.dropna(thresh=3)
```

[32]:
```
          0         1         2
4  0.778466 -0.319811  0.265054
5 -0.888870  0.259853 -0.429096
6 -0.482623  1.147385  0.468441
```

[30]:
```
#Filling In Missing Data
df
```

[30]:
```
          0         1         2
0  0.731182       NaN       NaN
1  1.010105       NaN       NaN
2 -0.068527       NaN  0.401530
3 -0.292148       NaN -1.658488
```

4	-0.142982	-0.036343	-1.212064
5	-1.034044	0.046531	0.908259
6	2.244516	0.069813	0.047924

[11]:
```
df.fillna(0)
```

[11]:

	0	1	2
0	1.215950	0.000000	0.000000
1	0.540135	0.000000	0.000000
2	0.829622	0.000000	-0.784088
3	0.703976	0.000000	2.101510
4	1.294844	-0.704270	-0.980718
5	-0.336609	0.470196	0.988733
6	0.319031	-1.390572	0.983515

[31]:
```
df
```

[31]:

	0	1	2
0	0.731182	NaN	NaN
1	1.010105	NaN	NaN
2	-0.068527	NaN	0.401530
3	-0.292148	NaN	-1.658488
4	-0.142982	-0.036343	-1.212064
5	-1.034044	0.046531	0.908259
6	2.244516	0.069813	0.047924

[28]:
```
df
```

[28]:

	0	1	2
0	0.280182	NaN	NaN
1	1.075388	NaN	NaN
2	-2.366018	NaN	0.002855
3	1.582293	NaN	-0.620989
4	0.593570	-0.331516	0.515896
5	0.035016	0.623496	2.146304
6	1.148683	0.182433	-1.010698

[35]:
```
#fillna with a dict, you can use a different fill value for each column
df.fillna({1: 0.5, 2: 0})
```

$\boxed{٠,٥}$ عبي الموجود في العامود الاول ، ما في العامود (٢) عبي zero

* بدنا عشان نحذف ال null
بنالغرض عشان نعمل Fill
اوقات حذف ال row هو الحل
بالنسبة لل null

[35]:

	0	1	2
0	0.271510	0.500000	0.000000
1	-0.473364	0.500000	0.000000
2	0.731081	0.500000	0.761048
3	-1.536339	0.500000	0.426161
4	0.778466	-0.319811	0.265054
5	-0.888870	0.259853	-0.429096

من dictionary داخل Fillna

```
6 -0.482623  1.147385  0.468441
```

```
[32]: df
```

```
[32]:           0         1         2
      0  0.731182       NaN       NaN
      1  1.010105       NaN       NaN
      2 -0.068527       NaN  0.401530
      3 -0.292148       NaN -1.658488
      4 -0.142982 -0.036343 -1.212064
      5 -1.034044  0.046531  0.908259
      6  2.244516  0.069813  0.047924
```

```
[34]: df.fillna(0, inplace=True)
```

بترجع لي كل ال null وبتقلبها zero

+ بتستعمل ال data fram لو ما عملت مساواة (data = data fram) لابد من inplace

مباشرة تعدل على داتا فريم

```
[36]: df
```

فيه zero بدل null

```
[36]:           0         1         2
      0  0.731182  0.000000  0.000000
      1  1.010105  0.000000  0.000000
      2 -0.068527  0.000000  0.401530
      3 -0.292148  0.000000 -1.658488
      4 -0.142982 -0.036343 -1.212064
      5 -1.034044  0.046531  0.908259
      6  2.244516  0.069813  0.047924
```

يعني التعديل ياخذ تأثير مباشرة مش بس يعرضه

```
[37]: df = pd.DataFrame(np.random.randn(6, 3))
```

```
[38]: df
```

```
[38]:           0         1         2
      0 -0.031204  1.443506  1.348347
      1 -0.407265 -0.373886 -0.380240
      2 -1.412912  0.663700 -1.592727
      3 -1.082016  0.235359  0.402048
      4  0.257679  1.328878 -1.966385
      5  0.109856 -0.161968 -0.064878
```

```
[ ]:
```

```
[39]: df.iloc[2:, 1] = NA
      df.iloc[4:, 2] = NA
```

```
[40]: df
```

```
[40]:           0         1         2
      0 -0.031204  1.443506  1.348347
```

	0	1	2
1	-0.407265	-0.373886	-0.380240
2	-1.412912	NaN	-1.592727
3	-1.082016	NaN	0.402048
4	0.257679	NaN	NaN
5	0.109856	NaN	NaN

[41]:
```
#fills the missing values with the last observed non-null value
df.fillna(method='ffill')
```

⇐ تبعبي ال null الموجوده من Column معين ب Non null يلي قبله

[41]:

	0	1	2
0	-0.031204	1.443506	1.348347
1	-0.407265	-0.373886	-0.380240
2	-1.412912	-0.373886	-1.592727
3	-1.082016	-0.373886	0.402048
4	0.257679	-0.373886	0.402048
5	0.109856	-0.373886	0.402048

Non-Null

copy (null)

[19]:
```
df
```

[19]:

	0	1	2
0	-0.190089	-1.199733	-0.690466
1	-0.934796	1.281987	0.327398
2	1.236654	NaN	-0.054925
3	0.323191	NaN	-1.259246
4	0.045598	NaN	NaN
5	0.435786	NaN	NaN

[20]:
```
# copy the last null value to the rows after , but only to the next two rows
df.fillna(method='ffill', limit=2)
```

ليس عبي ما two null value

[20]:

	0	1	2
0	-0.190089	-1.199733	-0.690466
1	-0.934796	1.281987	0.327398
2	1.236654	1.281987	-0.054925
3	0.323191	1.281987	-1.259246
4	0.045598	NaN	-1.259246
5	0.435786	NaN	-1.259246

توخذ آخر قيمة مش null وتعبي (two null) max.

[42]:
```
data = pd.Series([1., NA, 3.5, NA, 7])
```

[37]:
```
data
```

[37]:
```
0    1.0
1    NaN
2    3.5
3    NaN
4    7.0
```

```
dtype: float64
```

```
[44]: data.fillna(data.mean())
```

```
[44]: 0    1.000000
      1    3.833333
      2    3.500000
      3    3.833333
      4    7.000000
      dtype: float64
```

بيرجع بشوف المتوسط للـ (1, 3.5, 7)

العامود الاول

```
[44]: #Data Transformation
      data = pd.DataFrame({'k1': ['one', 'two'] * 3 + ['two'],
      'k2': [1, 1, 2, 3, 3, 4, 4]})
```

العامود الثاني

بيانات (dictionary من Data Fram)

فيه عامودين k1 k2

داخل ال list

اذا ضربنا ب رقم 3 رح تتكرر

ليصير one, two " 3 " مرات

```
[50]: data
```

```
[50]:    k1  k2
      0  one  1
      1  two  1
      2  one  2
      3  two  3
      4  one  3
      5  two  4
      6  two  4
```

two

pd.Data Fram من numpy بتحول ال dicting لل array

للـ Data Fram "Indexes"

*الـ Data Transformation في (Data science) بشبه كتير شغلات

في منهم الدرس الـ (duplicated) الموجودة عندي

```
[51]: data.duplicated()
```

```
[51]: 0    False
      1    False
      2    False
      3    False
      4    False
      5    False
      6     True
      dtype: bool
```

بشوف اذا في

(two row duplicated)?!

اذا في two row مكررات رح نعتبرهم Noise

بشوف ال row اذا شافهن قبل اولا

two 4

two 4

```
[52]: data.drop_duplicates()
```

شافه من قبل .: True

```
[52]:    k1  k2
      0  one  1
      1  two  1
      2  one  2
      3  two  3
      4  one  3
      5  two  4
```

بترجع ال duplicat row

وبتحذفه

ورح تحذف آخر واحد

م(0-6)

```
[45]: data['v1'] = range(7)
```

ح اضيفها مباشرة ⇒ Column
كله الاشي موجود

```
[55]: data
```

[55]:

	k1	k2	v1
0	one	1	0
1	two	1	1
2	one	2	2
3	two	3	3
4	one	3	4
5	two	4	5
6	two	4	6

```
[56]: data.drop_duplicates(['k1'])
```

الـ duplicate لقدر احدد Colum معين ح تشيل
duplicate فقط داخل هذا الـ Colum (k1)
ح تخلي لبس one two (أول سطرين)

[56]:

	k1	k2	v1
0	one	1	0
1	two	1	1

```
[46]: data
```

كل هذه العمليات ما رح تعدل على data
لأنه ما فيا مساواة كلها عبارة عن (View)

[46]:

	k1	k2	v1
0	one	1	0
1	two	1	1
2	one	2	2
3	two	3	3
4	one	3	4
5	two	4	5
6	two	4	6

```
[58]: data.drop_duplicates(['k1', 'k2'], keep='last')
```

بده يشيك على K1, K2, Keep
لو الـ (5,6) مكررين
ح يحذف (5)؟ لأنه من
كلمة last خليلك خير

[58]:

	k1	k2	v1
0	one	1	0
1	two	1	1
2	one	2	2
3	two	3	3
4	one	3	4
6	two	4	6

بدي انشئ Data Fram من (dictinary)

```
[66]: #Transforming Data Using a Function or Mapping
data = pd.DataFrame({'food': ['bacon', 'pulled pork', 'bacon','Pastrami',␣
 ↪'corned beef', 'Bacon','pastrami', 'honey ham', 'nova lox'],
'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
```

```
[67]: data
```

4-housing-data-set-jupyter

March 21, 2024

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
```

[2]:
```
housing = pd.read_csv('housing.csv')
```

[ ]:
```
```

[ ]:
```
```

Target Value:

[3]:
```
housing
```

(الهدف من هذا دراستكم تحمل توقع لل

Feature

[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms \
0	-122.23	37.88	41.0	880.0	129.0
1	-122.22	37.86	21.0	7099.0	1106.0
2	-122.24	37.85	52.0	1467.0	190.0
3	-122.25	37.85	52.0	1274.0	235.0
4	-122.25	37.85	52.0	1627.0	280.0
…	…	…	…	…	…
20635	-121.09	39.48	25.0	1665.0	374.0
20636	-121.21	39.49	18.0	697.0	150.0
20637	-121.22	39.43	17.0	2254.0	485.0
20638	-121.32	39.43	18.0	1860.0	409.0
20639	-121.24	39.37	16.0	2785.0	616.0

توقع لسعر البيت)

لو درس مناطق وليس بيوت (أي بيتنا واحد مواصفات)

هذا الـ problem تكون

Prediction Analysis
OR
Regression Analysis

	population	households	median_income	median_house_value \
0	322.0	126.0	8.3252	452600.0
1	2401.0	1138.0	8.3014	358500.0
2	496.0	177.0	7.2574	352100.0
3	558.0	219.0	5.6431	341300.0
4	565.0	259.0	3.8462	342200.0
…	…	…	…	…
20635	845.0	330.0	1.5603	78100.0
20636	356.0	114.0	2.5568	77100.0
20637	1007.0	433.0	1.7000	92300.0
20638	741.0	349.0	1.8672	84700.0

[20638 لل 0 من row عندي]

```
20639     1387.0     530.0     2.3886     89400.0

      ocean_proximity
0            NEAR BAY
1            NEAR BAY
2            NEAR BAY
3            NEAR BAY
4            NEAR BAY
...               ...
20635          INLAND
20636          INLAND
20637          INLAND
20638          INLAND
20639          INLAND

[20640 rows x 10 columns]
```

يعني عندي [10 Feature]

[4]:
```
housing.info()
```

→ نوع الـ Feature + بتلخص كل الـ Feature

هل هي Float او Int
هل هي numeric
او "object" " String

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

20640 rows = 20640

→ recode ع يكون عليها نشتغل لأنه فيه not (Non Null) هنا

→ object هو string

[5]:
```
housing.shape
```

→ بتعطي الـ rows والـ Column

[5]: (20640, 10)

[6]:
```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
```

```
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

اسم ال Column او Feature

```
[7]: housing["ocean_proximity"].value_counts()
```

بدل ادرس ال Feature بليا هيا Category

```
[7]: <1H OCEAN     9136
     INLAND        6551
     NEAR OCEAN    2658
     NEAR BAY      2290
     ISLAND           5
     Name: ocean_proximity, dtype: int64
```

كل ال Category وكم واحد منهم تكرر

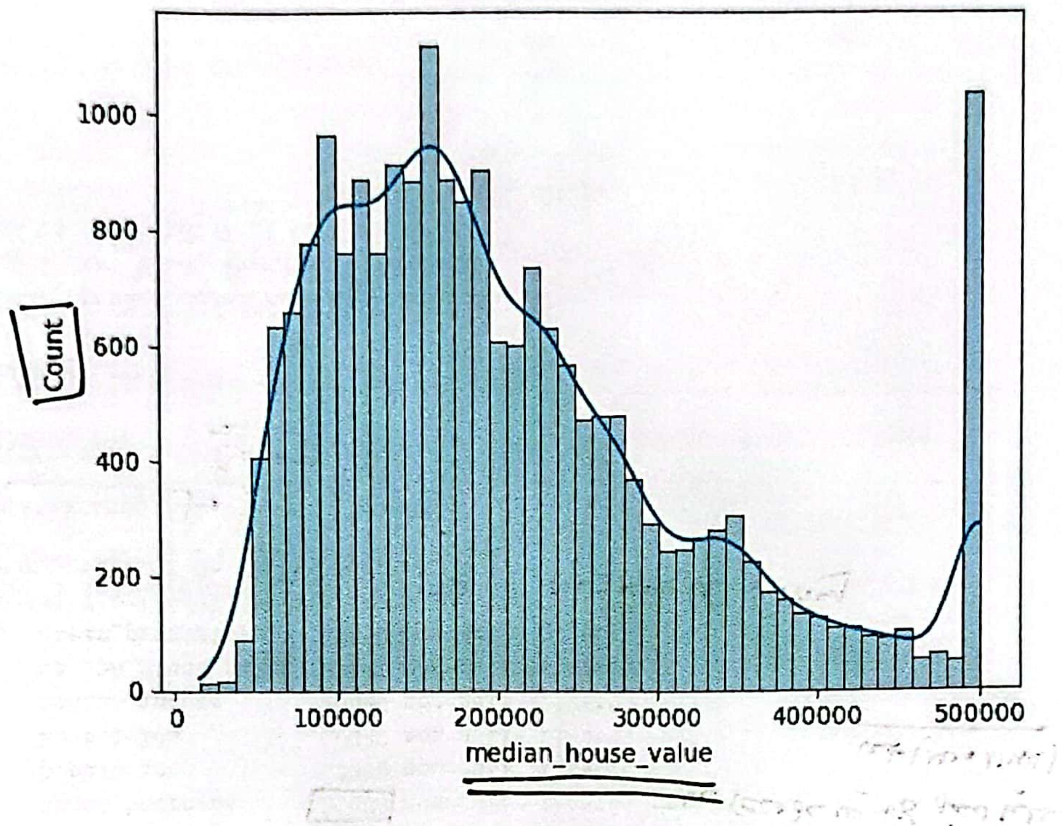
أقل جزء بيتكرر

[object بـ data type بليا ال] Category

مش int

```
[5]: import seaborn as sns
     sns.histplot(housing['median_house_value'], kde=True)
```

```
[5]: <AxesSubplot:xlabel='median_house_value', ylabel='Count'>
```



[ ]:

[6]: `housing.describe()`

[6]:

	longitude	latitude	housing_median_age	total_rooms \
count	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081
std	2.003532	2.135952	12.585558	2181.615252
min	-124.350000	32.540000	1.000000	2.000000
25%	-121.800000	33.930000	18.000000	1447.750000
50%	-118.490000	34.260000	29.000000	2127.000000
75%	-118.010000	37.710000	37.000000	3148.000000
max	-114.310000	41.950000	52.000000	39320.000000

اقل بيت لعمره

	total_bedrooms	population	households	median_income \
count	20433.000000	20640.000000	20640.000000	20640.000000
mean	537.870553	1425.476744	499.539680	3.870671
std	421.385070	1132.462122	382.329753	1.899822
min	1.000000	3.000000	1.000000	0.499900
25%	296.000000	787.000000	280.000000	2.563400
50%	435.000000	1166.000000	409.000000	3.534800

```
75%          647.000000   1725.000000    605.000000      4.743250
max         6445.000000  35682.000000   6082.000000     15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
```

قبل ال split كان عدد ال Data الأصلية 20640

[10]:
```
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=43)
```

[11]:
```
train_set.shape
```

عدد ال split

[11]: (16512, 10)

[12]:
```
test_set.shape
```

عدد ال split

[12]: (4128, 10)

[13]:
```
test_set?
```

[14]:
```
housing = train_set.copy()
housing.shape
```

[14]: (16512, 10)

[15]:
```
housing.info()
```

عدد ال record

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16512 entries, 72 to 14148
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           16512 non-null  float64
 1   latitude            16512 non-null  float64
 2   housing_median_age  16512 non-null  float64
 3   total_rooms         16512 non-null  float64
 4   total_bedrooms      16351 non-null  float64
 5   population          16512 non-null  float64
 6   households          16512 non-null  float64
 7   median_income       16512 non-null  float64
 8   median_house_value  16512 non-null  float64
 9   ocean_proximity     16512 non-null  object
```

```
dtypes: float64(9), object(1)
memory usage: 1.4+ MB
```

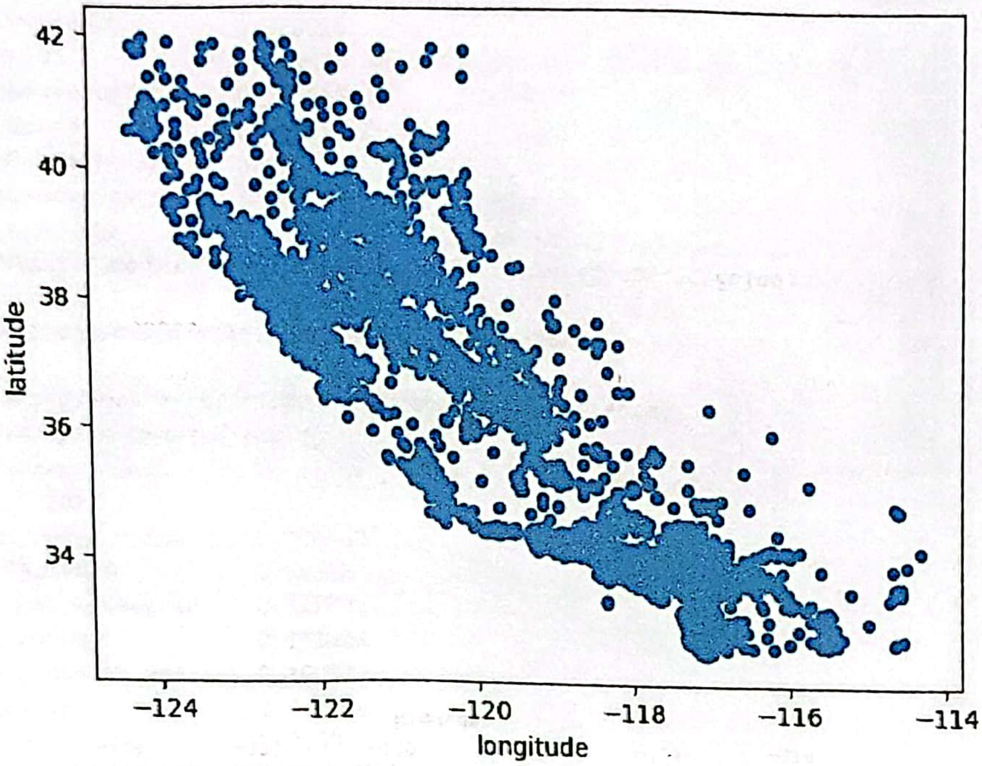
اشوف مين تجمع البيوت

```
[16]: housing.plot(kind="scatter", x="longitude", y="latitude")
```

خط الطول

خط العرض

```
[16]: <AxesSubplot:xlabel='longitude', ylabel='latitude'>
```

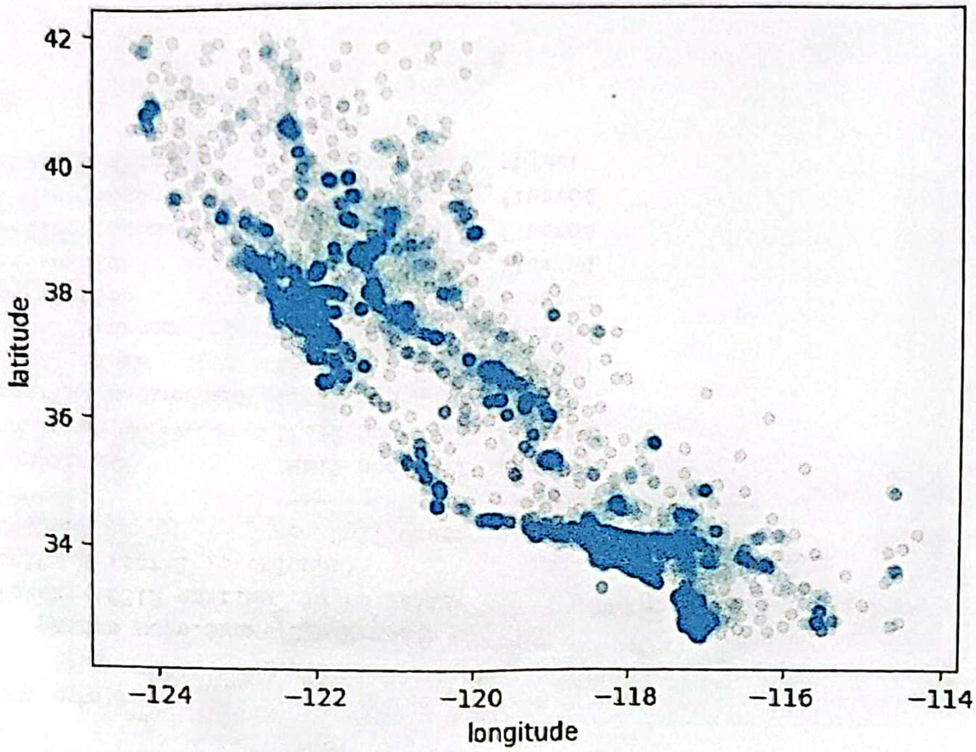



```
[17]: #A better visualization highlighting high-density areas
      housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)
```

```
[17]: <AxesSubplot:xlabel='longitude', ylabel='latitude'>
```

more highlight

(أوضح أكثر)



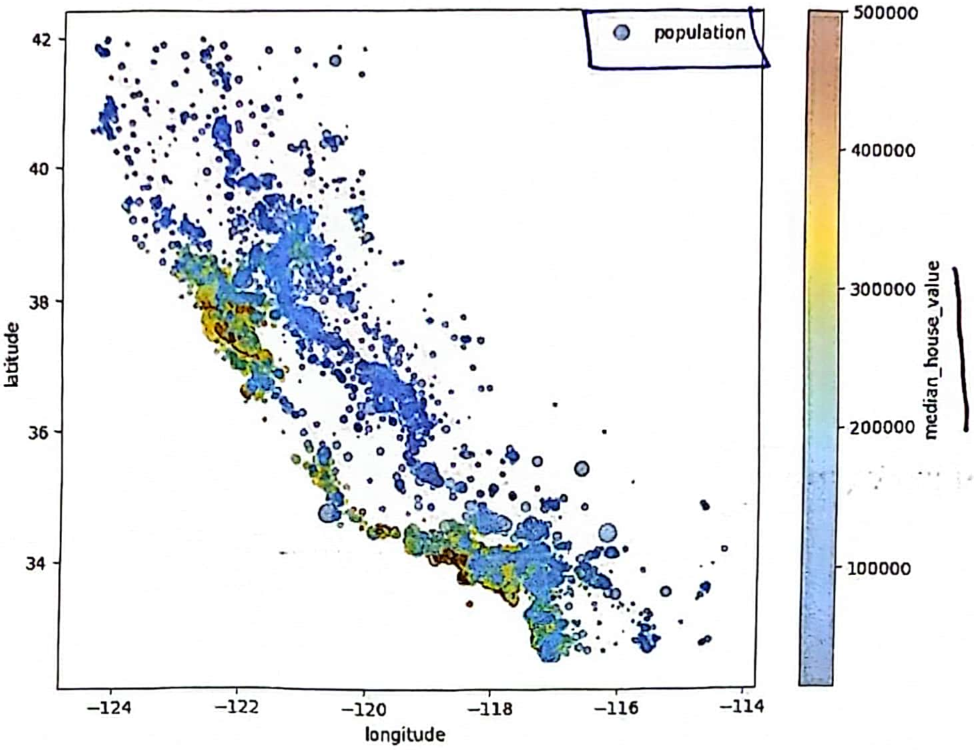
الطول والعرض

pmone din.

```
[18]: housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.5,
      s=housing["population"]/100, label="population", figsize=(9,7),
      c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True,
      )
      plt.legend()
```

كل ما زاد نصف القطر للدائرة يكون السكان أعلى

```
[18]: <matplotlib.legend.Legend at 0x23bf7dabdf0>
```



[19]:
```
housing.shape
```

[19]:
```
(16512, 10)
```

بترسم ال corr. بين ال features مع بعض

[20]:
```
corr_matrix = housing.corr()
corr_matrix.style.background_gradient(cmap='coolwarm')
```

⇐ (DataF . corr)

[20]:
```
<pandas.io.formats.style.Styler at 0x23bf9cfaf70>
```

[21]:
```
corr_matrix["median_house_value"].sort_values(ascending=False)
```

[21]:
```
median_house_value    1.000000
median_income         0.685886
total_rooms           0.136422
housing_median_age    0.104272
households            0.067724
total_bedrooms        0.053079
population           -0.024487
longitude            -0.043319
```

Handling Missing Data

Filtering Out Missing Data

[1] data = pd.DataFrame([[1., 6.5, 3.], [1., NA, NA],

....: [NA, NA, NA], [NA, 6.5, 3.]])

[2] cleaned = data.dropna()

```
In [21]: data
Out[21]:
     0    1    2
0  1.0  6.5  3.0
1  1.0  NaN  NaN
2  NaN  NaN  NaN
3  NaN  6.5  3.0
```

```
In [22]: cleaned
Out[22]:
     0    1    2
0  1.0  6.5  3.0
```


Handling Missing Data

Filtering Out Missing Data

- Drop rows if all values are NaN: data.dropna(how='all')
- Drop column: data.dropna(axis=1, how='all')

```
In [24]: data[4] = NA

In [25]: data
Out[25]:
     0    1    2   4
0  1.0  6.5  3.0 NaN
1  1.0  NaN  NaN NaN
2  NaN  NaN  NaN NaN
3  NaN  6.5  3.0 NaN
```

```
In [26]: data.dropna(axis=1, how='all')
Out[26]:
     0    1    2
0  1.0  6.5  3.0
1  1.0  NaN  NaN
2  NaN  NaN  NaN
3  NaN  6.5  3.0
```

Handling Missing Data

Filtering in Missing Data (fillna())

```
In [33]: #Filling In Missing Data
         df
Out[33]:
```

	0	1	2
0	0.271510	NaN	NaN
1	-0.473364	NaN	NaN
2	0.731081	NaN	0.761048
3	-1.536339	NaN	0.426161
4	0.778466	-0.319811	0.265054
5	-0.888870	0.259853	-0.429090
6	-0.482623	1.147385	0.468441

```
In [33]: df.fillna(0)
Out[33]:
          0         1         2
0 -0.204708  0.000000  0.000000
1 -0.555730  0.000000  0.000000
2  0.092908  0.000000  0.769023
3  1.246435  0.000000 -1.296221
4  0.274992  0.228913  1.352917
5  0.886429 -2.001637 -0.371843
6  1.669025 -0.438570 -0.539741
```


Handling Missing Data

Filtering in Missing Data

```
In [34]: df.fillna({1: 0.5, 2: 0})
Out[34]:
          0         1         2
0 -0.204708  0.500000  0.000000
1 -0.555730  0.500000  0.000000
2  0.092908  0.500000  0.769023
3  1.246435  0.500000 -1.296221
4  0.274992  0.228913  1.352917
5  0.886429 -2.001637 -0.371843
6  1.669025 -0.438570 -0.539741
```

df.fillna(0, inplace=True)

```
In [36]: df
Out[36]:
          0         1         2
0 -0.204708  0.000000  0.000000
1 -0.555730  0.000000  0.000000
2  0.092908  0.000000  0.769023
3  1.246435  0.000000 -1.296221
4  0.274992  0.228913  1.352917
5  0.886429 -2.001637 -0.371843
6  1.669025 -0.438570 -0.539741
```

Handling Missing Data

Filtering in Missing Data

	0	1	2
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614
2	0.523772	NaN	1.343810
3	-0.713544	NaN	-2.370232
4	-1.860761	NaN	NaN
5	-1.265934	NaN	NaN

(1) df.fillna(method='ffill')

	0	1	2
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614
2	0.523772	0.124121	1.343810
3	-0.713544	0.124121	-2.370232
4	-1.860761	0.124121	-2.370232
5	-1.265934	0.124121	-2.370232

(2) df.fillna(method='ffill', limit=2)

	0	1	2
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614
2	0.523772	0.124121	1.343810
3	-0.713544	0.124121	-2.370232
4	-1.860761	NaN	-2.370232
5	-1.265934	NaN	-2.370232

(3) data = pd.Series([1., NA, 3.5, NA, 7])

data.fillna(data.mean())

Data Transformation

- Removing Duplicates
- Transforming Data Using a Function or Mapping
- Replacing Values
- Detecting and Filtering Outliers

مثلاً Fun = 1,2,3 بدي اعمله
عادةً Fun. انه مثلاً normalization ثم
لا Data مبدئياً اعمل encoding
والحل removing duplicate ، جدول كامل
موجودين في mapping function عشان ما اخلي تكرره
انا بحط Function واحدة وطبقها على اكثر من (Data set)
فها داله Fun. انا بنشئ لحل اكثر من مشكلة
بدخل عليه اي Data set برجعلي اي (transform Data set)

Note: See Jupyter notebook file for more details

Handling Text and Categorical Attributes

[ما بتعامل مع String]

- Most Machine Learning algorithms prefer to work with numbers anyway.

يعني كل Category بتعطيها non String \ او م

- For this, we can use Scikit-Learn's Ordinal Encoder class:
- >>> from sklearn.preprocessing import OrdinalEncoder ← عرفته
- >>> ordinal_encoder = OrdinalEncoder() ← استخدمته

استخدمته

Handling Text and Categorical Attributes

- We have a categorical attribute *ocean_proximity*

```
>>> housing_cat = housing[["ocean_proximity"]]
>>> housing_cat.head(10)
      ocean_proximity
17606       <1H OCEAN
18632       <1H OCEAN
14650      NEAR OCEAN
3230           INLAND
3555        <1H OCEAN
19480          INLAND
8879        <1H OCEAN
13685          INLAND
4937        <1H OCEAN
4861        <1H OCEAN
```

[بطلع كل Category كم مرة تكررت]

```
In [101]: housing["ocean_proximity"].value_counts()
Out[101]: <1H OCEAN     9136
          INLAND        6551
          NEAR OCEAN    2658
          NEAR BAY      2290
          ISLAND           5
          Name: ocean_proximity, dtype: int64
```

⇒ جدول ال Category بديا اعملهم encoding عشان ما ضل يظهر عندي text بديا ليصيروا number

Handling Text and Categorical Attributes

- Ordinal encoding (converts each label into integer values)

كل label موجود بحوله الـ (int)

```
from sklearn.preprocessing import OrdinalEncoder
ordinal_encoder = OrdinalEncoder()
housing_cat_encoded = ordinal_encoder.fit_transform(housing_cat)
```

الـ Column يلي بدنا اياه encoding

(string)

number Column جديد لو جاي عنده كان فيه 5 Category ليس number

1 3 4 5

```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY',
'NEAR OCEAN'], dtype=object)]
```

2

```
array([0., 1., 2., 3., 4.])
```

One issue with this representation is that ML algorithms will assume that two nearby values are more similar than two distant values

يعني الـ (0) والـ (1) صاروا قريبين من بعض لأنه number ورا بعض

تابعين للـ [<1H] والـ [INLAND]، صار معناه انه هدول الـ (Category) نفسهم

الـ machin language الهم اقرب اشي على بعض

والـ (0) والـ (4) ابعد اشي عن بعض! لكن الـ (Dataset) لا

<1H : 0
Near oce. : 4

هدول الـ Category قريبات من بعض لبس الـ ordinal encoder اعطتهم ارقام بعيدة عن بعض وهاي مشكلة الـ (ordinal encoding)

[انه الارقام الهم علاقة مع بعض من الـ machin نفسهم]

Handling Text and Categorical Attributes

يعني الداتا الهم علاقة مع بعض

- Ordinal encoding: It is good if the data are ordinal)
  - (e.g., for ordered categories such as "bad", "average", "good", "excellent"),
  - (e.g. school grade: F, D, C, B, A), (Education level: BSc, MSc., PhD)

ما بزبط الحل Good 0 والـ bad 4

(لبس الحالة اذا بدي [ordinal enc] لازم اعطيه (manually) ارقام قريبة من بعض)

- It is obviously not the case for the **ocean_proximity** column

غلط لا لانهم الهم علاقة مع بعض

لكل attribut الـ لازم افهم الـ ما تعرف مين اقرب اي اشي عن بعض واعطيهم manually number

اذا استخدم الـ "ordinal encoding" Fun. كان يلي يعطي ارقام عشوائي

المطلوب ارقام ⇐ من عندي مثلاً اعطي الارقام اقرب اشي لبعض والبعيد العلاقات بينهم شكل منطقي

1 2 3 4

```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY',
'NEAR OCEAN'], dtype=object)]
```

5

```
array([0., 1., 2., 3., 4.])
```

(ordinal encoding) لبتجرب

- Colure column: [Green', 'Yellow', 'Black', White', 'Purple']

* الـ [ordinal encoding] لا يعني اذا كان الـ Category "not related"

← يعني عامود فيه انواع السيارات بلاطيه (ordinal) لا يدي ، مالهم علاقة ببعض

← اذا عامود فيه الـ سوات بعطيه ordinal عادي

اما اذا Category الهم علاقة مع بعض بعطيهم ارقام انسب !

Handling Text and Categorical Attributes

لاحل مشكلة ال encoding اذا مافي علاقة بستخدم ال [one-hot encoding]

- *One-hot encoding*:
  - Create one binary attribute per category
  - Each categorical values is replaced by one-hot vectors

اي Category

<1h ocean	1 0 0 0 0
Near Bay	0 1 0 0 0
near ocean	0 0 1 0 0
Island	0 0 0 1 0
Inland	0 0 0 0 1

- For the attribute: '<1H OCEAN'

```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY',
'NEAR OCEAN'], dtype=object)]
```

كل Category نرمز لها ، عددهم (5) نجعل Vector طوله (5)

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

[نستبدل ال one column ب (5) column]

كل column فيه 1 و 0 [ب ال Column الافضل [one hote encoding]

لان هاي الطريقة تعامل ال Category Indep. مافي اي ترتيب ثاني .

* اذا ال Category عبارة عن [not related] الافضل استعمل (One Hot)
اما اذا الها علاقة مع بعض نستعمل (ordinal encoding) لكن اذا قدرت استعملها (manually) بكون افضل

15

Handling Text and Categorical Attributes

- *One-hot encoding*

← بهاد المثال في بعض الداتا المرتبطة ببعض وبعضها مالها علاقة ببعض
فالافضل استخدمنا ال [one hot] مشان يتعامل كلهم مالهم علاقة ببعض

Before Encoding		After Encoding				
'<1H OCEAN'	→	1	0	0	0	0
'<1H OCEAN'	→	1	0	0	0	0
'NEAR BAY'	→	0	1	0	0	0
'NEAR BAY'	→	0	1	0	0	0
'NEAR OCEAN'],	→	0	0	0	0	1
'NEAR OCEAN'],	→	0	0	0	0	1
'<1H OCEAN'	→	1	0	0	0	0
IN land	→	0	0	1	0	0
Is land	→	0	0	0	1	0

16

Handling Text and Categorical Attributes

- Case Study: California Housing dataset
- Source Code: github
- Task: Data preparation and Visualization
- **Analysis**: Regression analysis for houses price predictions

Create a Test Set

→ Remove duplicate
→ transformation
→ encoding

- ① Split your dataset into train and ② test sets before you do any processing to the data

تقسيم الداتا مع الـ processing ام لعبد الـ processing ؟! نبدأ الـ (Data Science) بدل نقسم الداتا العديدنا على يلي بدل ايان

- It sound strange to voluntarily set aside part of the data at this stage!

السبب ؟ لما نقل test للـ model رح نجيب test Data اول مرة بنشوفها ، هاي الداتا مش معمول الها preprocessing

بـ ل تجربنا الـ model ، الصح اقسم الداتا مشان الـ test Data ما تكون عارفة اي اشي عن preprocessing ولا عن الـ model

- Look at the test set, you may select some seemingly interesting pattern in the test data that leads you to select a particular kind of Machine Learning model.

اذا الداتا كانت معمول لها

اما اذا عملت processing لعدينا وقسمت ، غالبا كان بتسرب معلومات للـ model وبطل عادل

كأنه الـ Data Set الـ test منه عارفة عن الـ model وعن الـ Category وعن كل شي

تأثر على اختيار الـ model

Data يلي ما شافها ، اول preprocessing

- Future unseen data will be processed in exact same way as the testing data, thus ensures consistency in model performance.

الـ test Data ما تعرف عن الـ model اشي .

Create a Test Set

- Splitting avoid Data leakage:

  ✓Accidental sharing of information between training and testing datasets

  (وصار الشي بدي امنعه) يعني معلومات عنا test Data م

  ✓Gives the model a 'heads-up' about the testing dataset and generate seemingly optimal evaluation scores

  ✓Since the model overfits the testing data, it cannot predict accurately on future unseen datasets

يعني الـ Model حقق نتائج عالية

للـ Data الرئيسية اما الـ New Data مثل !

بنمنعه عن طريق تقسيم الداتا قبل الـ pre-processing

19

Create a Test Set

- Creating test set: Pick some instances randomly,
- Typically 20% of the dataset (or less if your dataset is very large),

```
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(housing, test_size=0.2,
random_state=42)
```

size

Data frame

Data يلي فيه

((test set) السبب لتطلعلي مختلفة كل مرة لقسم فيها)

الـ 20% للـ test

20

معناه شو علاقة الـ Feature مع الـ target Varible

Looking for Correlations

```
corr_matrix = housing.corr()
```

بتحسب الـ Correla. مع كل الـ Feature مع بعض

target Value / Indep Varible / main Feature "يلي بدي ادرسها"

عشان ترتبها من الاعلى للاقل

```
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
median_house_value    1.000000
median_income         0.687170
total_rooms           0.135231
housing_median_age    0.114220
households            0.064702
total_bedrooms        0.047865
population           -0.026699
longitude            -0.047279
latitude             -0.142826
Name: median_house_value, dtype: float64
```

أول Feature مهمة

بندرس (regression Analysis) بدي التوقع سعر البيت

Looking for Correlations (Attribute Combinations)

new Column

الـ Correlation بيقيم عاليا

```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
```

```
median_house_value     1.000000
median_income          0.690647
rooms_per_household    0.158485
total_rooms            0.133989
housing_median_age     0.103706
households             0.063714
total_bedrooms         0.047980
population            -0.026032
longitude             -0.046349
latitude              -0.142983
```

ما كانت في Data Set (انا ضفتها)

new derive حصلت على أعلى Corre.

The new **rooms_per_household** attribute is much more correlated with the median house value than the total number of rooms or **households**!

Looking for Correlations (corr matrix)

[بتدرس العلاقة كل ال Feature مع بعض]

```
In [113]: corr_matrix.style.background_gradient(cmap='coolwarm')
Out[113]:
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
longitude	1.000000	-0.925418	-0.110523	0.044034	0.069503
latitude	-0.925418	1.000000	0.013790	-0.035624	-0.067868
housing_median_age	-0.110523	0.013790	1.000000	-0.361702	-0.321340
total_rooms	0.044034	-0.035624	-0.361702	1.000000	$\boxed{0.929059}$
total_bedrooms	0.069503	-0.067868	-0.321340	$\boxed{0.929059}$	1.000000
population	0.104090	-0.112972	-0.303651	0.863172	0.887092

* ال رقم بين two attr. في Data set [ال total bedrooms وال total rooms]

* اذا شفت ال two Feature (Higher Correlated) ممكن تستغني عن واحد منهم او تدمجهم في طريقة معينة صادفتي لكي يحصل ال accuracy عالي في النهاية

Replacing NaN values

حرفه لتبديل ال (non Value)

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")
```

* اذا عند ال عامود non Value بدل ما تحذفهم او تعبيهم ب zero او ميزه لبتحذم استراتيجية تسمى median بعبيهم ب متوسط القيم الموجودة في العامود

Feature Scaling

لسى تدخل Rang القيم قليل افضل للـ (machine learning)

الـ Feature Scaling لو اصلا اشي بحله مشكلة الـ outliers

- There are two common ways to get all attributes to have the same scale: *min-max scaling* (normalization) and *standardization* ()

[1]. **Normalization** "MinMaxScaler()" bound the result between (0,1).

بخلي الـ bound لـ Rang بصير Rang قليل + بقلل الـ standardization وبقلل من تأثير الـ outliers

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

بحول كل القيم منه للـ 1 لا حل مشكلة الـ outliers

- The cost of having this bounded range - in contrast to standardization - is that we will end up with smaller standard deviations, which can suppress the effect of outliers.

الـ Feature بلي فينا شي تميز منطقي لبعض (outliers) بحلها الـ Normalization

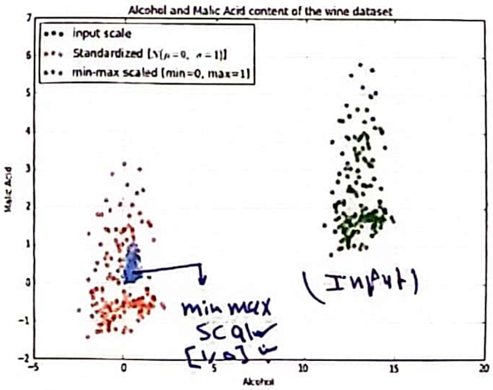
Feature Scaling

[2]. **Standardization** subtracts the mean value and then it divides by the standard deviation. It does not bound values to a specific range.

- Unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

* اذا بدي الغي تأثير الـ outliers تماما لبستخدم الـ normalization

* اما اذا ماكان عندك outliers بالبيانات طبيعية الافضل تستخدم الـ standardization



```
latitude              -0.146099
Name: median_house_value, dtype: float64
```

[22]:
```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
```

[23]:
```
housing["rooms_per_bedroom"] = housing["total_bedrooms"]/housing["total_rooms"]
```

[24]:
```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16512 entries, 72 to 14148
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   longitude            16512 non-null  float64
 1   latitude             16512 non-null  float64
 2   housing_median_age   16512 non-null  float64
 3   total_rooms          16512 non-null  float64
 4   total_bedrooms       16351 non-null  float64
 5   population           16512 non-null  float64
 6   households           16512 non-null  float64
 7   median_income        16512 non-null  float64
 8   median_house_value   16512 non-null  float64
 9   ocean_proximity      16512 non-null  object
 10  rooms_per_household  16512 non-null  float64
 11  rooms_per_bedroom    16351 non-null  float64
dtypes: float64(11), object(1)
memory usage: 1.6+ MB
```

[25]:
```
corr_matrix["median_house_value"].sort_values(ascending=False)
```

[25]:
```
median_house_value    1.000000
median_income         0.685886
total_rooms           0.136422
housing_median_age    0.104272
households            0.067724
total_bedrooms        0.053079
population           -0.024487
longitude            -0.043319
latitude             -0.146099
Name: median_house_value, dtype: float64
```

[35]:
```
corr_matrix = housing.corr()
corr_matrix
```

⇒ (يدرس علاقة الـ Feature مع بعض)

[35]:
```
                    longitude  latitude  housing_median_age  total_rooms  \
longitude            1.000000 -0.925418           -0.110523     0.044034
```

```
latitude            -0.925418  1.000000       0.013790    -0.035624
housing_median_age  -0.110523  0.013790       1.000000    -0.361702
total_rooms          0.044034 -0.035624      -0.361702     1.000000
total_bedrooms       0.069503 -0.067868      -0.321340     0.929059
population           0.104090 -0.112972      -0.303651     0.863172
households           0.057452 -0.073163      -0.304534     0.917526
median_income       -0.015301 -0.078997      -0.118781     0.200847

                    total_bedrooms  population  households  median_income
longitude                 0.069503    0.104090    0.057452      -0.015301
latitude                 -0.067868   -0.112972   -0.073163      -0.078997
housing_median_age       -0.321340   -0.303651   -0.304534      -0.118781
total_rooms               0.929059    0.863172    0.917526       0.200847
total_bedrooms            1.000000    0.887092    0.980707      -0.005755
population                0.887092    1.000000    0.914236       0.006393
households                0.980707    0.914236    1.000000       0.015438
median_income            -0.005755    0.006393    0.015438       1.000000
```

[27]:
```
# Tuesday lecture, March 28 starts here
corr_matrix = housing.corr()
corr_matrix["median_house_value"].sort_values(ascending=False)
```

⇒ بدرس بين target value وال Feature

[27]:
```
median_house_value    1.000000
median_income         0.685886
rooms_per_household   0.147782
total_rooms           0.136422
housing_median_age    0.104272
households            0.067724
total_bedrooms        0.053079
population           -0.024487
longitude            -0.043319
latitude             -0.146099
rooms_per_bedroom    -0.256558
Name: median_house_value, dtype: float64
```

(أهم Feature ال)

* دائماً ال Feature العلاقتها الضعيفة [Corr.] مع ال target value لازم نرجع فيها وما نحذفها

* اذا كان فيه [Corr.] عالي بين [two Indep. Feature] لازم اخليهم ما احذفهم ، ممكن دمج ال [higher corr.] يحسن نتائج ال Classif. وال prediction في ال model ، اذا بدي ادمج two Feat. لازم يكون الهم علاقة في بعض

[28]:
```
corr_matrix.style.background_gradient(cmap='coolwarm')
```

[28]: <pandas.io.formats.style.Styler at 0x23bf8a662e0>

[ ]:

→ [يعني بعمل drop ل Column]

[29]:
```
housing = train_set.drop("median_house_value", axis=1) # drop labels for
  ↪training set
housing_labels = train_set["median_house_value"].copy()
```

[30]:
```
housing_labels.shape
```

[نشيل ال target value من الداتا فريم كأنه بديا اياه ال Feature Inde. كالاول لل target]

ال train set رح اخلي زي ما هي مناكل ال set
اما ال housing نفس ال Data set شلنا منها ال Column يلي اسمها [median] ⇒

خزنته ماوال Column في [housing label]

تحل XX* Splitting قبل أي عملية (preprocessing)

row بطلع عدد ال [.shape ال] وال Column

```
[30]: (16512,)
```

[Column واحد فقط معناها انه] ⇒ [من train في Column طبعاً]

```
[31]: housing.isnull().sum()
```

[بطلع مجموع ال non value في كل Column]

```
[31]: longitude              0
      latitude               0
      housing_median_age     0
      total_rooms            0
      total_bedrooms       161
      population             0
      households             0
      median_income          0
      ocean_proximity        0
      dtype: int64
```

بدنا نحل مشكلة ال (non value)

```
[ ]:
```

```
[32]: #Remove the text attribute because median can only be calculated on numerical
      attributes:
      housing_num = housing.drop("ocean_proximity", axis=1)
```

تخزين

شلنا منه كل Feat. ما عدا ال Column. target + ocean proximity + columns number فيها

حاله بدي اشتغل ال number بدي اشيل ال ocean.

```
[40]: from sklearn.impute import SimpleImputer
      imputer = SimpleImputer(strategy="median")
      imputer.fit(housing_num)
      SimpleImputer(strategy='median')
      imputer.statistics_
```

(non value) بجيب ال median لكل Column. (اشرف ال)

* لحتى اعمل ال replace مع ال non value اعمله

مرة وحدة ومكانه كل ال Column فيهم non-value

(one Dim array)

تجيب ال number بـ non value [one Comm.] في

```
[40]: array([-118.51    ,   34.26    ,   29.     , 2127.     ,  435.     , 1166.     ,
        410.     ,    3.5348])
```

median value In First Column, Second Column ...

ما فيش اي Column نوع string

حالياً لا اقدر اعمل حاجة لأن في [String] Column

```
[41]: housing_num.median().values
```

```
[41]: array([-118.51    ,   34.26    ,   29.     , 2127.     ,  435.     , 1166.     ,
        410.     ,    3.5348])
```

بطلع ال median value من كل Column

```
[44]: X = imputer.transform(housing_num)
      X.shape
```

لحتى اعبي ال (Null Value) مش جيب لكن ما عبا لكن جوه 2-D ليعبي 6 عبا هم فيها

⇒ type X : ndarray

```
[44]: (16512, 8)
```

row → column

(Varible X)

```
[45]: X
```

تركنا ال housing ال Data Set يلي بنشتغل عليها

هنا ما درسنا الأعمدة label Column

عبارة عن two Dim. array

ليس ما فيها label

تبع (Data Fram)

```
[45]: array([[-122.29    ,   37.81    ,   49.     , ...,  560.     ,  152.     ,
         1.75     ],
       [-123.52    ,   41.01    ,   17.     , ...,  517.     ,  222.     ,
         2.1542],
       [-122.89    ,   40.76    ,   14.     , ...,  270.     ,   90.     ,
```


لكن ال (X) ما فيها اي (null value)

```
   2.3958],
...,
[-121.79   ,   37.03   ,   18.     , ...,  544.     ,  179.     ,
   3.934 ],
[-122.23   ,   39.86   ,   21.     , ...,  982.     ,  322.     ,
   1.8375],
[-117.08   ,   32.73   ,   36.     , ...,  619.     ,  233.     ,
   3.6125]])
```

[46]:
```
#The result is a plain NumPy array containing the transformed features. If you
want to
#put it back into a Pandas DataFrame, it's simple
housing_tr = pd.DataFrame(X, columns=housing_num.columns,
                          index=housing.index)
housing_tr
```

نقدر احول ال nparray لل Data Frame

= من خلال ال Columns تأخذ ال label هي ال

بحافظ على نفس ال (index)

جزء من Data Frame بعد ما طبقت عليه transformation كامل "null value" لعطي فيه null value

[46]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms \
72	-122.29	37.81	49.0	844.0	204.0
2602	-123.52	41.01	17.0	1564.0	345.0
19784	-122.89	40.76	14.0	712.0	131.0
3405	-118.36	34.26	34.0	3677.0	573.0
12492	-121.48	38.57	38.0	2809.0	805.0
...	...	...	...	...	...
18448	-121.81	37.23	16.0	1674.0	281.0
7985	-118.16	33.86	26.0	6607.0	1663.0
18687	-121.79	37.03	18.0	943.0	213.0
19776	-122.23	39.86	21.0	1730.0	350.0
14148	-117.08	32.73	36.0	1158.0	218.0

	population	households	median_income
72	560.0	152.0	1.7500
2602	517.0	222.0	2.1542
19784	270.0	90.0	2.3958
3405	1598.0	568.0	6.8380
12492	1243.0	785.0	1.8512
...	...	...	...
18448	850.0	254.0	5.3157
7985	4066.0	1558.0	2.5068
18687	544.0	179.0	3.9340
19776	982.0	322.0	1.8375
14148	619.0	233.0	3.6125

[16512 rows x 8 columns] => null Value ك تحويل لـ

[47]:
```
housing_tr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16512 entries, 72 to 14148
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           16512 non-null  float64
 1   latitude            16512 non-null  float64
 2   housing_median_age  16512 non-null  float64
 3   total_rooms         16512 non-null  float64
 4   total_bedrooms      16512 non-null  float64
 5   population          16512 non-null  float64
 6   households          16512 non-null  float64
 7   median_income       16512 non-null  float64
dtypes: float64(8)
memory usage: 1.1 MB
```

[48]:
```
housing_tr.isnull().sum()
```

[48]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
dtype: int64
```

null Value عدم ⇒ حلينا مشكلة ال null Value عن طريق ال (imputer)

Data set جزء من الأصلية أخذت Column واحد بس هو ocean.prox.

[49]:
```
housing_cat = housing[["ocean_proximity"]]
housing_cat.head(10)
```

String فيه [S cat.]

String Cat.

بدي اتعامل مع Cat. او (String) Column اللي فيه non-numeric data

[49]:
```
      ocean_proximity
72           NEAR BAY
2602           INLAND
19784          INLAND
3405        <1H OCEAN
12492          INLAND
18007       <1H OCEAN
4293        <1H OCEAN
10193       <1H OCEAN
18611      NEAR OCEAN
8382        <1H OCEAN
```

[57]:
```
housing_cat.shape
```

[57]:
```
(16512, 1)
```

row one column

منو [S cat.]

```
[51]: housing["ocean_proximity"].value_counts()
```

```
[51]: <1H OCEAN     7304
      INLAND        5250
      NEAR OCEAN    2101
      NEAR BAY      1853
      ISLAND           4
      Name: ocean_proximity, dtype: int64
```

"[Each الـ ما الهم علاقة مع بعض" one hot

"الهم علاقة مع بعض" ordinal

الفائدة الـ encoding

الـ ocean بالـ منو [S cat.] ح يتحول

للـ [S column] منو one hot

Convert or replace Column with vectors

لكل قيمة واحدة" واحد منهم (1) والباقي (0)"

Ex) <1H ⇒ 10000

```
[56]: #Thursday lecture, March 30
      from sklearn.preprocessing import OrdinalEncoder
      ordinal_encoder = OrdinalEncoder()
      housing_cat_encoded = ordinal_encoder.fit_transform(housing_cat)
      U= np.unique(housing_cat_encoded)
```

```
[53]: ordinal_encoder.categories_
```

```
[53]: [array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
             dtype=object)]
```

0 1 2 3 4

Sparse matrix

[جزء من preproc ما نتوقع]

```
[54]: U
```

```
[54]: array([0., 1., 2., 3., 4.])
```

```
[55]: from sklearn.preprocessing import OneHotEncoder

      cat_encoder = OneHotEncoder()
      housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
      housing_cat_1hot
```

هم Column الـ يلي بدي اعمله encoding

⇒ S column

باطول كان row:16512 Column:1

S: Column

النوع

```
[55]: <16512x5 sparse matrix of type '<class 'numpy.float64'>'
              with 16512 stored elements in Compressed Sparse Row format>
```

من كل Column سطر واحد منهم قيمته (1) والباقي (0)

لعمل الـ encoding لـ S column

```
[ ]: cat_encoder.categories_
```

```
[60]: housing
```

[60]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms \
72	-122.29	37.81	49.0	844.0	204.0
2602	-123.52	41.01	17.0	1564.0	345.0
19784	-122.89	40.76	14.0	712.0	131.0
3405	-118.36	34.26	34.0	3677.0	573.0
12492	-121.48	38.57	38.0	2809.0	805.0
…	…	…	…	…	…
18448	-121.81	37.23	16.0	1674.0	281.0
7985	-118.16	33.86	26.0	6607.0	1663.0
18687	-121.79	37.03	18.0	943.0	213.0

19776	-122.23	39.86	21.0	1730.0	350.0
14148	-117.08	32.73	36.0	1158.0	218.0

	population	households	median_income	ocean_proximity
72	560.0	152.0	1.7500	NEAR BAY
2602	517.0	222.0	2.1542	INLAND
19784	270.0	90.0	2.3958	INLAND
3405	1598.0	568.0	6.8380	<1H OCEAN
12492	1243.0	785.0	1.8512	INLAND
…	…	…	…	…
18448	850.0	254.0	5.3157	<1H OCEAN
7985	4066.0	1558.0	2.5068	<1H OCEAN
18687	544.0	179.0	3.9340	<1H OCEAN
19776	982.0	322.0	1.8375	INLAND
14148	619.0	233.0	3.6125	NEAR OCEAN

[16512 rows x 9 columns]

[نوع من انواع ال preproc. جوها]
[يحصر الداتا بين اعداد معينة]

[58]:
```
#Transformation Pipelines
# build a pipeline for preprocessing the numerical attributes:
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
num_pipeline = Pipeline([
('imputer', SimpleImputer(strategy="median")),
('std_scaler', StandardScaler()),
])
housing_num_tr = num_pipeline.fit_transform(housing_num)
```

بما انه عنا اكثر من preproc. بنستخدم شي اسمه pipe. عشان نطبق كل ال preproces. مع بعض

* ال pipe اذا طبقناها على train set بقدر اطبقه كمان على test set لما اجي اعمل testing

normalization بعد ما نحصر ال value بين [0,1] بحصر ال value Stand.

بطبق كل ال prep. يلي بدي اياهم ع كل column ال string

replaced non value + normalization

نستخدم ال Simpl. لنستبدل ال non value بال median

processed — اسم pip — اسم Data Fram

```
#Combine processed numerical attribute with encoded categoral value

from sklearn.compose import ColumnTransformer

num_attribs = list(housing_num)
cat_attribs = ["ocean_proximity"]
full_pipeline = ColumnTransformer([
("num", num_pipeline, num_attribs),
("cat", OneHotEncoder(), cat_attribs),
])
housing_prepared = full_pipeline.fit_transform(housing)
```

ال (String) لازم نعملها encoding بعدين نعمل normalization

بتطلع مين ال null attribu.

[number attru. فيها data fram]

[String attrib] column واحد

خزنت ال String في

عشان ادمج ال String مع number

بدي ادمج الداتا كلها مقسم فيه number والثاني String "onehot encoding"

لطبق

[59]:
```
num_attribs
```

[داتا ست فيها كل column طبقت عليها كل ال Analysis]

للشغل ال Analysis

[اختصرت كل الشغل يلي اشتغلت عليها مثل] number data مثل string data

[للشغل كل الشغل يلي اشتغلناه]

ل number Data : replace non value + Standarlization

ل String Data : encoding

* اذا اجاني داتا جديدة بطبق عليها نفس الكود عشان نقدر نعمل ال testing

* pre-proc. تكون ال machine model بالمقابل

```
[59]: ['longitude',
       'latitude',
       'housing_median_age',
       'total_rooms',
       'total_bedrooms',
       'population',
       'households',
       'median_income']
```

num-attrib. ⇒ تعطي column الاسماء ال الموجودين في Data fram "housing num" ⇒ ال null attribute مع اسماء ال column يلي بتحتوي number data

```
[61]: housing_prepared.shape
```

```
[61]: (16512, 13)
```

```
[62]: housing_prepared?
```

```
[63]: housing_tr_NN = pd.DataFrame(housing_prepared,␣
      ↪columns=num_attribs+['C1','c2','c3','c4','c5'])
```

```
[64]: housing_tr_NN
```

[64]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms \
0	-1.352219	1.015389	1.616949	-0.828008	-0.803175
1	-1.966357	2.514412	-0.923218	-0.495466	-0.463635
2	-1.651798	2.397301	-1.161359	-0.888974	-0.978965
3	0.610030	-0.647589	0.426246	0.480453	0.085408
4	-0.947786	1.371407	0.743766	0.079555	0.644083
…	…	…	…	…	…
16507	-1.112555	0.743692	-1.002598	-0.444661	-0.617753
16508	0.709890	-0.834967	-0.208796	1.833715	2.710219
16509	-1.102569	0.650003	-0.843838	-0.782284	-0.781503
16510	-1.322261	1.975701	-0.605697	-0.418797	-0.451595
16511	1.249133	-1.364309	0.585006	-0.682983	-0.769462

	population	households	median_income	C1	c2	c3	c4	c5
0	-0.792140	-0.920336	-1.119938	0.0	0.0	0.0	1.0	0.0
1	-0.831455	-0.735293	-0.905749	0.0	1.0	0.0	0.0	0.0
2	-1.057290	-1.084232	-0.777723	0.0	1.0	0.0	0.0	0.0
3	0.156915	0.179351	1.576241	1.0	0.0	0.0	0.0	0.0
4	-0.167665	0.752986	-1.066312	0.0	1.0	0.0	0.0	0.0
…	…	…	…	…	…	…	…	…
16507	-0.526990	-0.650701	0.769560	1.0	0.0	0.0	0.0	0.0
16508	2.413435	2.796394	-0.718903	1.0	0.0	0.0	0.0	0.0
16509	-0.806769	-0.848962	0.037384	1.0	0.0	0.0	0.0	0.0
16510	-0.406300	-0.470945	-1.073571	0.0	1.0	0.0	0.0	0.0
16511	-0.738195	-0.706214	-0.132982	0.0	0.0	0.0	0.0	1.0

[16512 rows x 13 columns]

Hashemite University
Computer Engineering
Data Science Course

Regression Analysis

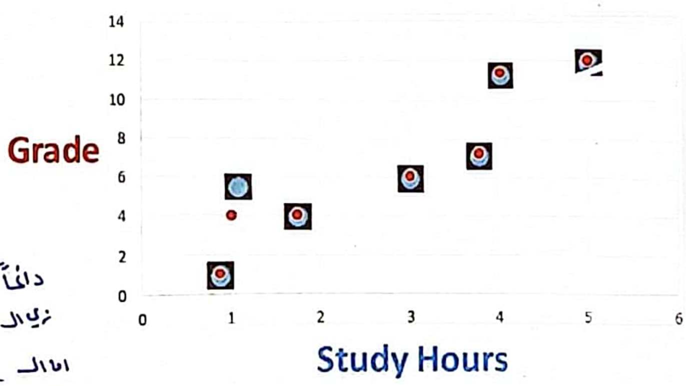
[prediction] لعمل Super. learning   لما نفع من القاعدة

Edited by Mohammad Al-hammouri

INTRODUCTION

* العلاقة بين Grade وال Study Hours تبين من خلال data set
بنرسم النقاط بعدين بنرسم ال (best fitting line) اللي بمر من خلال هاي النقاط

- Suppose we obtained grade scores for a group of students along with the study hours.
- A Grade (dependent variable) y-axis vs. Study hours (independent x-axis variable) " plot shows the data points.
  دائماً ال target value زي ال Grade بتكون dep.
- From this data set, we want to **predict** the grade of student who studies 2.5 hours?
  اما ال Feature بتكون Indep.

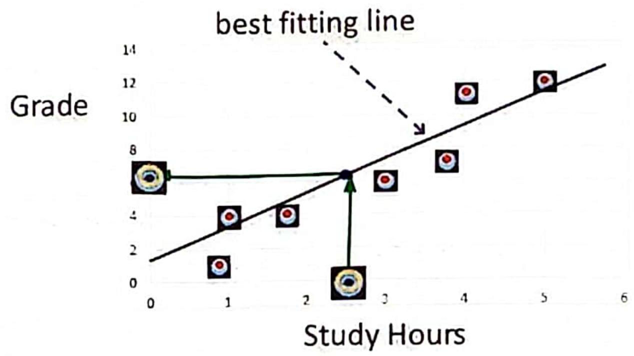


∘ [predict] لعمل عبارة عن regression Analysis
"بتجسب النقاط"
⇒ كل ما نرسم regression line أو best fitting line بربط النقاط
"عن طريق التجربة" لنعطي اقل error
⇒ لعديا قيمة اي Input لل Study Hours بعلم عد x-axis وبعدل النقاط بتكون ال predicting Grade.

INTRODUCTION

To predict grade for a student who studies 2.5 hours:

1. Perform regression (i.e. best fitting line) of the student data.
2. We can **predict** the grade of the student who studies 2.5 hours.
   Predicted grade=6.3



DEFINITIONS

ب regression Analysis لبس المتغير one target
one y-axis

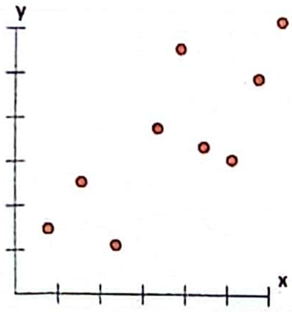
- Regression analysis [is estimating the relationships between a dependent variable and one or more independent variables]
  - y-axis • The dependent variable (e.g. y) is often called the target, outcome or response
  - Independent variables (x_0, x_1, ..x_n) are often called predictors or features
- Regression types
  - Linear regression: [computes the line that most closely fits the data; the model (e.g. line equation) is linear in the parameters
    عادة الـ model بكون (linear model)
    - Simple linear regression: one independent variable
    - Multiple linear regression: multiple independent variables
  - Non-linear regression (logistic): the model is non-linear in the parameters

تشير تصنف حسب value معين
"prediction" regression لل مدلعب classification لل يستخدم

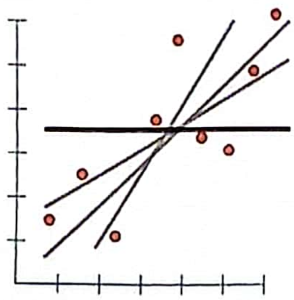
BEST LINE FITTING: BASICS

slop   intercept

- Line equation: $y = ax + b$
- The best fit → [optimum value of "a" and "b" which reduces a cost function (e.g. sum of squared errors)



Data Points to be fitted



We try few lines, but which one is the best fit?

(RMSE)

* بوسم (line) 6 بكيا ارسم
قدرش كل لنقطة لجيدة عن
الـ (line) واحسب الـ error
عن طريق Function
[الـ Actual القيمة الموجودة من
Dataset 6 الـ predicted
القيمة بلي بجد ما امر الـ record
كامل عند الـ model يلي بنيته
شو بتوقع توقع] الفرق بينهم
هو الـ error !

difference between Actual . Predicted Value

FITTING A LINE USING LEAST SQUARES

- One method is to select the line with least squared errors.
- The squared error for point (x_i,y_i) with respect to line $y=ax+b$:

squared error = [line value at x_i - y_i $]^2$

$$= [\,(a\,x_i - b) \quad - \quad y_i\,]^2$$

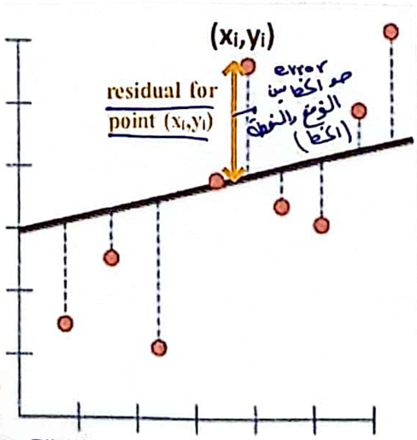
القيمة عند الـ line : Predict / والنقطة هي الـ Actual

- The "Sum of squared errors" is the sum of squared errors of all data points.

بدي احاول افضل اعمل تغيير في line لحتى احصل على

- The best fit is the line that has the least squares (i.e. minimum sum of squared errors)

square error

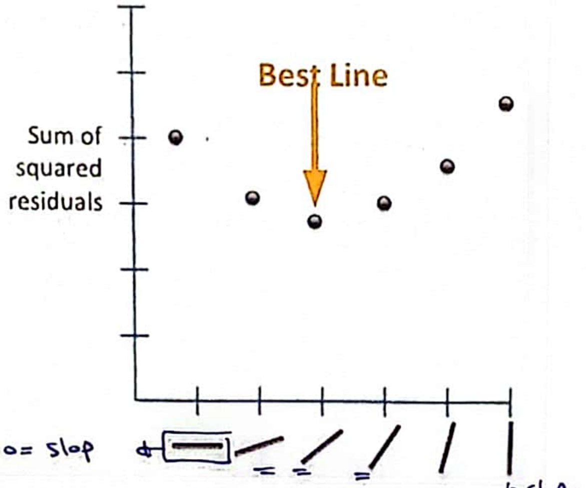


* كل الـ Algo. بترجع للـ (math) الـ (math) هو اساس كل تشغل الـ (machine learning)
الهدف انه اقلل من الـ error 6 تقليل من الـ error اذا ارسم (the best Fitting line) 6 بعرف انه 6 best انه بعطي
اقل error 6 بعدل هو الـ (line) عن طريق تغير الـ Varible [a,b] 6 بحال كان (Simple linear)
* كل ما زاد الـ (Indep. Varible) بصير الـ regressi. [more complex]

* لما تكون قيمة ال a, b كانت صح
نثبت ال model تالك
ال model صالح لوجود ال a, b

THE BEST FIT

- How to find optimum fit: $ax+b$
  - Trying different values of slope(a) and intercepts (b).
    - Suitable for large data set, using programs and computers.
    - In the diagram, different values of slope (a) have different values of sum of squares.
    - The best slope is the one produces least squares
  - Analytical methods based on reducing sum of square errors. We present two methods:



Lines with different "a" and "b"

* الهدف اخذ ابزنيا a وال b كحتى احصل (best Fit line)

REGRESSION FUNCTION

Any equation, that is a function of the dependent variables and a set of weights is called *a regression function*.

$y = a_1 x + a_0$

ال y (المتغير ال) Indep. Varible (x) مضروبين weight a_0 a_1

REGRESSION FUNCTION

Vector " Feature "

$(y = f(x, w))$

"y" is the dependent variable (temperature), "x" are the independent variables (humidity, pressure etc) and "w" are the weights of the equation (co-efficients of x terms).

more Indep. Varible

For example, the equation can be

$y = 0.5\,x1 + \underbrace{2.15}_{w}\,x2 + \underbrace{0.76}_{w}\,x3$

humidity pressure Hight Sea

Weight x (Indep.)

ال (w) الي بنضربهم الـ Indep Varible

HOW TO DETERMINE THE REGRESSION EQUATION

- We are doing this to make predictions going forward
- To learn the regression equation, we need to have some true data collected from the field

Indep. Varibles

Height above sea level in meters	Pressure in hpa	Humidity in percentage	Temperature in Fahrenheit (dep. Varible)
0	1015	67	84
500	1000	60	73
700	850	40	65

Example data for regression

HOW TO DETERMINE THE REGRESSION EQUATION?

- Let me denote "temperature" as **y** (dep.) and rest of the features (height, pressure, humidity) as **x**. We define the error function as sum of $(y-x*w)^2$

  vector — vector

  Feature — Actual Value — predicted Value

  Sum of error =

- We need to find out the **w** vector that minimizes the above function.

  [error]

- When is the function mathematically minimum? It is where the slope of the equation is 0

  slope of 0

  (error)

GRADIENT DESCENT

- Gradient Descent is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems.

- The general idea of Gradient Descent is to tweak parameters iteratively in order to (minimize a cost function.)

  Feature

  [Squared error]

GRADIENT DESCENT (COST FUNCTION)

target للـ (Value Actual)
او الـ End varible

$$f(m,b) = \frac{1}{N}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

element لعدد الـ

(Actual value) - (Predicted value)

* هون دائماً الـ predicted value نكون قريبة جداً من الـ model بطرحهم من بعض بطلع الـ error

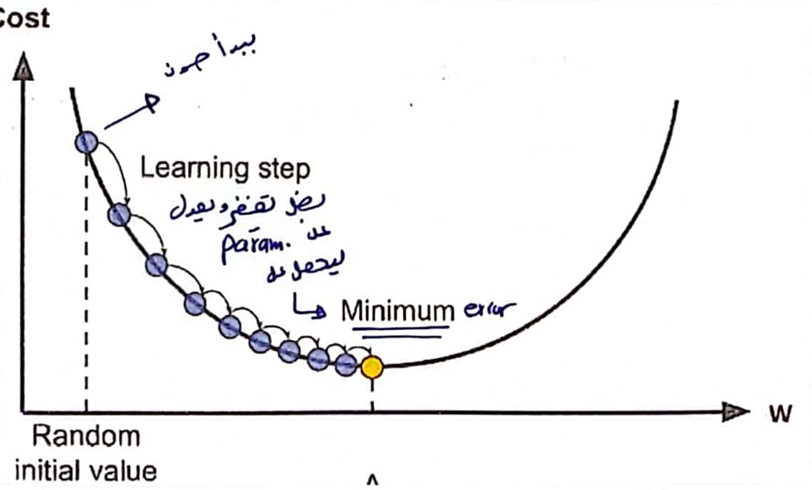
13

GRADIENT DESCENT

لكل مرة تقفز بناءً على learning rate

اذا كانت صغيرة كثير الـ update للـ parameter بكون قليل جداً الـ بالنتيجة تكون ادق

اما الحالة لتعديل كبير كثير يعني بسرعة احصل الـ update للنتيجة تصير garbage



We start with initial value of w and then improve it gradually

Vector of weight

14

بعد ما نحل المعادلة فوق الـ [m, b] هم الـ W
بدي اوجد افضل القيم للحصول على (min error)

GRADIENT DESCENT

ال w ممكن تكون اكثر من قيمة

حسب عدد ال (X) بكون عدد ال w

$w_0 \to x_0$
$w_1 \to x_1$
$w_2 \to x_2$

Updating the weight w:

$$\underset{new}{w} = \underset{old}{w} - alpha * delta$$

w: is the **coefficient** or weight being optimized,

alpha: learning rate (e.g. 0.1)

\ (learning rate) لحد اي optimization problem انت بتحدد ال

delta: is the error for the model on the training data attributed to the weight.

SIMPLE LINEAR REGRESSION WITH STOCHASTIC GRADIENT DESCENT (EXAMPLE)

$$\boxed{y = \underset{weight}{B0} + \underset{weight}{B1}\, x}$$

وحدة بس x ازون

⇒ one Ind. Varible
one dep. Varible

Consider the function above, what is the best value for the weight **w** (coefficients B0 and B1) which minimizing the error on a training dataset.

Assume initial value of 0.0 for both coefficients.

x	y
1	1
2	3
4	3
3	2
5	5

Solution: please check "Gradiant_Descent_Example.pdf"

Chapter 12

Linear Regression Tutorial Using Gradient Descent

Stochastic Gradient Descent is an important and widely used algorithm in machine learning. In this chapter you will discover how to use Stochastic Gradient Descent to learn the coefficients for a simple linear regression model by minimizing the error on a training dataset. After reading this chapter you will know:

- How stochastic gradient descent can be used to search for the coefficients of a regression model.
- How repeated iterations of gradient descent can create an accurate regression model.

Let's get started.

12.1 Tutorial Data Set

The dataset is the same as that used in the previous chapter on Simple Linear Regression. It is listed again for completeness.

```
x  y
1  1
2  3
4  3
3  2
5  5
```

Listing 12.1: Tutorial Data Set.

12.2 Stochastic Gradient Descent

Gradient Descent is the process of minimizing a function by following the gradients of the cost function. This involves knowing the form of the cost as well as the derivative so that from a given point you know the gradient and can move in that direction, e.g. downhill towards the minimum value. In machine learning we can use a technique that evaluates and update the coefficients every iteration called stochastic gradient descent to minimize the error of a model on our training data.

The way this optimization algorithm works is that each training instance is shown to the model one at a time. The model makes a prediction for a training instance, the error is calculated and the model is updated in order to reduce the error for the next prediction. This procedure can be used to find the set of coefficients in a model that result in the smallest error for the model on the training data. Each iteration the coefficients, called weights (w) in machine learning language are updated using the equation:

$$w = w - alpha \times delta \tag{12.1}$$

Where w is the coefficient or weight being optimized, $alpha$ is a learning rate that you must configure (e.g. 0.1) and gradient is the error for the model on the training data attributed to the weight.

delta

12.3 Simple Linear Regression with Stochastic Gradient Descent

The coefficients used in simple linear regression can be found using stochastic gradient descent. Stochastic gradient descent is not used to calculate the coefficients for linear regression in practice unless the dataset prevents traditional Ordinary Least Squares being used (e.g. a very large dataset). Nevertheless, linear regression does provide a useful exercise for practicing stochastic gradient descent which is an important algorithm used for minimizing cost functions by machine learning algorithms. As stated in the previous chapter, our linear regression model is defined as follows:

* لوكان عندي $x_1\ x_2\ y$

رح تكون المعادلة

$$y = B0 + B1 \times x \tag{12.2}$$

$y = B_0 + B_1 X_1 + B_2 X_2$

ليصير عندي (3) Varible لعقد للعالم

12.3.1 Gradient Descent Iteration #1

Let's start with values of 0.0 for both coefficients.

Data بحسب الجداول [x=1, y=1]

نفرضهن في البداية دائماً

x	y
1	1
2	3
4	3
3	2
5	5

$$\begin{gathered} B0 = 0.0 \\ B1 = 0.0 \\ y = 0.0 + 0.0 \times x \end{gathered} \tag{12.3}$$

$= 0$

We can calculate the error for a prediction as follows:

predicted y

Actual y

$$error = p(i) - y(i) \tag{12.4}$$

$= 0 - 1 = -1$

Where $p(i)$ is the prediction for the i'th instance in our dataset and $y(i)$ is the i'th output variable for the instance in the dataset. We can now calculate he predicted value for y using our starting point coefficients for the first training instance: $x = 1, y = 1$.

$$\begin{gathered} p(i) = 0.0 + 0.0 \times 1 \\ p(i) = 0 \end{gathered} \tag{12.5}$$

Using the predicted output, we can calculate our error:

$$error = (0 - 1)$$
$$\boxed{error = -1} \tag{12.6}$$

We can now use this error in our equation for gradient descent to update the weights. We will start with updating the intercept first, because it is easier. We can say that $B0$ is accountable for all of the error. This is to say that updating the weight will use just the error as the gradient. We can calculate the update for the $B0$ coefficient as follows:

[weight بنعمل update للـ] → w = w - alpha * delta

B0 [بعد ١ update بالأول للـ B0]

$$\underbrace{B0(t+1)}_{new} = \underbrace{B0(t)}_{old\ [0]} - alpha \times error \tag{12.7}$$

Where $B0(t + 1)$ is the updated version of the coefficient we will use on the next training instance, $B0(t)$ is the current value for $B0$, $alpha$ is our learning rate and error is the error we calculate for the training instance. Let's use a small learning rate of 0.01 and plug the values into the equation to work out what the new and slightly optimized value of $B0$ will be:

$$\overset{new}{B0(t+1)} = 0.0 - 0.01 \times -1.0$$
$$new\ [\,B0(t+1) = 0.01\,] \tag{12.8}$$

Now, let's look at updating the value for $B1$. We use the same equation with one small change. The error is filtered by the input that caused it. We can update $B1$ using the equation:

x = 1 , y = 1

$$\underbrace{B1(t+1)}_{new} = \underbrace{B1(t)}_{old} - alpha \times error \times x \quad (\text{أول اشي في Dataset}) \tag{12.9}$$

Where $B1(t + 1)$ is the update coefficient, $B1(t)$ is the current version of the coefficient, $alpha$ is the same learning rate described above, error is the same error calculated above and x is the input value. We can plug in our numbers into the equation and calculate the updated value for $B1$:

[لضلنا نكرر هاي العملية ٤ كمان مرة]

$$B1(t+1) = 0.0 - 0.01 \times -1 \times 1$$
$$\boxed{B1(t+1) = 0.01} \tag{12.10}$$

We have just finished the first iteration of gradient descent and we have updated our weights to be $B0 = 0.01$ and $B1 = 0.01$. This process must be repeated for the remaining 4 instances from our dataset. One pass through the training dataset is called an epoch.

(5 rows) هنا data set

12.3.2 Gradient Descent Iteration #20

لازم أعملها repeat لبعد ما أطلع أشوف شو آخر value للـ B1 و B0 وعلى أساسها الـ new value بحسب model أو ألجي عليه ولبعد منه بعمل prediction

Let's jump ahead. You can repeat this process another 19 times. This is 4 complete epochs of the training data being exposed to the model and updating the coefficients. Here is a list of all of the values for the coefficients over the 20 iterations that you should see:

B0	B1
0.01	0.01
0.0397	0.0694
0.066527	0.176708
0.08056049	0.21880847
0.118814462	0.410078328
0.123525534	0.4147894
0.14399449	0.455727313
0.154325453	0.497051164

أول إشي بنعملوا [the First epoch]

الـ (5) iteration (epoch) يعني لما أعمل مرة بـ (4) مرات زي كأني عامل (20) iteration، 4 × 5 = 20

* لما أعمل (20 iteration) زي كأني عامل (4 epochs) ٤ مرات يعني الـ (5)

الـ total بيكون هاد (20) من الداتا ولكل مرة بعمل update للـ B0 والـ B1

[الـ Second epoch بدي ارجع أكرر ١ كمان مرة ثانية ٤ ... الخ]

```
0.157870663   0.507686795
0.180907617   0.622871563
0.182869825   0.624833772
0.198544452   0.656183024
0.200311686   0.663251962
0.19841101    0.657549935
0.213549404   0.733241901
0.21408149    0.733773988
0.227265196   0.760141398
0.224586888   0.749428167
0.219858174   0.735242025
0.230897491   0.79043861
```

بعد 20\1 مرة
لكود وصلت انه
ا B_0 افضل
قيمة الها هي
0.23
مال $B_1 = 0.79$
افضل قيمة

Listing 12.2: Simple linear regression coefficients after 20 iterations.

I think that 20 iterations or 4 epochs is a nice round number and a good place to stop. You could keep going if you wanted. Your values should match closely, but may have minor differences due to different spreadsheet programs and different precisions. You can plug each pair of coefficients back into the simple linear regression equation. This is useful because we can calculate a prediction for each training instance and in turn calculate the error.

Below is a plot of the error for each set of coefficients as the learning process unfolded. This is a useful graph as it shows us that error was decreasing with each iteration and starting to bounce around a bit towards the end.

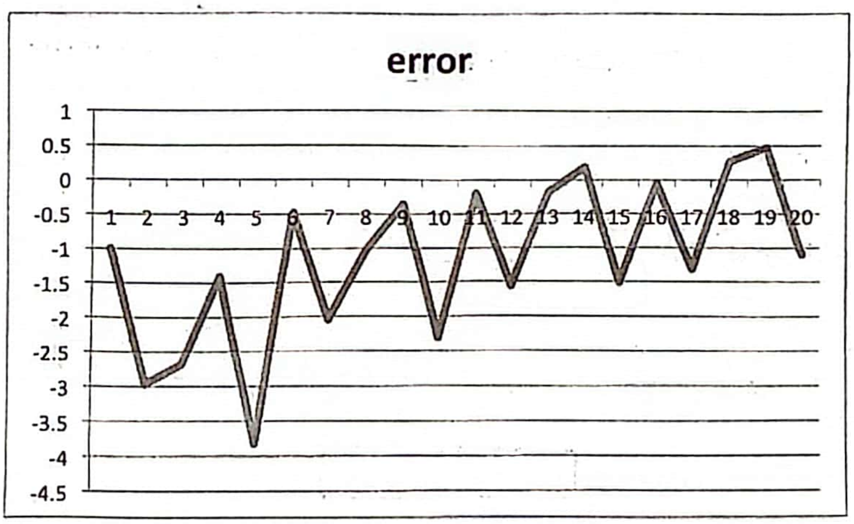


Figure 12.1: Simple Linear Regression Performance Versus Iteration.

You can see that our final coefficients have the values $B0 = 0.230897491$ and $B1 = 0.79043861$. Let's plug them into our simple linear Regression model and make a prediction for each point in our training dataset.

```
x  Prediction
1  1.021336101
2  1.811774711
4  3.392651932
3  2.602213322
```

```
5   4.183090542
```

Listing 12.3: Simple linear regression predictions for the training dataset.

We can plot our dataset again with these predictions overlaid (x vs y and x vs *prediction*). Drawing a line through the 5 predictions gives us an idea of how well the model fits the training data.

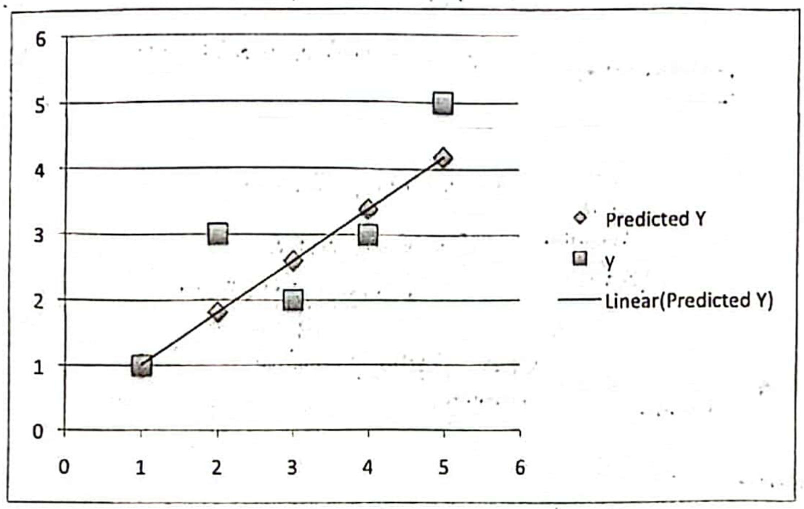


Figure 12.2: Simple Linear Regression Predictions.

We can calculate the RMSE for these predictions as we did in the previous chapter. The result comes out to be $RMSE = 0.720626401$.

12.4 Summary

In this chapter you discovered the simple linear regression model and how to train it using stochastic gradient descent. You learned:

- How to work through the application of the update rule for gradient descent.
- How to make predictions using a learned linear regression model.

You now know how to implement linear regression using stochastic gradient descent. In the next chapter you will discover the logistic regression algorithm for binary classification.

X	y
1	1
2	3
4	3
3	2
5	5

First → (1, 1)
Second → (2, 3)

$$[\; y = B_0 + B_1 * X \;]$$

X_1	X_2	y

لو كان two Ind. one dep. Variable

$$y = B_0 + B_1X + B_2X_2$$ بيصير

راح يجيب B_0, B_1, B_2

1 First iteration $\boxed{X=1}$

$B_0 = 0$ / $B_1 = 0$

$\hat{y} = 0 + 0 \times 1 = 0$ ($X=1$)

error $= \hat{y} - y = \boxed{0 - 1 = -1}$ ($y=1$)

→ learning rate $= 0.01$

$$B_{0\,new} = B_{0\,old} - \text{alpha} * \text{error}$$
$$= 0 - 0.01 * -1 = \boxed{0.01 \text{ (new } B_0)}$$

$$B_{1\,new} = B_{1\,old} - \text{alph} * \text{error} * X$$ → كونه في المعادلة ال $B_1 * X$

بعد ال iteration $= 0 - 0.01 * -1 * 1 = \boxed{0.01}$

اما ال B_0 مش مضروب ب (X) في المعادلة

2 Second iteration $\boxed{X=2}$

$B_0 = 0.01$, $B_1 = 0.01$

$\hat{y} = B_0 + B_1 * X$
$= 0.01 + 0.01 * 2$ (error)
⋮

$\boxed{\text{error} = -x}$

$B_{0\,new} = B_{0\,old} - \text{alpha} \times \text{error}$
$= 0.01 - 0.01 \times -1 = 0.02$

$B_{1\,new} = B_{1\,old} - \text{alpha} \times \text{error} \times X$

بكرر العملية (20) iteration → بعد 20 iteration → $B_0 = 0.23$, $B_1 = 0.79$

كل ما زدت عدد ال iteration بتحصل على نتيجة أدق
كل ما كان عدد الداتا اكثر بتحصل على نتيجة أدق

الآن بدي اجرا عمل (prediction) بناءً على آخر B_1, B_0

$y = 0.23 + 0.79 \times 1 = 1.02$ ($X=1$)
$\boxed{X=2} \Rightarrow 1.81$
⋮

X	prediction	y (Actual)
1	1.02	1
2	1.81	3
4	3.39	3
3	2.6	2
5	4.18	5

$$\Rightarrow RMSE(X, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h(x^{(i)}) - y^{(i)}\right)^2}$$

$$\text{error} = \sqrt{\frac{\left((1 - 1.0213) + (1.8 - 3) + \dots\right)^2}{5 \text{ (\# of row)}}}$$

1 = Actual, 1.0213 = prediction

(عشان احصل اقل error في النهاية)

Data Science

Logistic Regression

Prepared by:
Dr. Mohammad Alhammouri

Introduction

يعني (prediction)

- Logistic (regression) is one of the most popular machine learning algorithms [for binary classification.]

؟ كأنه لعطينا value معين (0) ولا (1)

- It is a simple algorithm that performs very well on a wide range of problems.

output > 0.5 ⇒ 1
output < 0.5 ⇒ 0
منفصله بعمل الـ classification (two class)

We will learn:

- How to calculate the logistic function.
- How to learn the coefficients for a logistic regression model using stochastic gradient descent.
- How to make predictions using a logistic regression model.

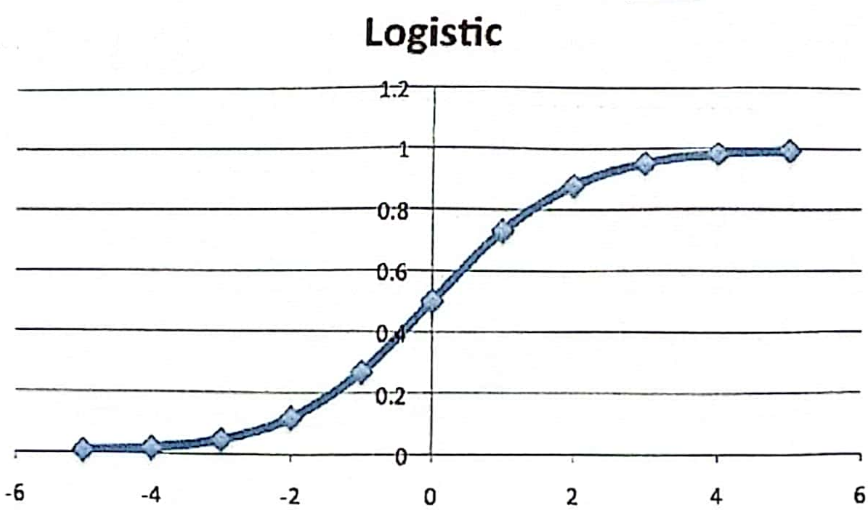
Logistic Function

$$\left[\frac{1}{1 + e^{-value}} \right]$$

- Can take any real-valued number and map it into a value between 0 and 1,
- How to learn the coefficients for a logistic regression model [using stochastic gradient descent.] [coefficient لل tweaking منعمل
  (min error) اعيد تعليم كده لحد ما اوصل لل]
- How to make predictions using a logistic regression model.

Logistic Function

- Below is a plot of the numbers between -5 and 5 transformed into the range [0 and 1] y-value
  between



Making Predictions with Logistic Regression

- Logistic Regression is used for classification
- Let's say we have a model that can predict whether a person is male or female based on their height.

Ex)

- **Prediction** = 0 (male) if **y**(height) < 0.5
- **Prediction** = 1 (fmale) if **y**(height) >= 0.5

← اذا ال y(height) اقل من (0,5)
لعتبره انه (class 0) المطلوب male

← اما ال y(hight) ≥ 0,5 اكبر من (0,5) لعتبره انه (class 1)
المطلوب Fmale

class A
class B
واحد
ال logistic Regression بعطيني 0 وال (1)

Prepare Data for Logistic Regression

y: 0,2 / 0,3 / 0,5 … ← فهم متعلقة لعمليها preprocessing

ما تقدر استخدام ال (logistic Regression) لانه ال (target value) لازم تكون (binary) اما (0) (1) F/T يعني (two class)

① → 1. **Binary Output Variable** — logistic regression is intended for binary (two-class) classification problems.

يعني اي قيمة اكبر من 0,5 لعتبره (1) / اي قيمة اقل من 0,5 (0) اعتباره، حتى يحولها (binary class)

$y \to \hat{y}$
0,2 → 0
0,3 → 0
0,5 → 1

شاذ عن ال value التانين → outlier (normalization (نطبق) ← بدوها بين 0 – 1)

[2]. **Remove Noise** — Logistic regression assumes no error in the output variable (y)

((no is normaliy destribution) (ليبن العلاقة بين (x, y)))

[3]. **Gaussian Distribution** — It does assume a linear relationship between the input variables with the output

(x1 / x2) [اذا كان في (Input) في علاقة كبيرة بينهم شيء وجه منهم ونحلي واحد منهم]

[4]. **Remove Correlated Inputs (multicollinearity)** — Like linear regression, the model can **Overfit** you have multiple highly-correlated inputs.

ال highly Correlated Input ← تجي (overfitting)

→ High Correlated

واضافنا تخفف ال (overfitting)

(x1 / x2) ← بحليها (Feature) بالي (Correlation) اعلى مع (target value)

Problems due to multicollinearity

تشبه
ال (multicollerated) (two data have the same data)

(5) **Redundancy** — two predictors (independent variables) might be providing the same information about the response variable thereby leading to unreliable coefficients of the predictors (especially for linear models)

* افهم المهم ال (Uniq response) لل (target value) ال (Uniq effect) لل (target value)

- The estimate of a predictor on the response variable will tend to be less precise and less reliable ← Redundancy اذا كان عندي
- An important predictor can become unimportant as that feature has a collinear relationship with other predictors

Ex) اذا استخدمت ال (predictor X_1) : X_1 is very important
is High correlated with predictor X_2
X_1:X_2 Feature من ال (Data set)

- Overfitting

7

(X_1, X_2) بيتم لسفة حالية ما بيطلع تأثيرهم على (target value useless) عدم الاهمية (لانه التنين بأثروا على بعض فما بفصل ...)

Problems due to multicollinearity

* your model is doing well in the train data
يعني ال model تاعك ليس اطعمته ال (train data) لتعطي نتائج عالية ولكن الطبعة (new data) لتعطل ال (overfit) لعين نتائج سيئة

- **Overfitting** — The best models are those in which each predictor variable has a unique impact on the response variable. When there are (redundant) or (correlated) predictors in the model that explains the response variable, the model tends to overfit. That means, the models do well on the train data but do a poor job on the test data thus defeating the whole purpose of model building

(the same ⇒ $X_1 = X_2$)

ال predictor $(X_1/X_2/X_3/...)$ تكون افضل اذا كان كل واحد منهم عالهم تأثير (Uniq) خاص فيه على (target value) ← (Y)

* من النهاية ال model تاعك بدك نقلل ال (multicorrelat) ال Feature مشان تتجنب (overfitting) ونقلل ال (redundancy)

Tutorial Dataset

أطبق عليهم ال
logistic Regression

class (0)

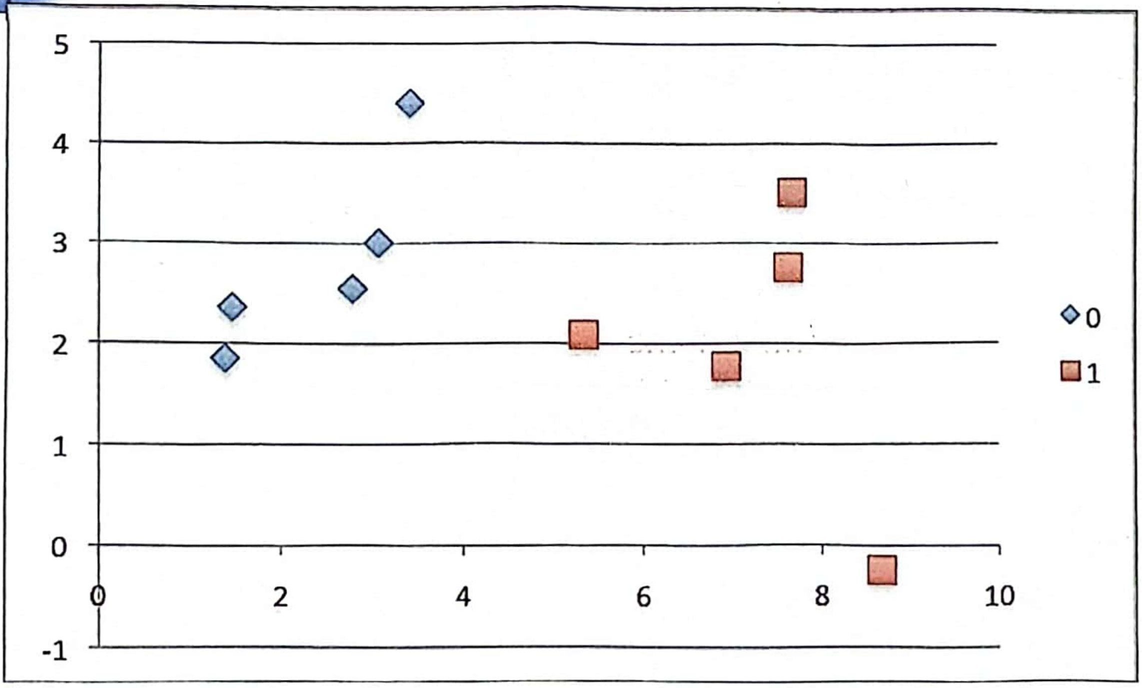
class (1)

X1	X2	Y
2.7810836	2.550537003	0
1.465489372	2.362125076	0
3.396561688	4.400293529	0
1.38807019	1.850220317	0
3.06407232	3.005305973	0
7.627531214	2.759262235	1
5.332441248	2.088626775	1
6.922596716	1.77106367	1
8.675418651	-0.242068655	1
7.673756466	3.508563011	1

(ملاحظة) قيم X_1 عالية بكون class(1)
اما ال X_1 قيم اقل بكون class(0)

two Indep. Value
في المعادلة الخطية
لازم يكون فيها (X_1 و X_2)
في (three coefficient)
B_0 B_1 B_2

Tutorial Dataset



Y Values

Example

- **Function = B0 + B1 * X1 + B2 * X2**

* بدي احامل احسب الـ $B_0/B_1/B_2$

- **Logistic function:**

$$\frac{1}{1+e^{-output}}$$

Calculate Prediction

- **Function = B0 + B1 * X1 + B2 * X2**

- **Logistic function:**

$$\frac{1}{1+e^{-output}}$$

$\boxed{\text{Initial}}$

- B0 = 0:0, B1 = 0:0, B2 = 0:0

Calculate Prediction

من (Data set)

- The first training instance is: $X1 = 2.7810836$,
- $X2 = 2.550537003$, $Y = 0$.

$$prediction = \frac{1}{1 + e^{-(B0 + B1 \times X1 + B2 \times X2)}}$$

$$prediction = \frac{1}{1 + e^{-(0.0 + 0.0 \times 2.7810836 + 0.0 \times 2.550537003)}}$$

$$prediction = 0.5$$

(First prediction)

$$\frac{1}{1+e^{0}} = \left(\frac{1}{2}\right)$$

Calculate New Coefficients

- We can calculate the new coefficients values using a simple update equation. learning rate = 0.3

(تعريف Coefficient لعد اول Round

من record

يلي صحتنا

$$\left[\, b = b_{old} + alpha \times (y - prediction) \times prediction \times (1 - prediction) \times x \,\right]$$

- ***alpha***: learning rate

معادلة التعريف بالـ weight الـ logistic Regression

اول record ⇒ $X_1 = 2.78$ / $X_2 = 2.55$ / $y = 0$

$$B0 = 0.0 + 0.3 \times (0 - 0.5) \times 0.5 \times (1 - 0.5) \times \underset{(X)}{1.0}$$

$B0$ ⇒ مش مرتبطة بـ اي (X) نعوض بدل $X = 1$

$$B1 = \cancel{B1}^{0} + 0.3 \times (0 - 0.5) \times 0.5 \times (1 - 0.5) \times 2.7810836$$

$$B2 = \cancel{B2}^{0} + 0.3 \times (0 - 0.5) \times 0.5 \times (1 - 0.5) \times 2.550537003$$

$$B0 = -0.0375$$
$$B1 = -0.104290635$$
$$B2 = -0.095645138$$

→ (لعد اول round)

Calculate New Coefficients

You can see that the model very quickly achieves 100% accuracy on the training dataset. The coefficients calculated $\boxed{\text{after 10 epochs}}$ of stochastic gradient descent are:

(100 iteration)

one
two iteration
⋮
ten

(1 epoch) الكامل
لعشر ال
10
(Iteration)
(epoch) بنسميه

$$B0 = -0.406605464$$
$$B1 = 0.852573316 \quad (14.9)$$
$$B2 = -1.104746259$$

(second record)
$X_1 = 1.46 / X_2 = 2.36 / y = 0$

$$\text{prediction} = \frac{1}{1 + e^{-(-0.4 + 0.85 \times 2.36 + -1.104 \times 2.36)}}$$

الخطوات (نفس اللي قبل) :

$B_0, B_1, B_2 \rightarrow$ second round

كيف رح نحسب ال accuracy ؟! تلا نعدل ال iteration ال (10)

صيغة ال (coefficient) B_0 B_1 B_2 لعند ما كلنا one Round (epoch)

بروح للي prediction و نخرج ال (B_0 B_1 B_2)

Make Predictions

$$\frac{1}{1 + e^{-out}}$$

X1	X2	Prediction ①		② original data set (Actual)
2.7810836	2.550537003	0.298756986	→ 0	0
1.465489372	2.362125076	0.145951056	→ 0	0
3.396561688	4.400293529	0.085333265	→ 0	0
1.38807019	1.850220317	0.219737314	→ 0	0
3.06407232	3.005305973	0.247059	→ 0	0
7.627531214	2.759262235	0.954702135	→ 1	1
5.332441248	2.088626775	0.862034191	→ 1	1
6.922596716	1.77106367	0.971772905	→ 1	1
8.675418651	-0.242068655	0.999295452	→ 1	1
7.673756466	3.508563011	0.905489323	→ 1	1

→ Accuracy 100%
$\frac{10}{10} = 100\%$
→ $\left(\frac{\text{التوقع الصحيح}}{\text{total}}\right) \times 100$

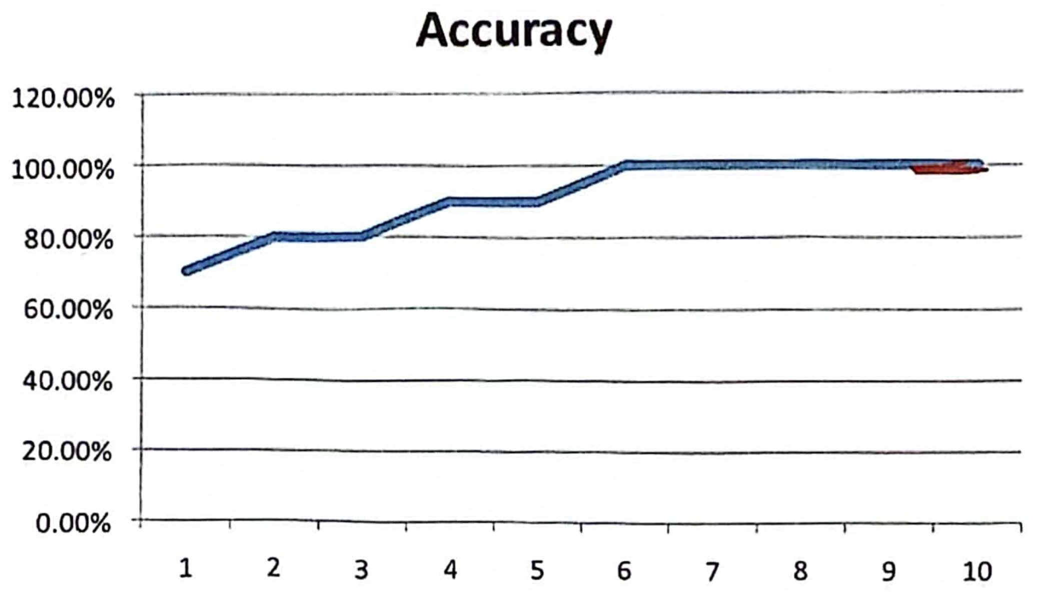
→ accuracy = 100%
$\frac{10}{10} = 100\%$

① لحساب ال accuracy بعمل قيم ال prediction value ال (0/1)

(logistic regression) يعني
اي قيم اقل من 0.5 ← 0
اكبر من 0.5 ← 1

② بعدين بروح لقارنهم في (original data set)

Repeat the Process



Accuracy

(التوقع الصحيح) م

$$accuracy = \frac{(Correct Predictions)}{Total Predictions} \times 100$$

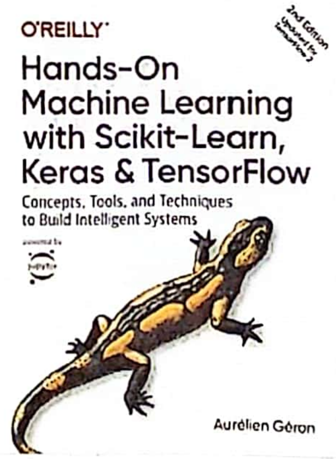
$$accuracy = \frac{10}{10} \times 100$$

$$accuracy = 100\%$$

Classification

Prof. Gheith Abandah

Reference



- Chapter 3: **Classification**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

Introduction

- YouTube Video: **Machine Learning - Supervised Learning Classification** from Cognitive Class

https://youtu.be/Lf2bCQIktTo

Outline

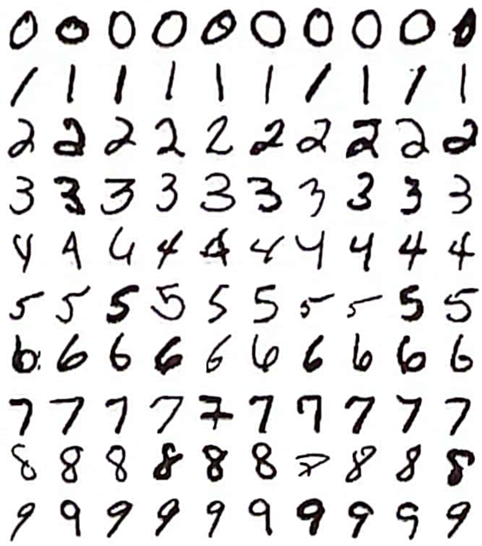
1. MNIST dataset
2. Training a binary classifier
3. Performance measures
4. Multiclass classification
5. Multilabel classification
6. Exercise

1. MNIST Dataset

(sklearn)
(Scikit-learn) عبارة عن DataSet موجودة في

- **MNIST** is a set of 70,000 small images of **handwritten digits**.
- Available from mldata.org
- **Scikit-Learn** provides **download** functions.

ماخذ sample انك كا بقاط الديك مثلاً الـ zero مكتوبة بـ 0 O 0 6 ... مطبقة الـ model انه تتعرف على صادال (zero) وتقول اذا هو zero او لا



1.1. Get the Data

```
>>> from sklearn.datasets import fetch_openml
>>> mnist = fetch_openml('mnist_784', version=1)
>>> mnist.keys()
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'details',
           'categories', 'url'])
```

DataSet اللي بدنا نشتغل عليها

1.2. Extract Features and Labels

pixel
(Feature) + sample

```
>>> X, y = mnist["data"], mnist["target"]
>>> X.shape
(70000, 784)
>>> y.shape
(70000,)
```

label   label

Features 784

70,000 images

labe ال
0
1
2
...
9

There are 70,000 images, and each image has $\boxed{\textbf{784}\text{ features.}}$
This is because each image is **28×28** pixels, and each feature simply represents one pixel's intensity, from **0** (**white**) to **255** (**black**).

يعني target الـlabel ال

1.3. Examine One Image

عرض ال
sample
الاول

```
import matplotlib as mpl
import matplotlib.pyplot as plt

some_digit = X[0]
some_digit_image = some_digit.reshape(28, 28)

plt.imshow(some_digit_image, cmap = mpl.cm.binary, interpolation="nearest")
plt.axis("off")
plt.show()
```

```
>>> y[0]
'5'
```

(فصلنا الداتا)

1.4. Split the Data

- The MNIST dataset is actually already split into a **training set** (the first 60,000 images) and a **test set** (the last 10,000 images).
- The training set is $\boxed{\textbf{already shuffled.}}$

**

```
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]
```


Outline

1. MNIST dataset
2. Training a binary classifier
3. Performance measures
4. Multiclass classification
5. Multilabel classification
6. Exercise

label عملت ال

data set الموجودة عل $\begin{bmatrix} 0 \\ 1 \\ 2 \\ \vdots \\ 9 \end{bmatrix}$ ح نكون فيها إما

2. Training a Binary Classifier

اذا بدي آخذ الرقم (5) وأحكي إذا هل هو (5) او لا ؟!

- A binary classifier can classify **two classes**. 5 or not (يلي بتهمني)
- For example, classifier for the number 5, capable of distinguishing between two classes, **5** and **not-5**.

(array) 70K sample

```
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)
```

True for all 5s, False for all other digits.

```
from sklearn.linear_model import SGDClassifier
```

(classification) تصنيف

Stochastic Gradient Descent (SGD) classifier

```
sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X_train, y_train_5)
```

somdigit = X[0]

digits اول موجود عندي → array (['5', '6', '4', ..., '4', '5', '6']

[5, 6, 7, 5, ...] 70K sample

```
>>> sgd_clf.predict([some_digit])
array([ True])
```


7 = 5 ? F

[5] → هل ال (5) = 5 ؟! T بكتب ال 1 = y_train_5

s = 6 ?! F بكتب zero ب y_train_5

بكتب zero ب y_train

لو حطيت مثلا (4) " .predict([4]) ح يطلع "False

كان جوا بدور على (5) او لا

لو (6) ح يطلع False

لو (5) ح يطلع True

من الآخر لتصنيف يا one يا zero ما فل عبارة عن [0، 1، 2، 3، 4، ...] بدي ايايا تكون

(binary classifier) لازم يكون ال zero أو one

Outline

1. MNIST dataset
2. Training a binary classifier
3. Performance measures → (classification) مقتطفات منها
4. Multiclass classification — سؤال ال binary ال multiclass
5. Multilabel classification — → ال accuracy من (3) استنتاج
6. Exercise

3. Performance Measures

1 • **Accuracy**: Ratio of correct predictions

2 • **Confusion matrix**

3 • **Precision** and **recall**

4 • **F1 Score** [جامع بين الـ precision و recall]

5 • **Precision/recall tradeoff**

الـ accuracy عبارة عن ratio

← فيها القاسم رح يكون عدد الـ prediction = الجواب الصحيح / # of predi.

Ex جاوبت عن 10 أسئلة الـ accuracy

فهمة من الأجوبة الصحيحة = 3 (أجوبة صحيحة من 10) / 10 (طول الـ list) → Correct predictions

length of the list

الـ Fold: انه بتقسم الـ Data لـ (K-fold) وبتحسب الـ accuracy

لعدد الـ (K) اذا (K=3) رح تحسب الـ accuracy

لـ 3 مرات كل مرة بتحسب الـ accuracy بـ test مختلف

* عرف الـ (Cross Validation): انا اقسم الـ test لأكثر من جزء

واحسب اكثر من (accuracy)

لنستفيد منها (اقسم الـ data لـ اكثر من ~~test~~ fold

وجرب الـ ~~test~~ لـ اكثر من Data test

كأنه صار عندي داتا دائماً جديد (test)

لهذا اكثر من Data عندي مليانه test

بس مو (ب) الـ class اذا عندي classifi

لـ B / A

فلازم يكون وضعت بشكل اكثر من test Fold

(عدد الصح / الكل / classific) عدد الـ

(فهمة من التوقعات الصح)

مدى دقة الـ model

3.1. Accuracy

الاجابة المتوقعة

```
y_pred = clone_clf.predict(X_test_fold)
n_correct = sum(y_pred == y_test_fold)
print(n_correct / len(y_pred))
```

الاجوبة المتوقعة

(Sum.) # of Correct

(Actual y)

(accuracy) عدد الاجوبة الصحيحة / عدد الاسئلة

Example how to find the accuracy.

(المصطلح لحساب الـ accuracy)

بدل الـ 3 سطر بلا مرة (ليشرح المشكلة)

المطلوب الـ

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([0.96355, 0.93795, 0.95615])
```

(3 مرات)

Feature label binary classification → (بدك الـ multiclass)

جرّب الـ classifier

الـ accuracy 3 مرات لكل مرة لـ (test data) مختلفة

Using the **cross_val_score()** function to find the accuracy on three folds

اعطاني 3 اجوبة لأنه موجود عندي

الـ $\boxed{cv=3}$ "Cross val"

لو خليت cv=10 رح يطلع (10) وانت بتوخذ

الـ (best one)

* لـ Cros. Valid. خلي الـ accuracy تعطيك مصداقية

أكثر لـ test لأنه طبقت الـ (test)

لأكثر من مرة وبكل مرة بحساب الـ accuracy

لـ (test data) مختلفة

Ex) Truth value
[5, 8, 5, 9, 5, 5, 5, 7, 5, 2]
predictions
[5, n, n, 5, n, 5, n, n, 5]
TP TN FN FP FN TP TN FN FP

3.1. Accuracy

- Use **cross_val_predict()** to predict the targets of the entire training set.

```
from sklearn.model_selection import cross_val_predict

y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
```

classifier (الموديل اللي اخترته) Feature label "binary label"

⇒ ١/ ترجعلي قائمة مثلا [..., TP, TN]
٢/ تعرف من خلال الـ Confusion matrix

نفس الداتا بتركشر انا تمرر الـ Data

الفرق بين الـ Cross val Score : Cross val predict

الفائدة: تعطيني accuracy [النسبة الإجابات الصحيحة / predi. الإجابات]

تعمل الـ Cross val Score بـ خطوة ← تتضيفني بـ
1) precion
2) recall
3) F1 Score
اطلع من خلالها ...

الحقيقة (Truth)

True positive / True negative / ١ False positive / ٢ False negative
TP / TN / FP / FN
5 5 / 8 N / 8 5 / 5 N

Truth val [5] prediction [5]

(TP) لو تعطيني الـ True Value اللي متوقع تطلع هي (5) والـ model توقعت انها (5)
انا متوقع انها (8) والـ classifier اتوقع انها (8) صح تتوقع شيء يا (5) لا
(TN) ["انا متوقع انها (8) وهي توقعت انها مش (5)" .. (5) اذا مش (8) يعني]
(FP) الـ Truth value (8) الـ machin توقعت انها (5) لا ... لما توقعت انها (5) ...
الـ Truth كانت (5) بس توقعت انها مش (5)

3.2. Confusion Matrix

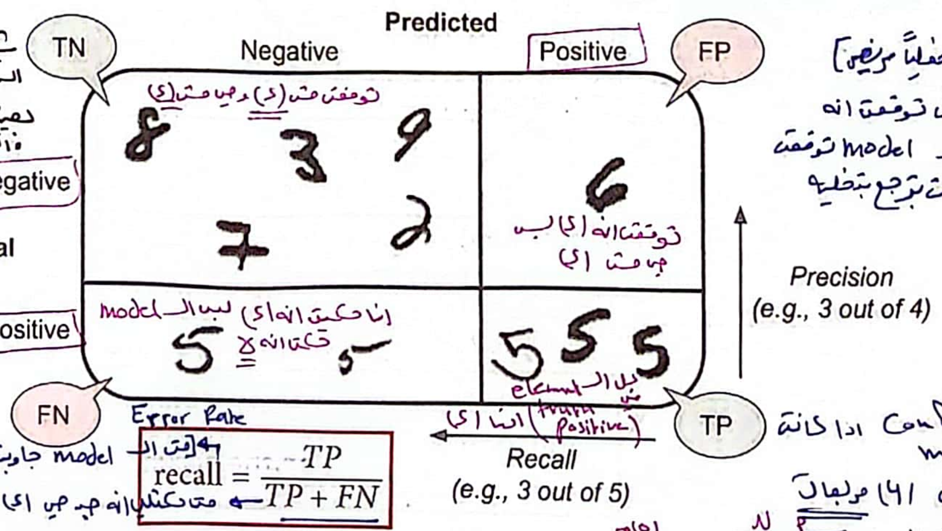
$$\text{precision} = \frac{TP}{TP + FP}$$

مدى دقة المدل ← انه الخطئين (model) دائماً (5) positive " بجمع جامعة P

* الـ (FN) اخطر شيء [مثلا انه يتوقع اعراض مرض انه مش مريض لكن هو فعلياً مريض]

* اما الـ (FP) انت توقعت انه ما معاه مرض لكن الـ model توقعت انه معاه مرض، فانت ترجع بتطلب بعض الفحوصات

TP: شايف انه سرق اجت الكاميرا لقطت انه سرق
FN: يعني شافته انه سرق والكاميرا لقطت انه X
TN: الشخص شافت انه ما سرق واصلا
الـ model حكم انه ما سرق
FP: يعني انه الشخص ما سرق والكاميرا حكمت انه سرق



توقعت مش (5) وهي مش (5)
توقعت انه (5) بس هي مش (5)
انا حكمت انها (5) لكن الـ model حكمت انه X
Error Rate

$$\text{recall} = \frac{TP}{TP + FN}$$

[هي الـ model جاوبت غلط] ← متى تطلعت انه هي هي (5)

* الـ Confusion matrix اذا كانت binary مكونة من (4) مربعات او نوافذ جنب بعض

	N	P
N [Not (5)]	TN	FP
P [Positive (5)]	FN	TP

دائما الـ Prediction and Truth ground
الـ model موجودة في X-axis
الـ y-axis هي (5) او لا (5) لا ... اللي يعطيني الـ prediction الجواب اللي انا لازم اطلع

* مثال اللي بيصير فيه سرقة انا بتعرض الحل فالمحل الكاشير بيلي عندي كاميرا عند الكمبيوتر يترجع انه يشاهد سرقة وانا شايف انه الشخص سرق يعني الحقيقة وبطلع عند الكاميرا برضه يعني الحقيقة شايف انه سرق اولا
Predi. ← الكاميرا عبارة عن (Model) تتحكم انه سرق لا [not or 5]
انا ← (Ground truth) لطلع في الصورة الحقيقية

3.2. Confusion Matrix

- Scikit Learn has a function for finding the **confusion matrix**.

```
>>> from sklearn.metrics import confusion_matrix
>>> confusion_matrix(y_train_5, y_train_pred)
array([[53057,  1522],
       [ 1325,  4096]])
```

TN FP
FN TP

Ground truth   prediction

- The first row is for the non-5s (the **negative** class):
  - 53,057 correctly classified (**true negatives**)
  - 1,522 wrongly classified (**false positives**)
- The second row is for the 5s (the **positive** class):
  - 1,325 wrongly classified (**false negatives**)
  - 4,096 correctly classified (**true positives**)

الي دخل في precision

	N	P
N	TN	FP
P	FN	TP

positive يعني انه model متى توقعت انها [5] واصابت أو اخطأت ⇓ تحديد F, T

يعني في (recall)

متى اصاب ال model حكت انه مش 5 "prediction"
متى اصابت انه مش 5 ومتى اخطأت ؟!

لما اصابت انه مش 5 ⇐ TN

لما اخطأت انه هو 5 فعلاً لكن هي حكت انه مش 5 ⇐ FN

لما اصابت يعني هو 5 وحكت 5 :: TP

لما اخطأت :: FP

3.3. Precision and Recall

Precision

$$\text{precision} = \frac{TP}{TP + FP}$$

Recall

$$\text{recall} = \frac{TP}{TP + FN}$$

```
>>> from sklearn.metrics import precision_score, recall_score
>>> precision_score(y_train_5, y_train_pred) # == 4096 / (4096 + 1522)
0.7290850836596654
>>> recall_score(y_train_5, y_train_pred) # == 4096 / (4096 + 1325)
0.7555801512636044
```

manual
بناءاً على الناتج
أو من sklearn

The precision and recall are smaller than the accuracy.
Why?

3.4. F1 Score

- The F1 Score combines the precision and recall in one metric (**harmonic mean**).

$$F_1 = \left(\frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}\right) = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

```
>>> from sklearn.metrics import f1_score
>>> f1_score(y_train_5, y_train_pred)
0.7420962043663375
```

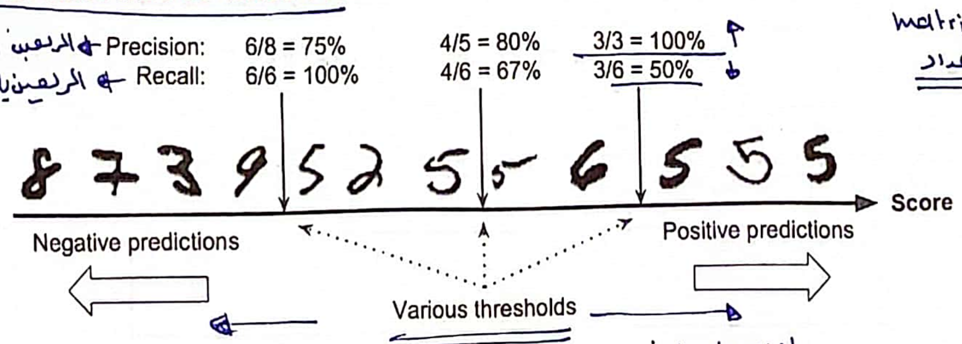

متى استخدم ال ؟!

3.5. Precision/Recall Tradeoff

- **Increase** the **decision threshold** to improve the precision when it is **bad** to have FP.

لعطيني القرار

- **Decrease** the decision threshold to improve the recall when it is **important not to miss FN**.



الرئيسين ← Precision

الرئيسين يلي تحت ← Recall

* نفرطال matrix
على خط الاعداد

N ← → P

لما اقلل ال Threshold بزيد ال negative

قد بزيد ال Recall

Recall ↑ ← prec ↓ ← ↓T

عكسية طردية

لما ازيد للسالبين

بزيد ال positive

وال positive

فتعلق ب precision

T↑

يزيد ↑ prec. ↑

يقل ↓ Recall ↓

3.5. Precision/Recall Tradeoff

- The function **cross_val_predict()** can return **decision scores** instead of predictions.

```
y_scores = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3,
                             method="decision_function")
```

عشان ترسم خط الانحدار

classifier ← (accuracy) بين كانه
Regression ← (negative mean sq. error)

- These scores can be used to compute precision and recall for all possible thresholds using the **precision_recall_curve()** function.

```
from sklearn.metrics import precision_recall_curve
precisions, recalls, thresholds = precision_recall_curve(y_train_5, y_scores)
```

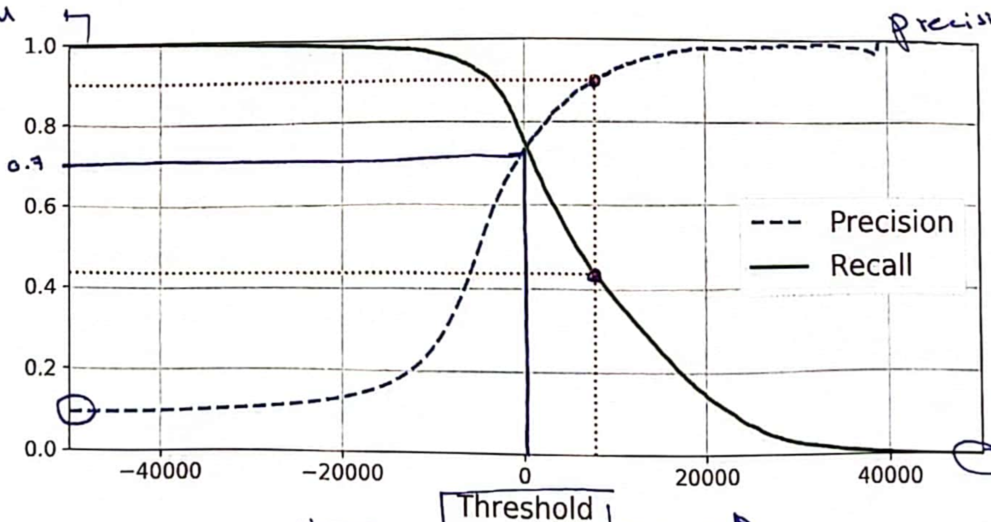
(variable (3) يرجع)

يعطيه ← label

3.5. Precision/Recall Tradeoff

(بزيد) Recall

precision (full)



pre (تقل)

Recall قليل

T تقل

T بزيد

3.5. Precision/Recall Tradeoff

- For **larger precision**, **increase the threshold**, and **decrease it** for **larger recall**.
- **Example**: To get 90% precision.

The first threshold with precision ≥ 90%

```
threshold_90_precision = thresholds[np.argmax(precisions >= 0.90)] # ~7816
y_train_pred_90 = (y_scores >= threshold_90_precision)
>>> precision_score(y_train_5, y_train_pred_90)
0.9000380083618396
>>> recall_score(y_train_5, y_train_pred_90)
0.4368197749492714
```

True when score ≥ new threshold

Outline

1. MNIST dataset
2. Training a binary classifier
3. Performance measures
4. Multiclass classification
5. Multilabel classification
6. Exercise

يلي اخذناه الـ binary classification يعني الـ (target value) عبارة عن (two class)

كان عند multiclass (digit من 0-9) حتى نقله (binary) متصور انه نعمله (classification)

الـ (digit zero) او الـ (digit one)

اذا عملنا الـ zero الـ target value

يلي منه zero نحوله (class zero) اما كل الـ class الثانيين نحولهم (not zero) حيث بحول الـ multi class لـ binary classification

4. Multiclass Classification

- Multiclass classifiers can distinguish between **more than two classes**.
- Some **algorithms** (such as Random Forest classifiers or Naive Bayes classifiers) are **capable of handling multiple classes** directly.
- **Others** (such as Support Vector Machine classifiers or Linear classifiers) are **strictly binary classifiers**.
- There are **two main strategies** to perform multiclass classification using multiple binary classifiers.

Very Good classifier مشتغل على (multiclass)

the same model

يعني نبني model يصنف جوا (class one) ولا (two) الخ

KNN

1, 2, 3 لنتعامل مع multiclass

← SVM

→ SVM: famous classifier لنتعامل binary classification

Top جدول

Good Result

لكن على multiclass classification ما بيعطي

4.1. One-versus-All (OvA) Strategy

نبني (10 binary classification)

- For example, classify the digit images into 10 classes (from 0 to 9) to **train 10 binary classifiers**, one for each digit (a 0-detector, a 1-detector, a 2-detector, and so on).
- Then to classify an image, get the decision score from each classifier for that image and select the class whose classifier outputs the **highest score**.

(digit) ؟

Ex) (8)

يعني نبني اول model بعد الـ classify zero * عندي (digit) نجربها على model الأول

ثاني ,, ,, الـ one مثلاً الـ accuracy كان 0,7

ثالث ,, ,, two على الـ model الثاني 0,2

⋮ الثالث 0,5

عاشر nine الثامن 0,97

بأخذ الـ model يلي اعطى أعلى (accuracy) حتى نجرب (digit) 8 على model لنشوف اذا 8 او لا ف منطق بيعطي اعلى (accuracy)

⇒ بجرب الصورة الـ (10 model) الـ (model) يلي بيعطي اعلى (accuracy) هو يلي بختاره (for classification) ⇒ بأخذ الـ (decision) منه

4.2. One-versus-One (OvO) Strategy

Ex) 10 class ⟹ $\frac{10 \times 9}{2} = 45$ classifier

- Train a binary classifier **for every pair** of digits.
- If there are N classes, need **N × (N – 1) / 2** classifiers. For MNIST, **need 45 classifiers**.
- To classify an image, run the image through all 45 classifiers and see which class **wins the most duels**.
- The main advantage of **OvO** is that each classifier only needs to be **trained on a subset** of the training set.
- OvO is preferred for algorithms (such as **Support Vector Machine**) that scale poorly with the size of the training set.

sum

الصف الـ (image) ← اشوف اذا (0 , 1) / (0 , 2) / (0 , ..., 9)

على digit one ازاي (1 , 0) / (1 , 1) / (1 , 2) / (1 , ..., 9)

27

ازاي الـ model ما بتربه على كل الـ (digit) كلها بتربه على الـ digit كلها الـ target ← (0/1) يعني يختار الجزء تبع (0) او (1) [أنا]

← صح بتربي الـ model عدد الـ 45 = classifier لكن فيه فائدة [انه ما بتربي الـ model على كل الـ Data]

طب لما كنا احكي (zero or not zero) كنت ادرب على كل (model)

الـ zero تكون target value ← لكل الـ digit التانيين

اما هاي كل ما بدي ابني (classifier) بنزل بس الـ digit zero والـ one ازاي

لو بدي ابني (model) (9 , 8) ما بستعمل كل الـ dataset بس فقط الجزء اللي يكون target value 8 / 9

← لما نكون الداتا اللي بدك تدرب عليها الـ model قليلة تقلل الـ (train time)

مثل الـ (SVM) رح تعطي نتائج افضل لما تكون قليل الـ (Data)

4.3. Scikit Learn Support of Multiclass Classification

- **Scikit-Learn** detects when you try to use a binary classification algorithm for a multiclass classification task, and it automatically runs **OvA** (except for **SVM** classifiers for which it uses **OvO**). (تبني) ↳ 45 classifier

أول خيار تبني 10 classifier فقط

```
>>> sgd_clf.fit(X_train, y_train)  # y_train, not y_train_5
>>> sgd_clf.predict([some_digit])
array([5], dtype=uint8)
```

OVA → بهالحالة تبني (10 classifier) 1 per digit

(classify) ← الـ y-train يكون digit 5 or not digit (5)

```
from sklearn.ensemble import RandomForestClassifier
forest_clf = RandomForestClassifier(random_state=42)
>>> forest_clf.fit(X_train, y_train)
>>> forest_clf.predict([some_digit])
array([5], dtype=uint8)
```

Better classifier than SGD (Random Forest...)

4.3. Scikit Learn Support of Multiclass Classification

- Note that the multiclass task is harder than the binary task.
- **Binary task**

(binary classification)

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([0.96355, 0.93795, 0.95615])
```

target value بالمقابل / ال y-train كاملة لعنده y-train[5] نجدوميها (1) الـ digit خوتو ركا وبلوه منها (0) كالـ digit نو (not 5)

- **Multiclass task** ⇒ بمقابله (X train و y) كاملين

حيا كدالـ classic من 0 لـ 9

```
>>> cross_val_score(sgd_clf, X_train, y_train, cv=3, scoring="accuracy")
array([0.8489802 , 0.87129356, 0.86988048])
```

النتايج اقل من (binary) لانه في multiclass

مهمته اصعب لعنه لسبب انه (digit) بده يعرف شو هو! لسبب دقته اقل

اما في (Binary) جبر اذا اكا او (لا)

دائما الـ (binary) الـ result بتكون افضل

4.4. Error Analysis

اذا اجا (Confusion matrix) لفور تحليلها

(normalization) لعمل للـ (X) كاملة مش (Y) (SGD)

لعمله دو نعمل multiclass classification مش binary

```
>>> y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
>>> conf_mx = confusion_matrix(y_train, y_train_pred)
>>> conf_mx
          0     1     2     3     4     5     6     7     8     9
array([[5578,    0,   22,    7,    8,   45,   35,    5,  222,    1],
       [   0, 6410,   35,   26,    4,   44,    4,    8,  198,   13],
       [  28,   27, 5232,  100,   74,   27,   68,   37,  354,   11],
       [  23,   18,  115, 5254,    2,  209,   26,   38,  373,   73],
       [  11,   14,   45,   12, 5219,   11,   33,   26,  299,  172],
       [  26,   16,   31,  173,   54, 4484,   76,   14,  482,   65],
       [  31,   17,   45,    2,   42,   98, 5556,    3,  123,    1],
       [  20,   10,   53,   27,   50,   13,    3, 5696,  173,  220],
       [  17,   64,   47,   91,    3,  125,   24,   11, 5421,   48],
       [  24,   18,   29,   67,  116,   39,    1,  174,  329, 5152]])
```

كل السطر الاول يكون للـ class الصفر يليصور (digit zero)

الرقم الي بدل الـ 0 = actual الـ correctly predicted as zero الـ 5578

Ex) y train الحقيقة | y-train prediction
3 | 4 ✗
2 | 2 ✓
1 | 1 ✗
0 | 5 ✓

accuracy = $\frac{2}{4}$

Many images are misclassified as 8s.

(Confusion matrix) لنفس الشي حسب الـ

العامودي الطور الثاني اول (العامود) (8) انه ما في (digit zero) وتصنفه انيا (1)

كل الارقام منطقية الا العامود رقم (8) منه مشكلة !!

الرقم (8) لصنف انه (8) من غير class فه مشكلة !!

فعندنا مشكلة في (model 8)

← يمكن التفسير انه (8) قريبة من كل الارقام

Outline

1. MNIST dataset
2. Training a binary classifier
3. Performance measures
4. Multiclass classification
5. Multilabel classification
6. Exercise

← متى multiclass الـ classifier شغال output تبعه ؟! فقط (1) هو هل هو (digit) هل هو (1) ...

لكن في (multilabel classification) انه مثلاً Email classifier بحكيلك اذا (spam او not urgent) هل هو (urgent او ham) اذا الـ model بطلع وصفين هو (multilabel)

*ممكن الشكلة تكون (binary classification) لكن هي (multilabel)

5. Multilabel Classification

لو بدنا نصنفه spam و not spam وصنفه انه urgent و not urgent [هو (binary classif.) لكن (multi label classif.)

- Classifiers that output **multiple classes for each instance**.

(هون الـ (digit) بصنفه اذا 0 او 1 او 2 هل هو odd OR even)

(derive two Feature)

بنفس الوقت الـ model (output two label)

```
y_train_large = (y_train >= 7)
y_train_odd = (y_train % 2 == 1)
y_multilabel = np.c_[y_train_large, y_train_odd]

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_multilabel)
```

① y_train_large ⇒ بتعطي (1) اذا الرقم اكبر او بيساوي (7)

② y_train_odd ⇒ بتعطي (1) اذا odd

الـ digit → First label; (odd/even) → Second label

(knn) ← Popular algorithm

```
>>> knn_clf.predict([some_digit])
array([[False,  True]], dtype=bool)
```

(not larg) (odd)

اي output الـ model نتيجه two OR three ... للنسبة (multilabel)

اذا اعطيته 8 < large or not larg; odd or not odd

y train	y train large	y train odd
8	1	0
2	0	0
0	0	0
5	0	1
9	1	1
7	1	1

Summary

1. MNIST dataset
2. Training a binary classifier
3. Performance measures
4. Multiclass classification
5. Multilabel classification
6. Exercise

Exercise

- Try to build a classifier for the MNIST dataset that achieves over 97% accuracy on the test set. Hint: the `KNeighborsClassifier` works quite well for this task; you just need to find good hyperparameter values (try a grid search on the `weights` and `n_neighbors` hyperparameters).

Hashemite University
Data Science

K-Nearest Neighbors (KNN)

Prepared by:
Dr. Mohammad Alhammouri

Introduction

Knn assum Similarity between data and Avalible case

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

يعني يكون عندي Data معينة بدي اجيب داتا جديدة وافحصها تابعة لأي class

- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories

نقارنها مع كل الـ Data الموجودة ونشوف مين اقرب لأي class ونضيفها إله مين فيهم

- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

← مشكلة الـ Knn إنه بكل مرة يكون عندي (new sample) عشان احسب لأي class لازم اقارنه بكل الـ Sample الموجودة

∴ هاي الالقوريثم تسمى [lazy Algor.] لأنه كل (new Point) عشان احطها للـ class معينة لازم تحسب المسافة بينها وبين كل الـ (Data) الموجودة. نشوف مين اقرب للنقطة واحطها للـ class الأقرب

Introduction

- K-NN algorithm stores all the available data and classifies a new data point based on the similarity.
- It is also called [**a lazy learner algorithm**] because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

يعني بكون عندنا Data من الاصل اجاها (new point) تجي الرسم المطلوب بتحسب المسافة المطلوبة وبتعملها وتشوف (new point) جاي لل Similer تكون ؟! لأي Class فترضها للـ Class

في الـ [Algor.] الماضية 6 بكون عنا Data set مثلا الـ (linear regresion) بنطبق على الداتاسيت مبدأ الـ linear regression 6 نبدء بالعمل بالـ linear reg. لعمل prediction [يعني لما نعمل model لازم يكون عمل model كامل 6 بلي بتنحسب الـ Cofficient [a, b] تبع المعادلة الـ linear انا جبت (Data point) لما بعمل بنحسب معادلة 6 العلاقة بين Data

الـ Knn عند Data اجت Data جديدة لازم لحسب المسافة بين هاي Data وكل الـ Data الموجودة

3

KNN-Example

(K = 5)

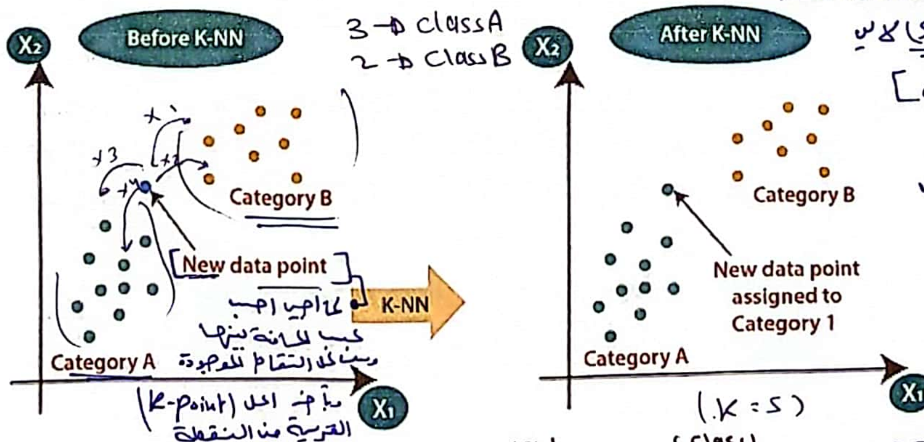
1

يعني اعداد الـ K ؟! لازم حرصنا ابدا [Algor.]

بتكون انه بدي اسب الـ K هي (5) [K = 5] ← بحسب المسافة بين النقطة وكل النقاط الموجودة وبآخذ اقرب (5) لاسي [اقرب (5) لـ distance] (بحسب المسافة)

يعني $\sqrt{(X_1 - X_2) + (Y_1 - Y_2)}$

3 → class A
2 → class B



[New data point] لما اجيب بحسب المسافة بينها وبين كل النقاط الموجودة بآخذ الـ (K-point) القريبة من النقطة

بس يطلع اقرب (5) يكون الـ (5) يكون (3) منهم تابعين للـ class A و (2) منهم تابعين للـ classB ← لذا الـ Point ح انضم للأكثر [class A]

(K = 5) (class) Data

[outliers] ← □

new Point (1)

Ex

|K = 5|
(بآخذ اقرب (5) بالي)
الـ (*) اح تكون تابعة لل class △ 6 لأنه اقرب الـ □ class (outliers)

Data (class)

← y (ح ينضم الى) new point

عدد الاكبر للـ class القريبة منى لنضم الها

4

How does KNN work?

← الشكله في (Knn) كيف اختار عدد الـ K

في الصعب الـ [Algo.] بتساعدني

- **Step-1**: Select the number K of the neighbors.
- **Step-2**: Calculate the Euclidean distance of K number of neighbors

  له بيني وبين كل Data set
- **Step-3**: Take the K nearest neighbors as per the calculated Euclidean distance.

  له بأخذ اقرب (K) اللي 6 حسب الغالبية

  اذا (K=7) وكان A=3 و (B=4) بيدرج تحت B
- **Step-4**: Among these k neighbors, count the number of the data points in each category.

اختار افضل رقم للـ K

نعملها (run) لحديثه

بدون اختار Knn

او manually اجرب افضل

K=7

K=3 اذا تنفعنا

K=3 افضل من ماليش يدينا (7)

بختار اكبر من (7)

- **Step-5**: Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6**: Our model is ready.

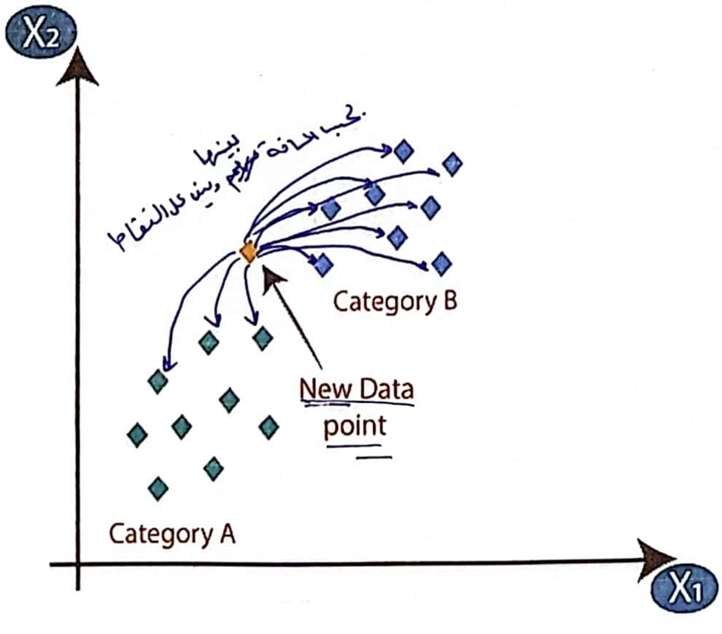
له لحدين بعدها المرز

انا لحين لعمل الـ assign للـ (new point)

للـ class الجديد وبعطيها label

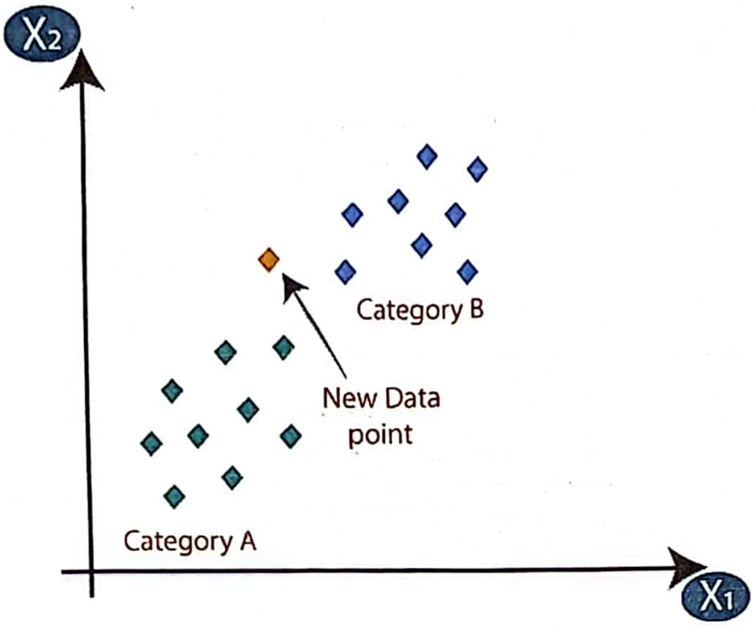
How does KNN work?

- Firstly, we will choose the number of neighbors, so we will choose the **k=5**



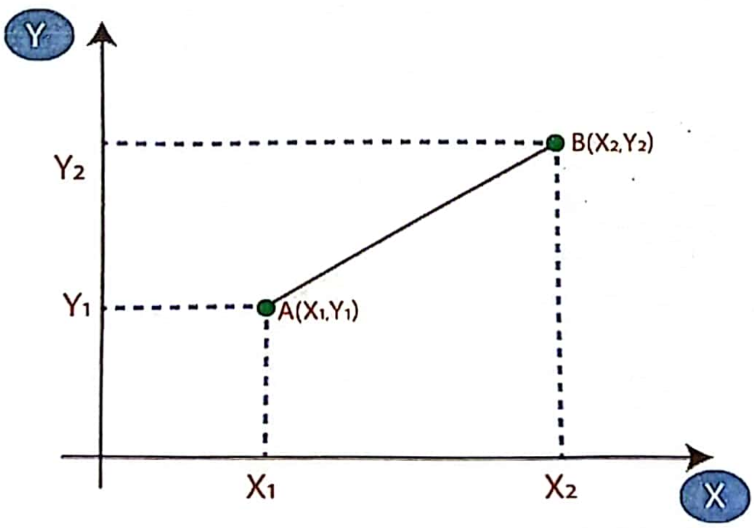
How does KNN work?

- Firstly, we will choose the number of neighbors, so we will choose the **k=5**



How does KNN work?

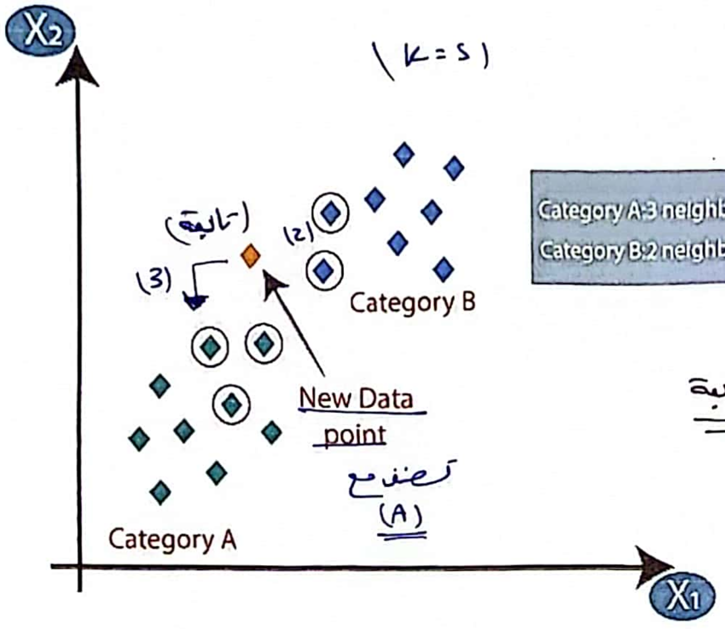
- Calculate the Euclidean Distance between the new point and all points in the training set



Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

How does KNN work?

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B.



Category A:3 neighbors
Category B:2 neighbors

K [odd] مش [even]

← بفضل اختار (K) عدد فردي
عشان اقدر اميز لاي (class) تابعة
اما اذا زوجي وكان (4) لاول و (4) للثاني
حارح اعرف اختار واحد لاي class تابعة

9

How does KNN work?

- Advantages of KNN Algorithm: ← كانه بتوخذ (K) اقرب اشي فلو كان outliers ماتتأثر فيه (كانه بتحسب المسافة بينها وبين الكل)
- It is simple to implement.
- It is robust to the noisy training data → outliers) ⇒ [ماتتأثر فيه الـ outliers]
- It can be more effective if the training data is large. ⇒ وهيك كل ما زاد بكون effective أكثر
  أعلى حجم الـ processing عالية لكن بكل مرة بتعطيني نتائج أدق
  كانه بدي احسب distance بكل مرة

- Disadvantages of KNN Algorithm:
- Always needs to determine the value of K which may be complex some time. ⇒ احيانا ما يمكن اختياري بكون خطأ (كلازم احدد قيمة الـ (K) من اول ما ابدا)
- ← دائماً عالي The computation cost is high because of calculating the distance between the data points for all the training samples.

10

Euclidean distance

Euclidean distance is calculated as the square root of the sum of the squared difference between a point a and point b across all input attributes *i*.

$$EuclideanDistance(a, b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Tutorial Dataset

* إذا جبنا الـ knn وال (y) كان ارقام مش [0 1] لازم نعمل (preprocessing)

X1	X2	Y [class]
3.393533211	2.331273381	0
3.110073483	1.781539638	0
1.343808831	3.368360954	0
3.582294042	4.67917911	0
2.280362439	2.866990263	0
7.423436942	4.696522875	1
5.745051997	3.533989803	1
9.172168622	2.511101045	1
7.792783481	3.424088941	1
7.939820817	0.791637231	1

→ y (X) ⇒ 0.0, 0.7, 0.2

مش 0 binary classifi.

Euclidean distance Example

(Data point)

Instance	X1	X2
1	3.393533211	2.331273381
2	3.110073483	1.781539638

(1) بطرح من بعض (-) بعدين تربيع (2)

(3) ± بعدين تحت الجذر (4)

ال distance الجذر التربيعي لمجموع حاصل الطرح تربيع

$SquaredDifference1 = (X1_1 - X1_2)^2 \Rightarrow SquaredDifference1 = 0.08034941698$

$SquaredDifference2 = (X2_1 - X2_2)^2 \Rightarrow SquaredDifference2 = 0.3022071889$

+ distance

$$Distance = \sqrt{SumSquaredDifference}$$

$$Distance = 0.618511605$$

Euclidean distance

بدي احسب المسافة بين هاي النقطة وكل النقاط في Dataset

New instance X1 = 8.093607318, X2 = 3.365731514,

Compute the distance between the new instance and all instances in the training dataset (Tutorial dataset)

[K=3]

No.	X1	X2	Y (class)	(X1-X1)^2	(X2-X2)^2	Sum	Distance
1	3.393533211	2.331273381	0	22.09069661	1.070103629	23.16080024	4.812566908
2	3.110073483	1.781539638	0	24.83560948	2.5096639	27.34527338	5.229270827
3	1.343808831	3.368360954	0	45.55977962	6.91395E-06	45.55978653	6.749798999
4	3.582294042	4.679179110	0	20.35194747	1.725144587	22.07709206	4.698626614
5	2.280362439	2.866990263	0	33.79381602	0.248742835	34.04255886	5.834600146
6	7.423436942	4.696522875	1	0.449128333	1.771005647	2.220133979	1.490011402 (1)
7	5.745051997	3.533989803	1	5.515712096	0.028310852	5.544022948	2.354574897
8	9.172168622	2.511101045	1	1.163294486	0.730393239	1.893687725	1.376113268 (1)
9	7.792783481	3.424088941	1	0.090494981	0.003405589	0.09390057	0.306431999 (1)
10	7.939820817	0.791637231	1	0.023650288	6.625961377	6.649611665	2.578684096

تابع لل class

new instance → 11 8.093607318 3.365731541

لما (K=3) بأخذ اقرب (3)

ال (3) نقاط ناقصين لل class 1 هي ال new instance تابع لل (class 1)

Euclidean distance

With **K** = 3, the 3 most similar neighbors to the data instance are:

No.	Distance	Y
9	0.306431999	1 ←
8	1.376113268	1 ←
6	1.490011402	1 ←

The prediction for this instance is 1

Data Science

Principal Component Analysis (PCA)

Prepared by:
Dr. Mohammad Alhammouri

Introduction

دائماً بنربط ال PCA مع ال (Correlation)
← اذا ال train time وال Complexity كثير لهيك ال PCA خيار جيد ✓
لانه لل accuracy very important احياناً 6 اعمل manually لل Reduction Feature انه ادرسهم وادمج ال Feature ال Correlated مع بعض

- PCA is a statistical technique to convert high dimensional data to low dimensional data. ممكن احصل نتائج افضل
- PCA selects the most important features that capture maximum information about the dataset. (Features)
- PCA is a linear dimensionality reduction technique (algorithm) that transforms a set of correlated variables (p) into smaller k (k<<p) number of uncorrelated variables called ***principal components***

لهيك عندي Dataset فيها (100 Feature) وفيها target value

لما استخدم ال PCA بدي اقلل ال dimension لهيك اقلل عدد ال Feature
ليش بدي اقلل عدد ال Feature !? لانه في كثير (تشابه) Correlarity بين ال Feature و بدي احاول انه ادمج ال Correlarity لهيك اذا التشابه عالي بين Feature ادمجهم مع بعض وادخلهم Feature واحد
الشي لبستفيد !? بقلل ال train time 6 لهيك بدلا ما ادرب ال model على (100 Feature) بدربه على (7 Feature)
فال PCA لبشتغل شغل عظيم انه بدمج ال Feature مع بعض وبتحولهم لل [Principal Components]
لهيك ال (100 Feature) بصيرو (6 Componets) بصير اتعامل مع (6 Compo.) مش ضروري انهم الشي عاملهم

Introduction

(1) املاشي بأخد ال Feature يلي فيها (highest Variance) (يعني فيها Very uniq Pattern) ل target value

اصلا بندرس ال Correlation بين ال Feature و كل (Compo.)

لازم يكون لكل target value ← (uniq epach) افضل شغل

standard deviation

- The feature that causes highest variance is the first principal component. The feature that is responsible for second highest variance is considered the second principal component, and so on

← اذا كان عندي two Feature الهم the same embay صار ال (multi correlaity) كانه الهم نفس التأثير عل (target value) ف ال PCA بتخلص من multi correlaity

- PCA considers the correlation among variables. If the correlation is very high, PCA attempts to combine highly correlated variables and finds the directions of maximum variance in higher-dimensional data.

لما يعني بتقوم ال Feature كلهم بتشوف ال Correlation بينهم ف بتدمج ال Feature Correlated مع بعض لحد بينشئ ادمج ال Feature مع بعض يعني انا قللت ال Values نزلت Variance

- It is important to mention that principal components do not have any correlation with each other. (uniq epach)

← صار دليل بدي اعطي ايا ن ↓ لحا بتنج عندي PCA (Principle Comp.)

← الساتا يتراطح ل target Value ما فين حدا يشابها

3

معناته انه في حد ما في علاقة بينهم

ما في Correlation بينهم وكل واحد منهم ال uniq epach عن target value

Advantages of PCA

- The training time of the algorithms reduces significantly with less number of features.

اذا قللت عدد ال Feature نزلت ال training time نزلت ال (Complexity)

- It is not always possible to analyze data in high dimensions. For instance if there are 100 features in a dataset, Total number of scatter plots required to visualize the data would be 100(100-1)2 = 4950 (100 x 99)

مشان اعمل Visualization لل Data

↓

لما اعمل Visualize بدي اوجد العلاقة بين كل Feature وال other Feature

ال PCA بتحول ال (High dim. problem) ل (low dim. problem) نبقلل عدد ال Feature، ال Feature يلي رضلوا بحتوا ال Highest Variance

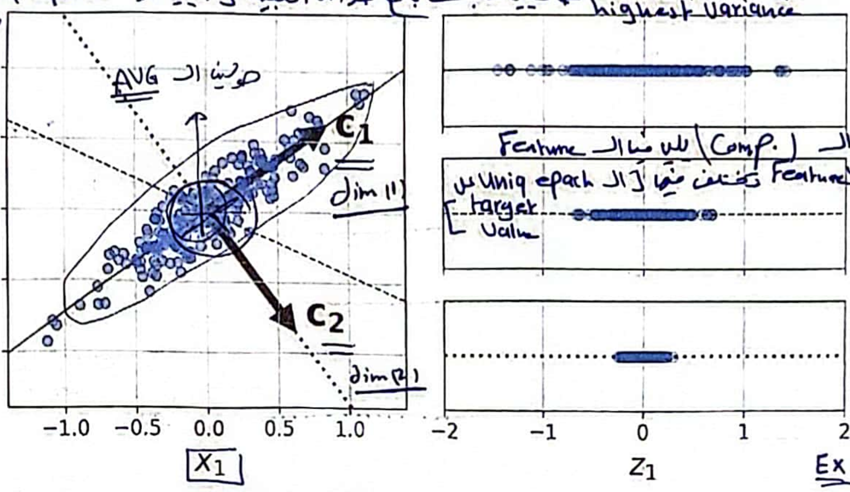
لو عندي (20 Feat.) ← (s) فيهم متشابهين بدمجهم وبخليهم (one Feature)

أقل اختلاف ما بينهم

Preserving the Variance

← Contains the highest Variance ← يعني ما بتشابه مع مراعاة القيمة [يعني من unique epoch لل Contain لل target value]

يعني فيها متشابها ال Variance (s) متشابهين الما مختلفة عن القيمة



صويتيا ال AVG

dim (1)

dim (2)

ال Feature يلي فيها (Comp.) ال ← في ال

ال Feature يختلف فيها] ال unique epach لل [target value]

C_1 [Highest Variance] ← أول Comp.

C_2

C_3

داخا فيها أهم Comp. (Highest epach, Highest Variance)

* ال dim. تمثله بـ اكثرمن line او اكثر من dim.

two Comp. [C_1، C_2]

C_1: يحتوي ال highest Variance

اعلى التقاط لعاملها projection على ال (C_1) لقدر اتخلي ال C_1 (اعلى التقاط)

C_2: اعلى التقاط عل C_2

لما احكي Feature مهمة] Ex [Feature one is Very important ← معناها انه اعلى تاثير عل output لو اخدت ال Correlation target value بينها وبين

The projection onto the solid line preserves the maximum variance.

← ما في شي لا نوقعه لجيدة عل ال AVG اما ال Other Feature تكون صويتين ال AVG تكون اهمهم اقل

variance [measures how far each number in the set is from the mean] (average), and thus from every other number in the set.

رح تكون أعلى (Correlation)

ال ← (principal Comp.) هي نشرب ال Correlation، بنشوف ال Correlated Feature وبندمجهم مع بعض، اول (Comp.) هي اهم (Comp.) تكون فيها ال (highest Variance) - (highest epach) عل (target value)

Data pre-processing before PCA

قبل ما اعمل PCA

① • Feature set must be normalized before applying PCA, the variance scall should be huge.

لما اعمل ال normalization تأثير ال Variance بكون اكبر

• PCA is applied only to numeric data

لازم اعمل encoding لكل ال Data قبل PCA

Eigenvalues and eigenvectors

A is the correlation matrix

Corr. matrix (A)

Correlation matrix (تحتوي العلاقة بين كل Feature مع بعض)

eigenvector

$$[\; Ax = \lambda x \;]$$

Comp. value

λ c_1 c_2 c_3 ...

The vector **x** is called the ***eigenvector*** of **A** — Correlation matrix

The eigenvectors of **A**, and represent the principal components (the directions of maximum variance)

لما اعمل (Principal Comp.) اهم

استخدال (Correlation matrix)

بدرس ال Correlation بين Feature وبين ال eigenvector

(λ) Eigenvalues [represent the amount of variance explained by each component]

كم قيمة ال Variance الموجودة في كل (Comp.)

(كل ما زاد ال Variance ال Principal Comp. تكون اهم ويكون رقمها (1))

Eigenvalues and eigenvectors

A is the correlation matrix

نفترض انها 1 numeric 2 normalize

هيك جاهزة

df = read.csv("breast_cancer.csv")

pca = princomp(df, cor=true)

بيعمله ال Fun — data frame وبنشأ ال pca

لهيك اعتمد (Correlation matrix)

princomp(): apply PCA for our dataset.

ونشأ ال PCA

Get Eigenvalues

لشان اشوف عشوائية get_eigenvalue(pca) → object مخزن فيه كل (principal comp.)

بزيد كل ما نزلنا كانه تاني Comp. نجمع Comp. اول 44+18 = ☐

The first PC alone captures about 44.27% variability

el (power!!)

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	1.328161e+01	4.427203e+01	44.27203
Dim.2	5.691355e+00	1.897118e+01	63.24321
Dim.3	2.817949e+00	9.393163e+00	72.63637
Dim.4	1.980640e+00	6.602135e+00	79.23851
Dim.5	1.648731e+00	5.495768e+00	84.73427
Dim.6	1.207357e+00	4.024522e+00	88.75880
Dim.7	6.752201e-01	2.250734e+00	91.00953
Dim.8	4.766171e-01	1.588724e+00	92.59825
Dim.9	4.168948e-01	1.389649e+00	93.98790
Dim.10	3.506935e-01	1.168978e+00	95.15688
Dim.11	2.939157e-01	9.797190e-01	96.13660
Dim.12	2.611614e-01	8.705379e-01	97.00714
Dim.13	2.413575e-01	8.045250e-01	97.81166
Dim.14	1.570097e-01	5.233657e-01	98.33503
Dim.15	9.413497e-02	3.137832e-01	98.64881
Dim.16	7.986280e-02	2.662093e-01	98.91502
Dim.17	5.939904e-02	1.979968e-01	99.11302
Dim.18	5.261878e-02	1.753959e-01	99.28841
Dim.19	4.947759e-02	1.649253e-01	99.45334
Dim.20	3.115940e-02	1.038647e-01	99.55720
Dim.21	2.997289e-02	9.990965e-02	99.65711

أول Comp فيها 44% تم Variance

ال eigen value رح تزيد

هياكلها كانت مضروبة ب (e)

كل ما نزلنا لتحت بده تقل ال eigen value + Variance

اما ال Cumulative رح يزيد كأنه بجمع

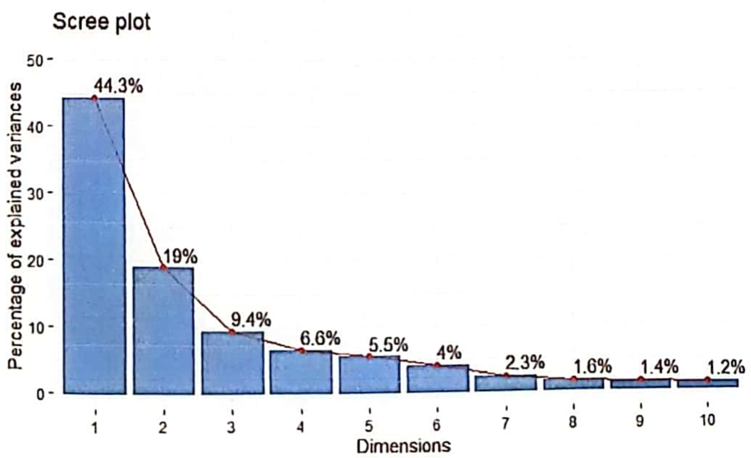
ال eigen value كل ما زادا ال number معناته ال (Comp.) أهم

Get Eigenvalues

get_eigenvalue(pca)

According to ***Kaiser's rule***, it is recommended to keep the components with eigenvalues greater than 1.0.

أفضل اشي انه اخلي pca يلي بكون ال eigenvalue اكبر من (1)



PCA on Breast Cancer

The following image shows the first 10 observations in the new (reduced) dataset. **PC1** stands for Principal Component 1, **PC2** stands for Principal Component 2 and so on. The shape of the dataset is 569 x 6 ($\boxed{\text{original are 30 features}}$)

لو احنا عدينا 30 Feature طبقنا الـ PCA صاروا (6)

الـ record الواحد ما يكون فيه (30) Column صار (6)

← حجم الـ Data ↓ قل لأنه حجم الـ Feature اقل

Feature or Comp

	PC1	PC2	PC3	PC4	PC5	PC6
0	9.192837	1.948583	-1.123166	3.633731	-1.195110	1.411425
1	2.387802	-3.768172	-0.529293	1.118264	0.621775	0.028656
2	5.733896	-1.075174	-0.551748	0.912083	-0.177086	0.541452
3	7.122953	10.275589	-3.232790	0.152547	-2.960878	3.053422
4	3.935302	-1.948072	1.389767	2.940639	0.546747	-1.226495
5	2.380247	3.949929	-2.934877	0.941037	-1.056042	-0.451039
6	2.238883	-2.690031	-1.639913	0.149340	0.040360	-0.128949
7	2.143299	2.340244	-0.871947	-0.127043	-1.427437	-1.257039
8	3.174924	3.391813	-3.119986	-0.601297	-1.522290	0.559545
9	6.351747	7.727174	-4.341916	-3.375202	1.710263	-0.723910

11

داتا سيت تشرح انواع الزهور

IRIS data set

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

IRIS data set

X_train, X_test, y_train, y_test = train_test_split()

Normalization

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```


Applying PCA

import →

```
from sklearn.decomposition import PCA

pca = PCA()
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

عملت object ←

بحسب ال pca

Applying PCA

The PCA class contains explained_variance_ratio_ which returns the Variance caused by each of the principal components

explained_variance = pca.explained_variance_ratio_

dep. on two Comp. just

0.722265	72% من variance
0.239748	23%
0.0333812	3%
0.0046056	0.004%

ممكن التقريب لل two Comp. في ال training بدل ما استخدم ال 4-Feature الموجودين في (Data set) تبعاته لتعطي 72% + 22%

It can be seen that first principal component is responsible for 72.22% variance. Similarly, the second principal component causes 23.9% variance in the dataset.

Applying PCA (1 component)

ال PCA لما تشغلها بدون اي (parameter) بتشكل ال (Possible principal Comp.) كلها احنا (n-comp = 1) معناه انه بس بدي اول Comp.

حسب يا الطريقة بحدد Comp. بدك

```
from sklearn.decomposition import PCA

pca = PCA(n_components=1)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

Training and Making Predictions

```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

يحتوي كل (one comp.)

ال (y-train) مابحل عليها الا (train) ممكن اعملها encoding

Performance Evaluation

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

cm = confusion_matrix(y_test, y_pred)
print(cm)
print('Accuracy' + accuracy_score(y_test, y_pred))
```

Accuracy: 0.933333333333

one comp. 93% ل

PCA result (2 component)

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

Here the number of components for PCA has been set to 2.
components are as follows:

accuracy
two comp لل
83%

```
[[11  0  0]
 [ 0 10  3]
 [ 0  2  4]]
0.833333333333
```


PCA result (3 component)

With three principal components, the result looks like this:

accuracy
ثلاث comp.
93%

```
[[11  0  0]
 [ 0 12  1]
 [ 0  1  5]]
0.933333333333
```

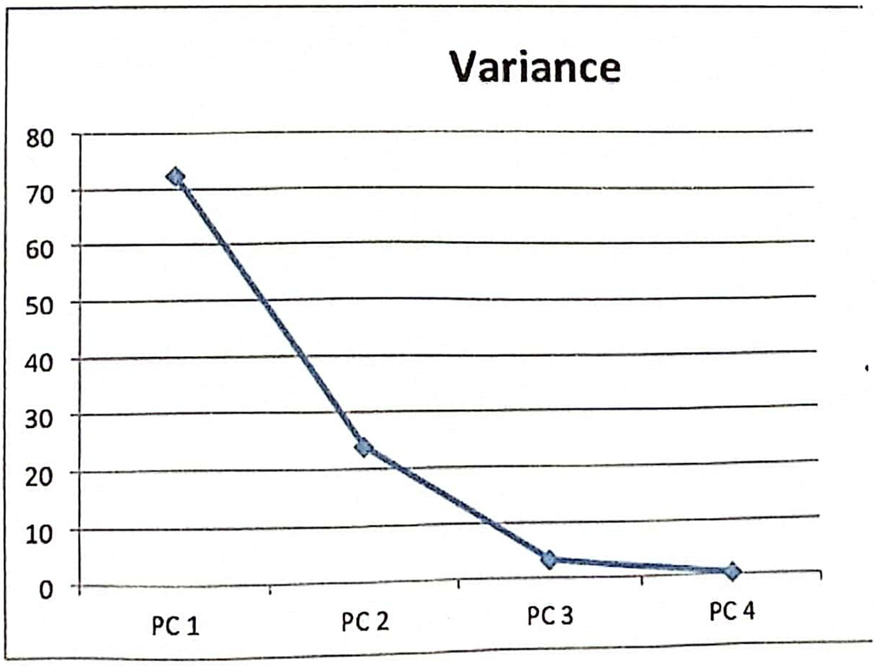
Results with Full Feature Set

```
[[11  0  0]
 [ 0 13  0]
 [ 0  2  4]]
0.933333333333
```

The accuracy received with full feature set is for random forest algorithm is also 93.33%.

Variance with difference PCA Components

[احياناً ال PCA لتعزيز التجربة ، اول PCA لد ثاني PCA الخ]



Data Science

Support Vector Machine [svm]

Prepared by:
Dr. Mohammad Alhammouri

Introduction

نفضل استخدامه في binary classification مش multiclass classif. / [غالباً بيعطيني العالي accuracy) لهيك هو من ال classifier المصنفين في machine learning]

- Support vector machine (SVM) is highly preferred by many as it produces significant accuracy with less computation power.

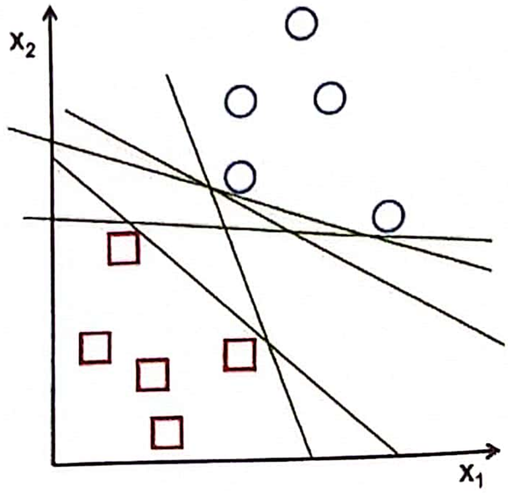
← بيستخدم في regression لكن هو افضل في classification

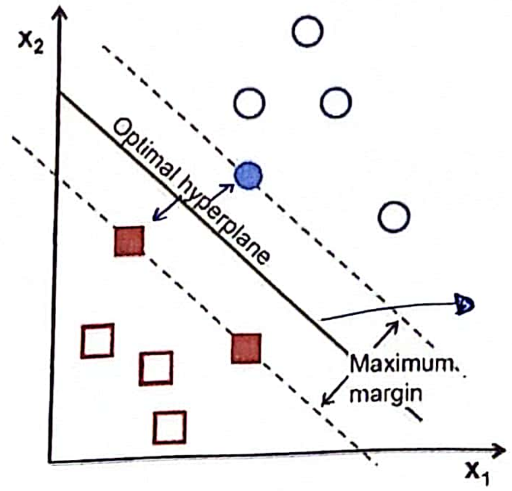
- SVM can be used for both regression and classification tasks. But, it is widely used in classification objectives.

- SVM objective: find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.

svm : بتحاول تجرب ترسم اكثر من خط آخر اشي بتهتم بال accuracy الخ

Optimum hyperplane





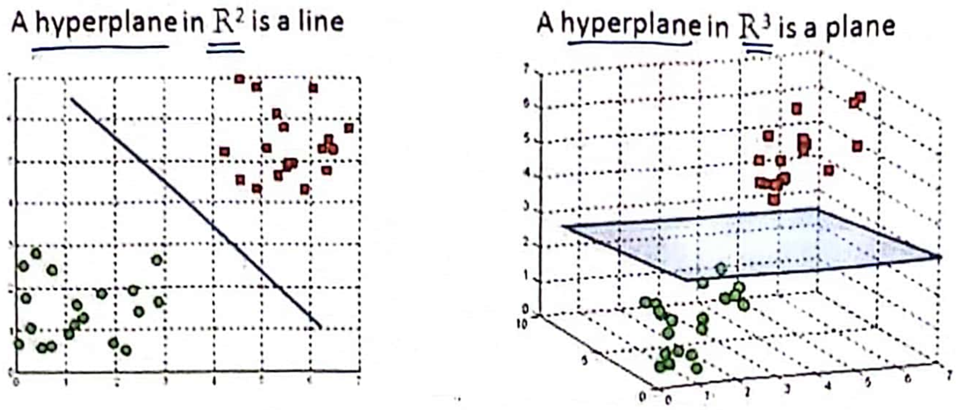
الافضل خط ما كش
ما اتهزلاي class

Optimum hyperplane

- There are many possible hyperplanes that could be chosen
- Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes
- Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

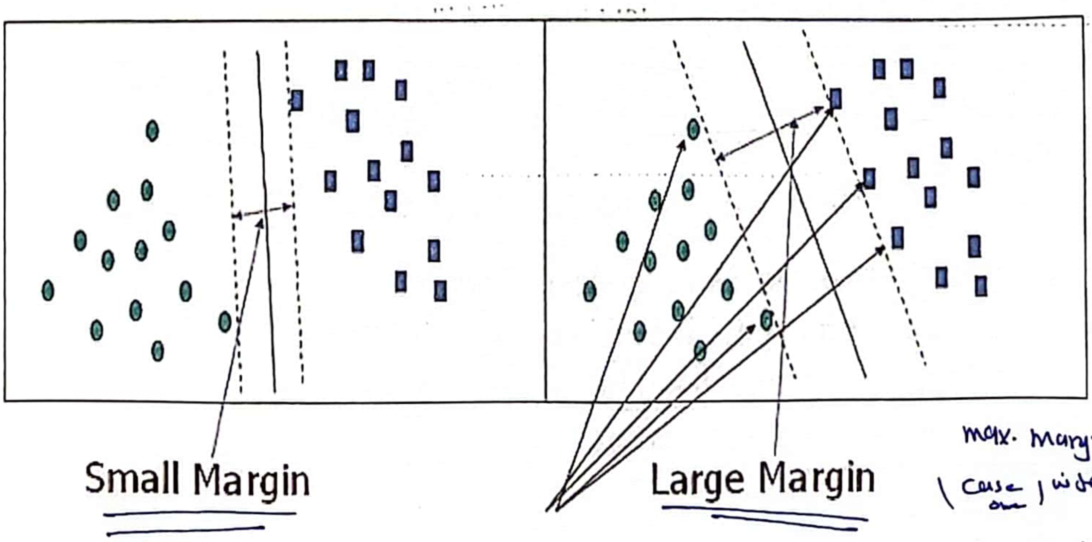
فبـ SVM رح يحسن ويقلل [كل ما اجا Data جديدة واتعلم ال Data جديدة بحسن عليه]

Optimum hyperplane



Hyperplanes in 2D and 3D feature space

Optimum hyperplane



max. Margin
(case one) افضلها

← السؤال فيه انه ~~اقصر~~ افضل خط طريق two class (hyperplan)

Output Margin

$\sigma\left(\frac{1}{1+e}\right)$

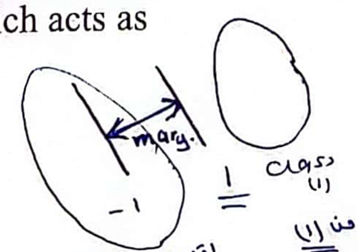
linear regression ⇒ $y=ax+b$
المعادلة الخط المستقيم

- In logistic regression, we take the output of the linear function and squash the value within the range of [0,1] using the sigmoid function. If output > (0.5) we assign it a ***label 1***, else we assign it a ***label 0***.

العطيه اي value لبطوريس ال (١،٠)
$\left[\frac{1}{1+e}\right]$

- In SVM, we take the output of the linear function and if that output is greater than **1**, we assign a ***label 1,*** and if the output is less than 1, we identify is with another label (***label -1***). OR label 0
- Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values([-1,1]) which acts as margin

نحصر كل الـ value بين الـ (١-، ١)
التصنيف الـ value $\frac{1}{-1}$



اعلى من (1) الصنفه (class 1)
اقل من (1) الصنفه (-1)

Preparing Data For SVM

اذا كان عندك Category في Data لازم تحولها الـ (binary ~~numeric~~)

- **Numerical Inputs**: SVM assumes that your inputs are numeric. If you have categorical inputs you may need to covert them to binary dummy variables (one variable for each category).

- **Binary Classification**: Basic SVM as described in this chapter is intended for binary (two-class) Classification problems. Although, extensions have been developed for regression and multiclass Classification.

عشان نتحول الـ regression الـ classification
التركيز عندنا هيكون على Classification

SVM : اكثر من افضل الـ Algor. يلي تحل Classification OR [binary class. for data]

[SMV]Kernals

- A **kernel** transforms an input data space into the required form Binary Classification:
- SVM uses a technique called the [**kernel trick.**]
- The kernel takes a low-dimensional (2D) input space and transforms it into a higher-dimensional space.
- It implicitly computes the [**dot product**] (measure similarity or distance between data points in a higher-dimensional space without actually needing to compute the transformation explicitly.

ما بنحسب الـ transformation فقط الـ dot product

لـ اضرب علاقة الـ (two vector) الـ weight تبع الـ $x_2 + x_1$ الـ dot product

لـ شوف الـ (similarity) بين الـ point + لتقسيمهم لجزئين [بحلل المتشابهين مع بعض والمختلفين مع بعض] بيرسم فاصل بينهم بحيث لو اجاه new point لقدر ميز وين تكون

* الـ SVM [basicly مهمتها تجيب الداتا [classes] تحللها بطريقة بحيث ترسم (hyperplan) لفصل بين [(two clase)

← لما تكون liner [هم اصلاً مفصولين لبعض لكن ارسم الـ plan ليكون (optimum) ← انه يكون الـ max margin From both side (لبعد نفس المسافة عن اي class)

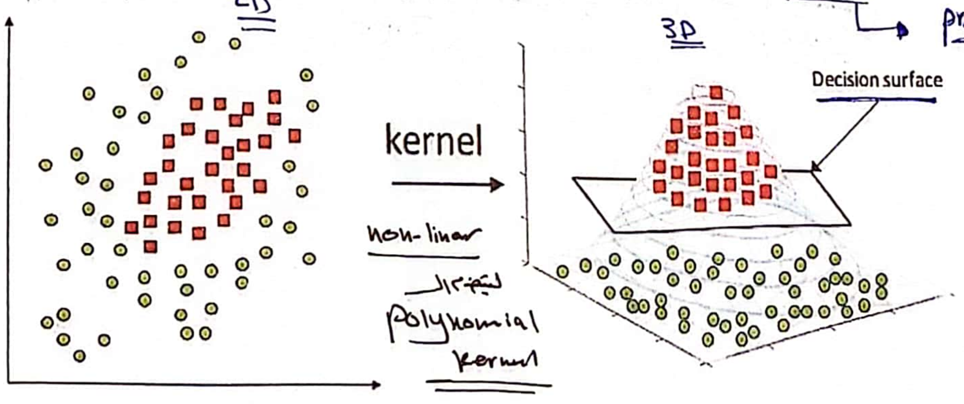
← احياناً المشكلة تكون عندي [non-linear] النقاط مخربطين لبعض ما اقدر ارسم خط الفصل بينهم الـ SVM تستخدم Kernal لـ بحول النقاط للـ (higher dim.)

SMV Kernals

(لـ اقدر افضل بينهم بشكل اسهل)

لـ بعني من الـ 2D لـ 3D

(الـ decision boundy اقدر اعمله بشكل اسهل) ← هي تحل لي الـ problem (trick)



ومعنى تحل تشتغل من Original Feature

kernel trick allows us to operate in the original feature space without computing the data coordinates in a higher dimensional space.

من math لما احولها من الـ 2D لـ 3D [بعني كل point بدي احولها للـ 3D وهاي الاشي بحتاج Computation عالي لكن هون ما بحول كل نقطة بجد الـ dotproduct between points

SMV Kernals

- **Linear Kernel** : can be used as a normal dot product for any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

[يعني بس تضرب ال pair + تجمعهم]

- **A polynomial kernel** is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.

تستخدم لل

* ال Svm عشان اعمل (binary Classification) لازم اعمل preprocessing لل y

ال y لازم احولها لل two class: -1 or 0 / 1 or 1 — ال output تبع ال Fun > 0 [تعطيه label 1]

يعني بالنهاية لازم اعمل (binary class.) — اذا Fun < 0 [تعطيه label -1]

7 * ب logistic Regression آخذ ال value بين ال (0) وال (1) اذا اكبر من 0.5 تعطيها (label 1)

اذا أقل تعطيها (label 0)

$$\frac{1}{1+e^{-(output)}}$$ = يطلع value بين ال zero وال one

* ال SVM بتشتغل (numeric Input) افضل اشي تستخدم لل (binary Class.)

Chapter 27

Support Vector Machine Tutorial

Support Vector Machines are a flexible nonparametric machine learning algorithm. In this chapter you will discover how to implement the Support Vector Machine algorithm step-by-step using sub-gradient descent. After completing this chapter you will know:

- How to use sub-gradient descent to update the coefficients for an SVM model.
- How to iterate the sub-gradient descent algorithm to learn an SVM model for training data.
- How to make predictions given a learned SVM model.

Let's get started.

27.1 Tutorial Dataset

A test problem was devised so that the classes are linearly separable. This means that a straight line can be drawn to separate the classes. This is intentional so that we can explore how to implement an SVM with a linear kernel (straight line). An assumption made by the SVM algorithm is that first class value is -1 and the second class value is $+1$.

```
X1            X2            Y
2.327868056   2.458016525   -1
3.032830419   3.170770366   -1
4.485465382   3.696728111   -1
3.684815246   3.846846973   -1
2.283558563   1.853215997   -1
7.807521179   3.290132136   1
6.132998136   2.140563087   1
7.514829366   2.107056961   1
5.502385039   1.404002608   1
7.432932365   4.236232628   1
```

class (-1)

class (1)

⇒ biny classi.

two class لانها يلينا

-1 1

SUM

weight تجيب ال

تكون المعادلة

بعد ما اجيب ال (weight) اقدر اعمل

classify لاي

new distance

ok

new data

Listing 27.1: SVM Tutorial Data Set.

The visualization below provides a scatter plot of the dataset.

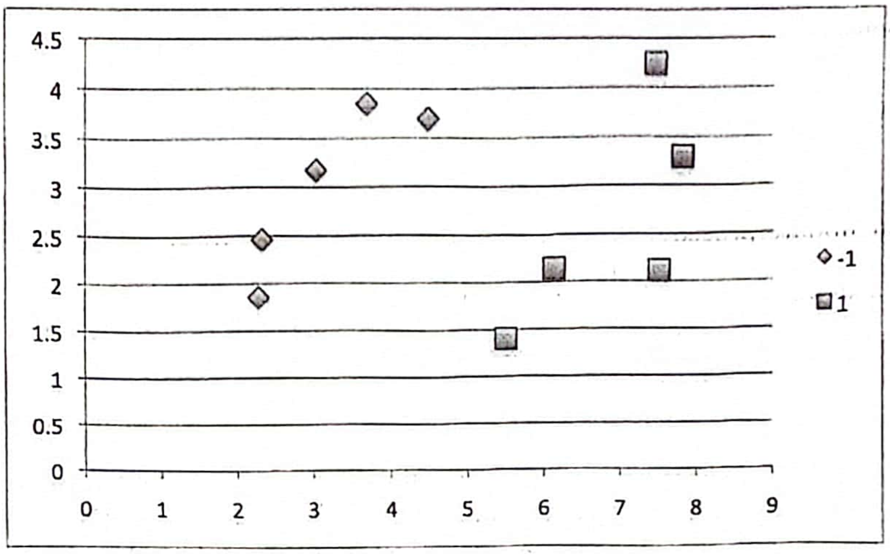


Figure 27.1: SVM Tutorial Dataset Scatter plot.

27.2 Training SVM With Gradient Descent

This section describes the form of the SVM model and how it can be learned using a variant of the gradient descent optimization procedure.

27.2.1 Form of Linear SVM Model

The Linear SVM model is a line and the goal of the learning algorithm is to find values for the coefficients that best separates the classes. The line is typically in the form (grouping the terms for readability):

B1 ،B2 ال (drop B0) هكذا بهمال

$$B0 + (B1 \times X1) + (B2 \times X2) = 0 \tag{27.1}$$

Where $B0$, $B1$ and $B2$ are the coefficients and $X1$ and $X2$ are the input variables. This will be the form of the equation that we will be using with one small modification, we will drop the bias term ($B0$) also called the offset or the intercept. For example $B0 = 0$.

ال weight في البداية (zero)

$$(B1 \times X1) + (B2 \times X2) = 0 \tag{27.2}$$

This means that the line will pass through the origin ($X1 = 0$ and $X2 = 0$). This is just to make the tutorial easier to follow and because our simple problem does not really need it, you can add the bias term back in if you like.

27.2.2 SVM Optimization Method

The optimization algorithm to find the coefficients can be stated as a quadratic programming problem. This is a type of constraint optimization where fast solvers can be used. We will not be using this approach in this tutorial. Another approach that can be used to discover the coefficient values for Linear SVM is sub-gradient descent. In this method a random training pattern is selected each iteration and used to update the coefficients. After a large number

بناءً لال output بيصير المثل
على ال weight بعض المثل بخليهم
بحدما احصل اقل error

of iterations (thousands or hundreds of thousands) the algorithm will settle on a stable set of coefficients. The coefficient update equation works as follows. First an output value is calculated as:

اذا ال output اكبر من ١ هاد ال instance
ما بنا نرجع نعدل ال update تعول ال weight
ما بنرجع نكمل

$$output = Y \times (B1 \times X1) + (B2 \times X2) \tag{27.3}$$

Two different update procedures are used depending on the output value. If the output value is greater than 1 it suggests that the training pattern was not a support vector. This means that the instance was not directly involved in calculating the output, in which case the weights are slightly decreased:

output اكبر من ١ ع التعديل

رح نقلل شوي ~~ضعف~~ ضعيف

$$\underbrace{b}_{\text{(new weight)}} = (1 - \frac{1}{\underbrace{t}_{t:\ \text{iteration}}}) \times b \tag{27.4}$$

Where b is the weight that is being updated (such as $B1$ or $B2$), t is the current iteration (e.g. 1 for the first update, 2 for the second and so on). If the output is less than 1 then it is assumed that the training instance is a support vector and must be updated to better explain the data.

اذا ال output اقل من ١ ع التعديل

$$\underbrace{b = (1 - \frac{1}{t}) \times b}_{\text{(learning weight)}} + \frac{1}{lambda \times t} \times (y \times x) \tag{27.5}$$

Where b is the weight that is being updated, t is the current iteration and lambda is a parameter to the learning algorithm. The lambda is a learning parameter and is often set to very small values such as 0.0001 or smaller. The procedure is repeated until the error rate drops to a desirable level or for a very large fixed number of iterations. Smaller learning rates ofter require much longer training times. The number of iterations is a downside to this learning algorithm.

27.3 Learn an SVM Model from Training Data

In this section we will work through a few updates to the coefficients to demonstrate the SVM learning algorithm. We will use a very large lambda value: $lambda = 0.45$. This is unusually large and will force a lot of change on each update. Normally $lambda$ values are very small.

27.3.1 Learning Iteration #1

We will start by setting the coefficients to 0.0.

$$\begin{aligned} B1 &= 0.0 \\ B2 &= 0.0 \end{aligned} \tag{27.6}$$

We also need to keep track of which iteration we are on: $t = 1$. We will train the model using the order of the training patterns. Ideally, the order of the patterns would be randomized to avoid the learning algorithm getting stuck. The first training pattern we will use to update

$B_0 = B_1 = 0$ / $t = 1$

the coefficients is: Instance: $X1 = 2.327868056$, $X2 = 2.458016525$, $Y = -1$. We can now calculate the output value for this iteration.

$$\begin{aligned} output &= Y \times (B1 \times X1) + (B2 \times X2) \\ output &= -1 \times (0.0 \times 2.327868056) + (0.0 \times 2.458016525) \\ output &= 0.0 \quad [0 < 1] \end{aligned} \tag{27.7}$$

Easy enough. The output is less than 1.0, therefore we will use the more complex update procedure that assumes the training pattern is a support vector:

$$\begin{aligned} b &= (1 - \frac{1}{t}) \times b + \frac{1}{lambda \times t} \times (y \times x) \\ B1 &= (1 - \frac{1}{1}) \times \overset{\text{القديم}}{0.0} + \frac{1}{0.5 \times 1} \times (-1 \times 2.327868056) \\ B1 &= -5.173040124 \end{aligned} \tag{27.8}$$

لأنه احنا ضربنا X_1

And for $B2$ this is:

$$\begin{aligned} B2 &= (1 - \frac{1}{1}) \times 0.0 + \frac{1}{0.5 \times 1} \times (-1 \times \overset{X_2}{2.458016525}) \\ B2 &= -5.462258944 \end{aligned} \tag{27.9}$$

27.3.2 Learning Iteration #2

We now have updated coefficients that we can use on the next iteration of the learning algorithm with the second instance from the training dataset: Instance: $X1 = 3.032830419$, $X2 = 3.170770366$, $Y = -1$. Again, we must keep track of the iteration: $t = 2$. Let's repeat the process.

$$\begin{aligned} B1 &= -5.173040124 \\ B2 &= -5.462258944 \end{aligned} \tag{27.10}$$

The output value is calculated as:

$$\begin{aligned} output &= -1 \times (-5.173040124 \times 3.032830419) + (-5.462258944 \times 3.170770366) \\ output &= 33.00852224 > 1 \end{aligned} \tag{27.11}$$

The output value is larger than 1.0, suggesting that this training instance is not a support vector. We can update the $B1$ coefficient accordingly:

$$\begin{aligned} b &= (1 - \frac{1}{t}) \times b \\ B1 &= (1 - \frac{1}{2}) \times -5.173040124 \\ B1 &= -2.586520062 \end{aligned} \tag{27.12}$$

القيمة القديمة هي لـ B_1

And for the $B2$ coefficient:

$$\begin{aligned} B2 &= (1 - \frac{1}{2}) \times -5.462258944 \\ B2 &= -2.731129472 \end{aligned} \tag{27.13}$$

27.3.3 More Iterations

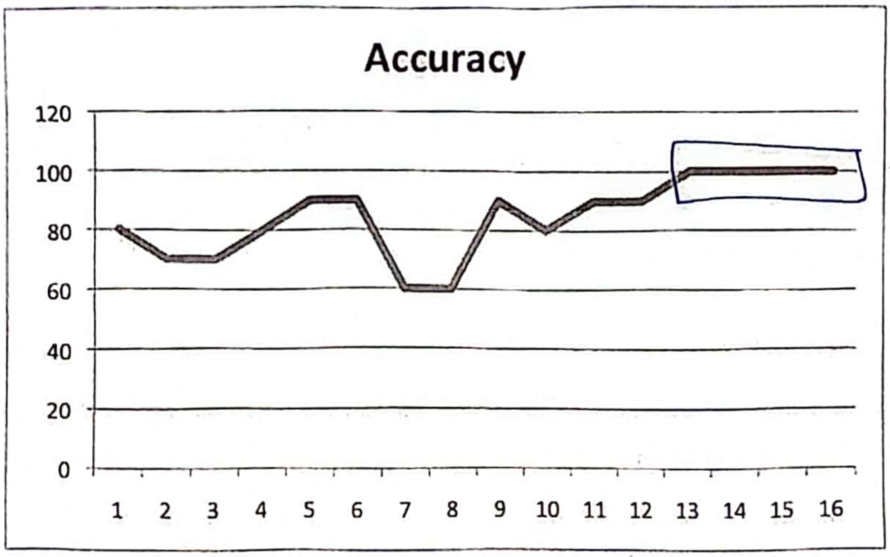
ولما نكرر الـ Data set مرة كاملة تسمى (epoch)
لحين (Full Round) لكامل الـ Data set

Repeat this process for the rest of the dataset. One pass through the dataset is called an epoch. Now repeat the process for a further 15 epochs for a total of 160 iterations (16 epochs × 10 updates per epoch). It is possible to keep track of the loss or the accuracy of the model for each epoch. This is a great way to get insight into whether the algorithm is converging or whether there is a bug in the implementation. If you plot the accuracy for the model at the end of each epoch, you should see something that looks like the following graph:

Data set

15+1

الكامل



وصل الـ accuracy 100% بعد 16

Figure 27.2: Support Vector Machine Model Accuracy.

You can see that after 16 epochs that we achieve an accuracy of 100% on the training data. You should arrive at final values for the coefficients that look like the following:

16 بعد iteration

$$
\begin{aligned}
B1 &= 0.552391765 \\
B2 &= -0.724533592
\end{aligned}
\tag{27.14}
$$

The form of the learned hyperplane is therefore:

$X_1 = \#$
$X_2 = ?!$

اذا جاءتنا Data جديدة بدو نعرف اي الـ class بنعوضهم في المعادلة

$$0 + (0.552391765 \times X1) + (-0.724533592 \times X2) = 0 \tag{27.15}$$

(drop) B_0

اذا الـ out. > 0 صنفها class (1)
out. < 0 صنفها (class -1)

27.4 Make Predictions with SVM Model

Now that we have coefficients for the line, we can make predictions. In this section we will make predictions for the training data, but this could just as easily be adapted to make predictions for new data. Predictions can be made using the following equation:

$$
\begin{aligned}
output &= (B1 \times X1) + (B2 \times X2) \\
Y &= -1 \text{ IF } output < 0 \\
Y &= +1 \text{ IF } output > 0
\end{aligned}
\tag{27.16}
$$

Using the above coefficients and these prediction rules, we can make a prediction for each instance in the training dataset:

```
Output        Crisp  Y
-0.495020399  -1     -1
-0.622019096  -1     -1
-0.20066956   -1     -1
-0.75170826   -1     -1
-0.081298299  -1     -1
1.928999147   1      1
1.836907801   1      1
2.624496306   1      1
2.022225129   1      1
1.036597783   1      1
```

Listing 27.2: Accuracy of the Learned SVM Model.

Comparing the crisp prediction (*Crisp*) to the expected output column (Y), we can see that our model has achieved 100% accuracy.

27.5 Summary

In this chapter you discovered how to implement the Support Vector Machine algorithm from scratch using the sub-gradient descent optimization technique. You learned:

- How to update coefficient values for SVM using sub-gradient descent.
- How to iterate the sub-gradient descent procedure to find good coefficient values.
- How to make predictions using a learned SVM model.

You now know how to implement the Support Vector Machine algorithm from scratch using sub-gradient descent. This concludes your introduction to nonlinear machine learning algorithms. In the next part you will discover ensemble machine learning algorithms starting the bagging.

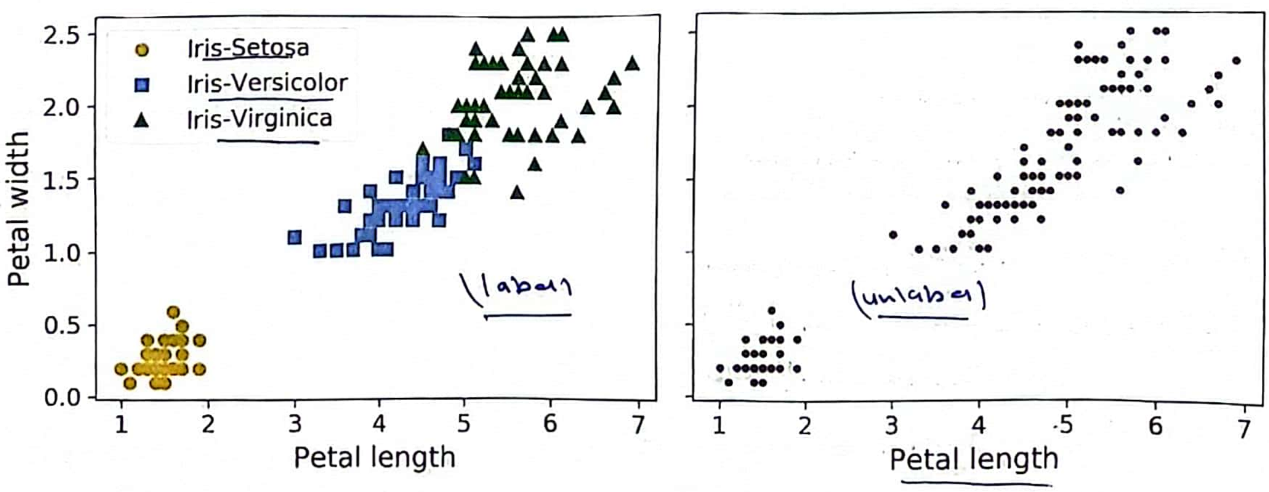
Unsupervised Learning and Clustering

[Data is not labeled]

Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow,
Aurélien Géron

Clustering

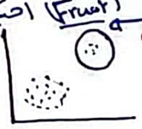
- The task of **identifying similar instances** and assigning them to clusters, i.e., groups of similar instances.
- **Classification** (left) versus **clustering** (right)



Clustering Applications

(نبناءً على الأوامر اللي عندك) هم

- **Customer segmentation**: useful for recommender systems.
- **Data analysis**: discover clusters of (similar instances) as it is often easier to analyze clusters separately. (القريبة من بعض)
- (PCA) **Dimensionality reduction**: find affinity (features) to the found cluste clusters
- (Fruit) احتيال **Anomaly detection**: any instance that has a low affinity to all the clusters is likely to be an anomaly.



- [**Semi-supervised learning**:] perform clustering and propagate the labels to all the instances in the same cluster.

ال Data ← جزء منها label data (supervised) لطبقها super. الخاص
جزء منها (un label data)
↓
لطبق عليها ال Clustering

(لتقسيم الـ Clustering والشرحها)

K-Means

- **Quick** and **efficient** algorithm

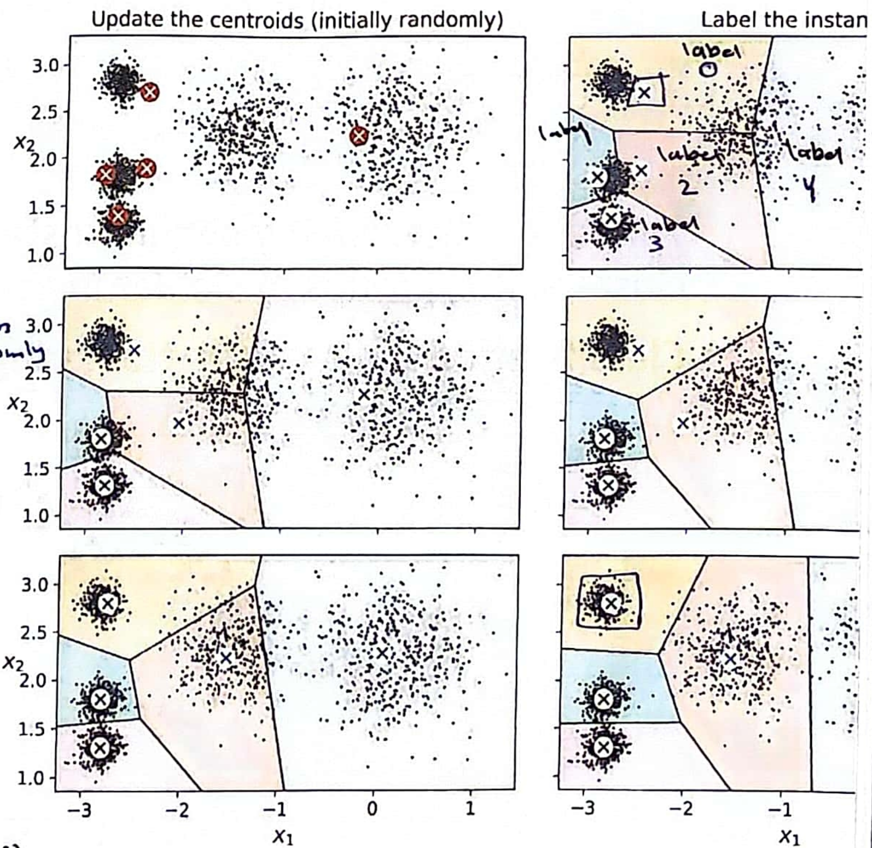
1. need to specify the number of clusters
2. The centroids are initialized randomly

(Center) لتقسيم ال

- **Scale** before clustering

3. the instance are labelled
4. the centroid are update. and the instance are relabeled

- Need to **specify** the **number of clusters**



K-means بتوزع النقاط عشوائياً (center) ال
النقاط ال Center ممكن يكون Randomly
ممكن نبادل ال Data اللي ترسمها
نعمل (label) لل (instance) حسب قربها من center
+ ممكن نعمل update لل (Centroid) لحد (best fit) داخل cluster

~~Sentr~~ [Centroids the best Fit]
(بسبب تبديل او تغيير ترتيب عدد ال cluster
او ال (Centroid) من البداية ما كان مناسب)

* كيف بنعرف ازاي انه الـ clustering the best
!? Centroid الـ initialization الـ is the best
له من خلال (model inertia)
(AVG)
is the mean squared distance between each instance and its closest centroid
له بحسب المسافة بين النقاط مربوط (center) لكل cluster
فلما اختار بشكل عشوائي بين البداية رح تضل تحسب لتوصل للـ اقل (inertia)

K-Means

- Cluster to 5 clusters

[0, 4]
X y

```
from sklearn.cluster import KMea
k = 5
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)
y_pred
array([4, 0, 1, ..., 2, 1, 0],
      dtype=int32)
# Hard clustering:
X_new = np.array([[0, 2], [-3, 3]])
kmeans.predict(X_new)
array([1, 2], dtype=int32)
```

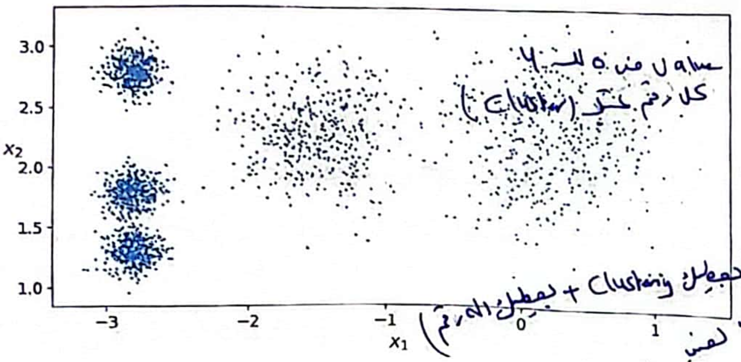
Data

y الـ value من 0 لـ 4
كل رقم بيمثل (cluster)

بـ K-means بتوصل لعمل y pred.?!
تابعة لأي (cluster) كل نقطة

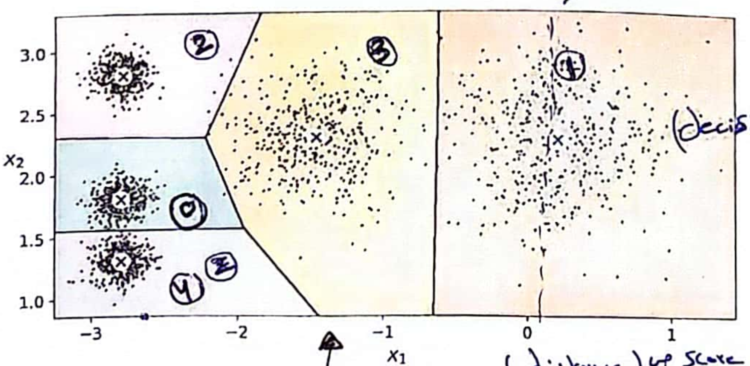
بتعطيك clustering + بتعطيك الـ predict رقم
لعمل
(Sharp decision)

cluster 1 — cluster 3
[0, 2] — [-3, 3]



K-Means

boundary



Can be a dimensionality reduction technique.

```
kmeans.cluster_centers_
array([[-2.80389616, 1.80117999],
       [ 0.20876306, 2.25551336],
       [-2.79290307, 2.79641063],
       [-1.46679593, 2.28585348],
       [-2.80037642, 1.30082566]])
```

Centroid لكل الـ Cluster
(Cluster)

(decision) بتمثل array بتعطيك

```
# Soft clustering, a score per
#  cluster:
kmeans.transform(X_new)
array([[2.81093633, 0.32995317,
        2.9042344 , 1.49439034,
        2.88633901],
       [1.21475352, 3.29399768,
        0.29040966, 1.69136631,
        1.71086031]])
```

[0,2]
[-3,3]

الـ score هي (distance)
Cluster Code
0 — C1 (cluster 1)
1 — C2 (cluster 2)
2 — C3
3 — C4
4 — C5

اقل رقم

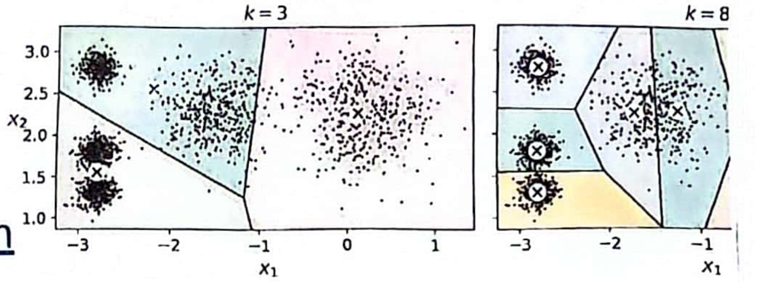
The score can be the distance between the instance and the centroid

Score : distance

بتعطيها اقل قيمة بالـ (distance)

K-Means

- It is important to specify the **right** number of clusters *k*.

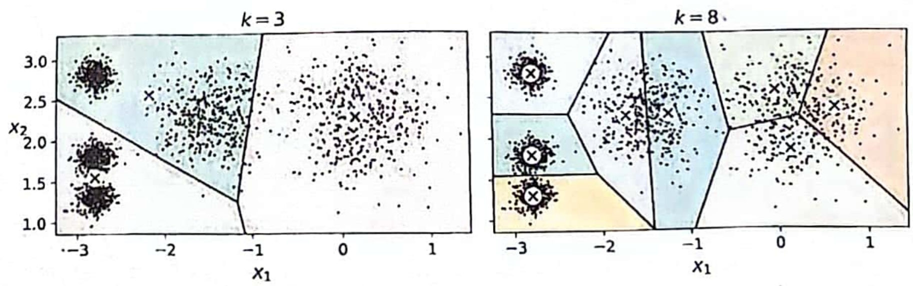


- Find *k* that gives highest mean silhouette coefficient.

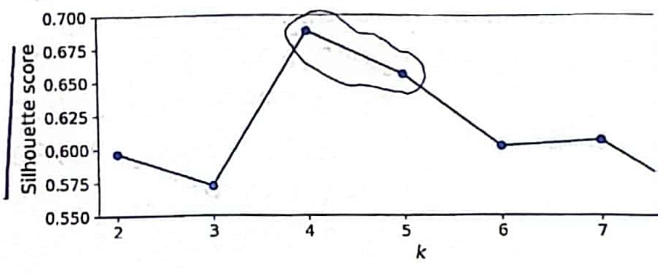
The silhouette coefficient can vary between -1 and +1:

- a coefficient close to +1 means that the instance is well inside its own cluster and far from other clusters
- coefficient close to -1 means that the instance may have been assigned to the wrong cluster

K-Means




```
from sklearn.metrics import
                    silhouette_score
silhouette_score(X, kmeans.labels_)
0.655517642572828
```



Centroid initializations

If know approximately where the centroids should

```
good_init = np.array([[-3, 3], [-3, 2], [-3, 1], [-1, 2], [0, 2]])
kmeans = KMeans(n_clusters=5, init=good_init, n_init=1)
```

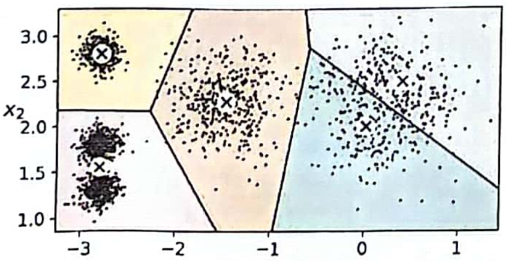
Another solution is to run the algorithm multiple times with different random initializations and keep the best solution. This is controlled by the **n_init** hyper-parameter (by default, it is equal to 10)

How exactly does it know which solution is the best?

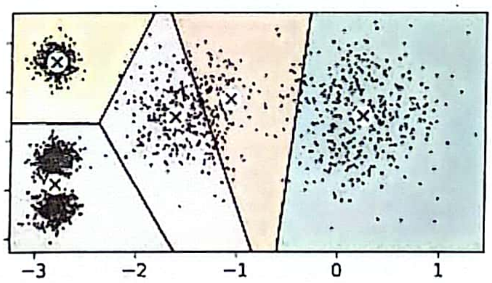
model's *inertia*: is the mean squared distance between each instance and its closest centroid

The **KMeans** class runs the algorithm n_init times and keeps the model with the lowest *inertia*

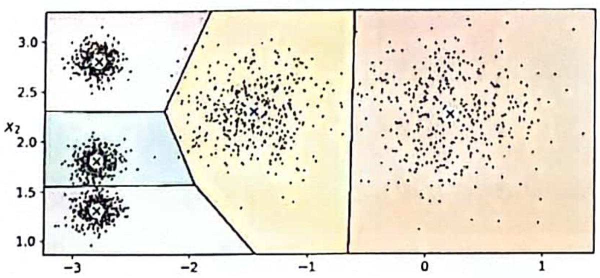
Centroid initializations



inertia= 223.3



inertia= 237



inertia= 221

What is the optimal # of cluster ?!

بالبداية اهم شي منا (K-means) الخوارزم كم (cluster) الانا بدي

دائماً بلاقي اله (lowest inertia) بكون ال model الصح بناء عليه بكون كم cluster فيه

Finding the Optimal Number of Clusters (inertia)

- You might be thinking that we could just pick the model with the lowest inertia, right?
- It is not that simple. The inertia for **k=3 is** $\boxed{\textbf{653.2}}$, which is much higher than for $\boxed{\textbf{k=5}}$ (which was 211.6), but with $\boxed{\textbf{k=8,}}$ the inertia is just **119.1**

لهيك هو افضل اشي بانه اقل (Inertia) ?!! لا

وانه اقل عدد cluster لاقل شي ?! برضه مش صح

ب اختيار عدد ال cluster

عشان اختار اقل Inertia ?

لا بكون اقل وحدة

بكون عندي cluster كثير

- The inertia is not a good performance metric when trying to choose k since it keeps getting lower as we increase k)
- The more clusters there are, the closer each instance will be to its closest centroid, and therefore the lower the inertia will be

دائماً بحاول الاقل optimal

↑cluster ⇒ ↑centroid ⇒ ↑ النقاط قريبة من centroid ⇒ ↓ Inertia

Finding the Optimal Number of Clusters (inertia)

The *inertia* drops very quickly as we increase *k* up to 4, but then it decreases much more slowly as we keep increasing *k*

بعد ال (4) بنقل بكون قليل

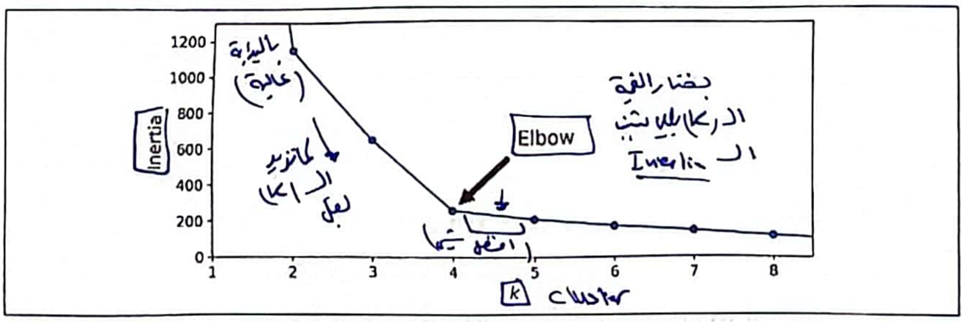


Figure 9-8. Selecting the number of clusters k using the "elbow rule"

Finding the Optimal Number of Clusters (silhouette coefficient) 2

مش تراره 100%
(Indecalion) تحويل

(بكون على مستوى الـ labels clustering)

هنا لـ من خلاله لتقدر ارم (the best value of k)

- Find k that gives highest mean silhouette coefficient.

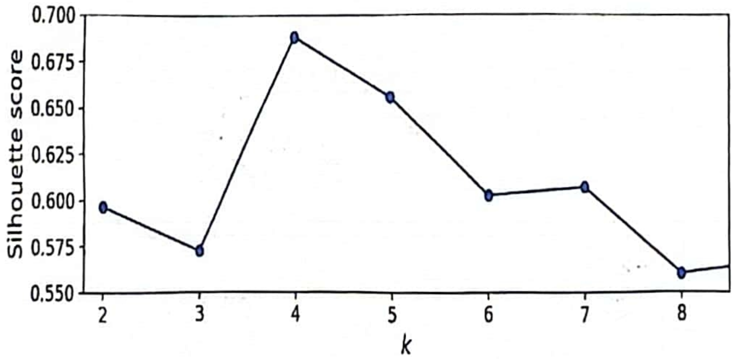
The silhouette coefficient can vary between -1 and +1:

- A coefficient close to **+1** means that the instance is well inside its own cluster and far from other clusters
- A coefficient close to **0** means that it is close to a cluster boundary
- A coefficient close to **-1** means that the instance may have been assigned to the wrong cluster

ماشي بل boundary
كل ما قربنا من (+1)
بـ ابتعد الـ cluster
بدي اقلل من boundary
فاصل بعيد عن النقاط

الانا assign الـ instance
wrong cluster

Finding the Optimal Number of Clusters (silhouette coefficient)




```
from sklearn.metrics import
                    silhouette_score
silhouette_score(X, kmeans.labels_)
0.655517642572828
```

Clustering Applications (Using Clustering for Preprocessing)

- Clustering can be an efficient approach to dimensionality reduction
- In particular as a preprocessing step before a supervised learning algorithm
- *Example: digits dataset* which is a simple MNIST-like dataset containing 1,797 grayscale 8×8 images representing digits 0 to 9.

كل Para القريبة من بعض نجمعها (Cluster)

```
from sklearn.datasets import load_digits

X_digits, y_digits = load_digits(return_X_y=True)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_digits, y_digits)
```

Clustering Applications (Using Clustering for Preprocessing)

- let's fit a Logistic Regression model:

```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)
```

Let's evaluate its accuracy on the test set:

```
>>> log_reg.score(X_test, y_test)
0.9666666666666667
```

Clustering Applications (Using Clustering for Preprocessing)

- Let's see if we can do better by using Kmeans as a preprocessing step
- We get 98.2% Accuracy after using Kmeans with (50 clusters)

```
from sklearn.pipeline import Pipeline
```

بدي اعمل تشغيلين ① ② ، بعمل اكتر step مع بعض

```
pipeline = Pipeline([
    ("kmeans", KMeans(n_clusters=50)),
    ("log_reg", LogisticRegression()),
])
pipeline.fit(X_train, y_train)
```

Now let's evaluate this classification pipeline:

```
>>> pipeline.score(X_test, y_test)
0.9822222222222222
```

Score

Clustering Applications (Using Clustering for Preprocessing)

- What is the optimum number of clusters in this case!
- Let's use **GridSearchCV** to find the optimal number of clusters
- With k=90, we get the highest accuracy 98.4%

```
from sklearn.model_selection import GridSearchCV

param_grid = dict(kmeans__n_clusters=range(2, 100))
grid_clf = GridSearchCV(pipeline, param_grid, cv=3, verbose=2)
grid_clf.fit(X_train, y_train)
```

```
>>> grid_clf.best_params_
{'kmeans__n_clusters': 90}
>>> grid_clf.score(X_test, y_test)
0.9844444444444445
```

Neural Networks

Prof. Gheith Abandah
Edited by: Dr. Mohammad Alhammouri

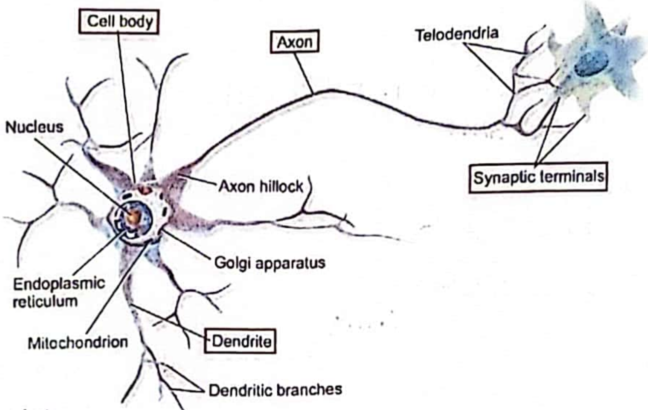
Outline

1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs

1. Introduction

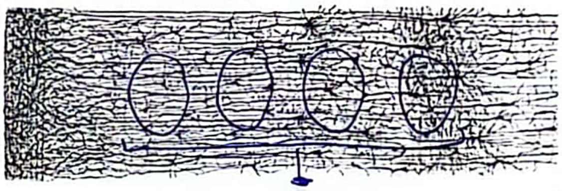
- **Artificial neural networks** (ANNs) are inspired by the brain's architecture.
- First suggested in 1943. Is now **flourishing** due to the availability of:
  - Data
  - Computing power ↑
  - Better algorithms

السبب 6 لأنه الـ NN بدها Computing power عالي



(مثل NN)

Input →



output

learning

عبارة عن اكثر من (layer)

3

الـ NN ← (very complicated)

الـ learning يتم للأكثر من مرحلة 16 كونه layer بمواصفات عالية عشان نعمل processing عليها خاصة اذا كانت عندي (images)

(deep learning)

← كل layer لحالها (learning) للـ(Feature) معين وهكذا

مثل (NN) لتشغيل الـ GPU مش الـ CPU العادي

مم تتكون الـ NN ؟!

1 ⇐ z>0
0 ⇐ z<0

2. The Perceptron

(step. Fun.)

[y = step(x1w1 + x2w2 + x3w3)]

المعادلة النهائية للـ (y)

- The **Perceptron** is a simple ANN, invented in 1957 and can perform linear (binary classification) or regression.

تتعامل as

(binary classifier) ما وظيفتها (y)

اذا الخطأ اي قيمة سالبة يعطيني (0) / اي قيمة موجبة يعطيني (1)

- **TLUs** are the building blocks of single-layer (perceptrons) which are the simplest form of neural networks.

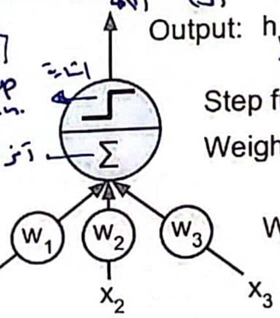
ممكن يكون اكثر من (TLU) جنب بعض اذا one layer اذا multi layer

(Sum.) (3)

تضرب كل Inp. بـ (W) التابعة الها تجمعهم مع بعض

$W_1 . X_1 + W_2 . X_2 + \dots$

(TLU: Threshold logic unit)



(1) Step Fun.

اشارة

Output: $h_w(\mathbf{x}) = \text{step}(\mathbf{w}^t . \mathbf{x})$

Step function: step(z)

Weighted sum: $z = \mathbf{w}^t . \mathbf{x}$

Weights (2)

Inputs (1)

مكونة من ① output neural ② Input ③ weight

دمج الـ basic يلي بستخدمه في بناء الـ (Percep.)

- Common **step function**:

ح تشوف قدام كمان

$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} \qquad \text{sgn}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

4

Functions →
- logistic ⇒ قيمة بين الـ (0) والـ (1)
- Sgn ⇒ قيمة بين (-1) والـ (1)
- step

output digit 2 ← بفعل اذا اولا

digit ③ ← بفعل اذا ال output ثالثا

digit ② digit ①

ال (N) بحكيل انا صادي ال digit ① اولا

(3-class) ← ③ ② ①

output

2. The Perceptron

bias ال بتحكم فيو شان

- The Perceptron has an **input layer** with **bias** and **output layer**.

اوجد ال output زي ما بدي

- With **multiple output nodes** it can perform multiclass classification. (بعمل) ← للاستفادة منه
- Hebbian learning "**Cells that fire together, wire together.**"

بحكي عن (connection weight) between two neural is increase

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta\left(y_j - \hat{y}_j\right)x_i$$

new weight · old weight · فرق output

$\hat{y}_j$ is the output of the j^{th}

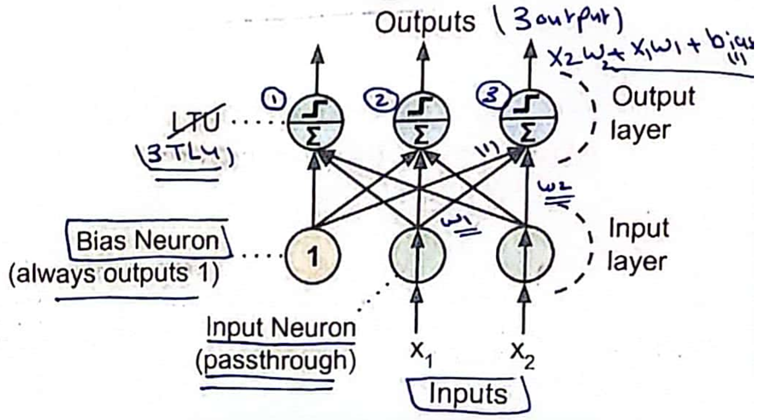
y_j is the target output of

لحال ال output بيزيد weight لحالها بزيد

في ال NN ال (W) ما ضل ثابت ظل تتغير

اول ال weight داخل من عملية parameter tweaking ←

كل TLU (one output) اصلا عند



A Perceptron is simply composed of a single layer of TLUs, with each TLU connected to all the inputs

كل TLU لازم تكون مشبوكة مع كل ال (Input)

5

وهاد الهدف من (NN) مش بس Input واحد مربوط مع output!

كل output مربوط مع كل ال Input

تصير عملية ال learning اكثر complex

← كل ال Hidden layer بتكون من (TLU) مشبوكة مع كل ال Input ومع كل ال output ومكررة اكثر من مرة لبين ما يصير deep (learning network)

2. The Perceptron

- Scikit-Learn provides a **Perceptron** class.

iris data set فيها Feature (4) بس نستخدم ال two Feature

iris data set فيها class (3) يعني ال target (3)

ال target ال column فيه (3) category

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron

iris = load_iris()
X = iris.data[:, (2, 3)]  # petal length, petal width
y = (iris.target == 0).astype(np.int)  # Iris Setosa?

per_clf = Perceptron(random_state=42)
per_clf.fit(X, y)

y_pred = per_clf.predict([[2, 0.5]])
```

(كل rows بالعامود 2,3)

prediction

اذا petal length = 2، اذا petal width = 0.5 شو قيمة (y-pred)؟

(Input Output layer) y · ŷ (new (y))

y	ŷ
0	1
1	0
2	0
0	1
1	0
2	0
2	0
1	0
0	1

ما هو (iris setosa) ال 0 = Encoding بدي اعمل binary classification بلعب ال class مقطعها (0) all encoding فصار في y (new y) 1,0

0 : another class
1 : class 0

target = 0 → T
target ≠ 0 → F

6

بحل مشكلة الـ (XOR)

	Xor
0 0	0
0 1	1
1 0	1
1 1	0

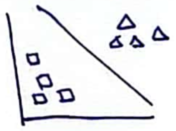
لو رسمناها ح يطلع زي تحت

2. The Perceptron

non-linear problem تعتبر xor problem

- The perceptron **cannot solve non-linear problems (like xor).** The **Multi-Layer Perceptron** (MLP) can.

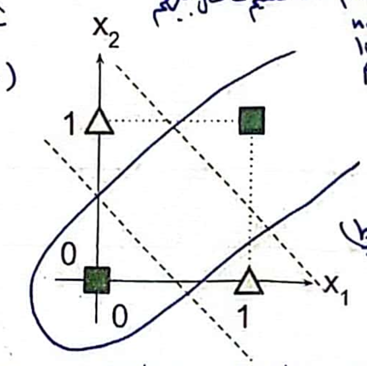
Ex)

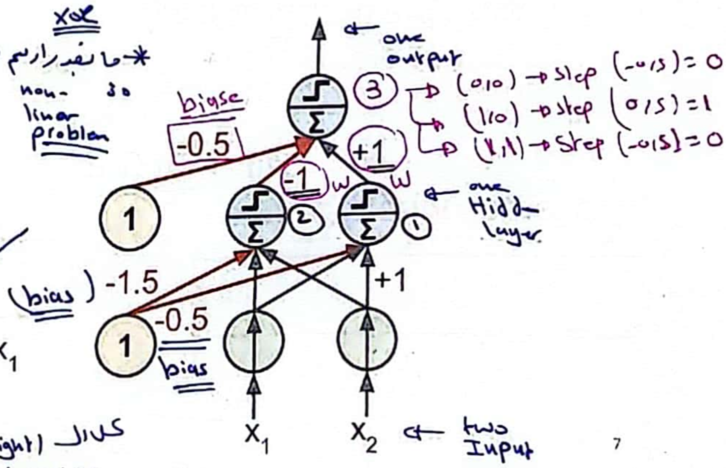


(linear problem) مشكلة نقدر نرسم خط يفصل بينهم !

xor

* ما نقدر نرسم خط مستقيم لافصل بينهم — so non-linear problem





كل الـ (weight) عبارة عن (1) الا الموجودين الاحمر بتوضيحة لـ (biase) بلي عليهم في القيم الموجودة weight

← لما يكون (0,0) لعطيني (0)
← لما يكون (1,0) لعطيني (1)
← لما يكون (1,1) لعطيني (0)
هذا حل المشكلة الـ (non-linear)

②
$x_1 = x_2 = 0$ (0,0)
$x_2 \cdot 1 + x_1 \cdot 1 - 1.5 = -1.5$
step (-1.5) = 0 (Input)

(1,0)
$x_2 \cdot 1 + x_1 \cdot 1 - 1.5 =$
$0 + 1 - 1.5 = -0.5$
step (-0.5) = 0 (Input)

(1,1)
$x_2 \cdot 1 + x_1 \cdot 1 - 1.5 =$
$1 + 1 - 1.5 = 0.5$
step (0.5) = 1 (Input)

①
$(x_1 = x_2 = 0)$ Initial (0,0)
$x_2 \cdot 1 + x_1 \cdot 1 - 0.5 = -0.5$
Step (-0.5) = 0 (Input)

(x_1, x_2)
(1,0)
$x_2 \cdot 1 + x_1 \cdot 1 - 0.5 =$
$0 + 1 - 0.5 = 0.5$
Step (0.5) = 1 (Input)

(1,1)
$x_2 \cdot 1 + x_1 \cdot 1 - 0.5 =$
$1 + 1 - 0.5 = 1.5$
Step (1.5) = 1 (Input)

Outline

1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs

طريقة (3)

Input x weight + biase

② ← (0,0) → ①
0 X -1   1 X 0
-0.5 (bias)
0 - 0 - 0.5 = -0.5

② ← (1,0) → ①
0 X -1   1 X 1
-0.5
1 - 0 - 0.5 = 0.5

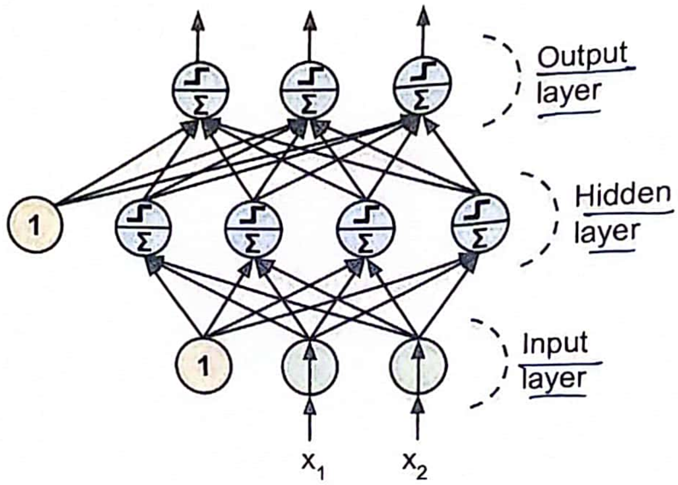
② ← (1,1) → ①
1 X -1   1 X 1
-0.5
1 X 1 - 1 X 1 - 0.5 = -0.5

يعني عندي اكثر من (hidden layer) هم

3. Multi-Layer Perceptron (MLP)

- An MLP is composed of a (pass-through) **input layer**, one or more layers of TLUs, called **hidden layers**, and a final layer of TLUs called the **output layer**.
- When an ANN has **two or more** hidden layers, it is called a **deep neural network** (DNN).

يعني عندي اكثر من (hidden layer)

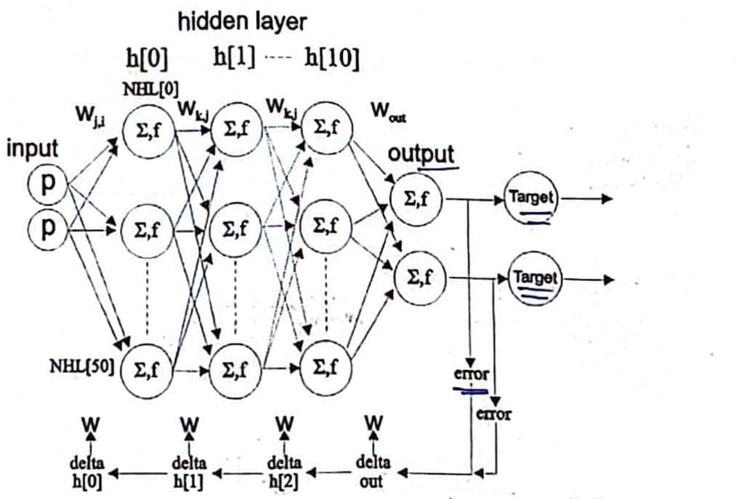


3. Multi-Layer Perceptron (MLP)

- Trained using the **backpropagation training algorithm.**
  - For each training instance the algorithm first makes a (prediction) (**forward pass**), measures the error,
  - then goes through each layer in reverse to measure the error contribution from each connection (**reverse pass**),
  - and finally slightly tweaks the connection weights to reduce the error (**Gradient Descent step**).

لجدما يحسب ال output يرجع يعدل ال weight بحيث (new input)

يحسب ال output لشوف الفرق وهكذا لحد ما يحصل على optimal weight



(output - target = error)

(update)

لتستفيد من (error) لتعديل على (weight)

ليحصل على (min error) لحد ما يعمل update كل (w) لحتى يحصل اقل error ويوقف

بعد كل عملية (output) برجع بحسب ال error لكل hidden layer بحسب الفرق بين ال target value و ال predicted value برجع لتعدل ال (error)

Outline

1. Introduction
2. The perceptron
3. Multi-layer perceptron (MLP)
4. Regression MLPs
5. Classification MLPs

4. Regression MLPs

- Typical MLP architecture for **regression**:

Hyperparameter	Typical Value
# input neurons	One per input feature (e.g., 28 x 28 = 784 for MNIST)
# hidden layers	Depends on the problem. Typically 1 to 5.
# neurons per hidden layer	Depends on the problem. Typically 10 to 100.
# output neurons	1 per prediction dimension
Hidden activation	ReLU (or SELU, see Chapter 11)
Output activation	None or ReLU/Softplus (if positive outputs) or Logistic/Tanh (if bounded outputs)
Loss function	MSE or MAE/Huber (if outliers)

نفس عدد الـ Input من data set

كل Feature بدها (Input neuron)

5 (simple reg.) حسب الـ problem

حسب الـ problem → hidden لعبة تصميم

توقع واحد (1) طلوعية يلي هي عملية التوقع

موجود مش TLU

الأفضل لتوقع معرفة الإجابة سالبة (-) ← Relu
حليا (0) اي قيمة موجبة (+)
حليا زي ما هي

لسه ما تستخدم (step) في (Regression)

الـ (regression) بعكس الـ (classification) ما بدي (1/0) بدي رقم

(من ناتج الـ error)

في regression بستخدم (MSE) لنطلع الـ error بناء عليه update على (الأوزان weight)

5. Classification MLPs

بستخدم ال step Fun. Ex) → بعطيني الجواب (0 او 1) بدي one neural

- MLPs can also be used for classification tasks
- A binary classification problem, you just need a single output neuron using the logistic activation function: the output will be a number between 0 and 1

ham / spam
not / urgent → two → بدي ال output neural بعدهم

- MLPs can also easily handle (multi-label binary classification) tasks
- Example: email classification system, predicts whether each incoming email is **ham** or **spam**, and simultaneously predicts whether it is an **urgent** or **non-urgent**
  - We need two output neurons, both using the logistic activation function

not ham / ham
not urgent / urgent → بدل ال email تحزر مرتين مرة ← هو كل (multi label) لكن binary classific. ∴ بكل مرة بدل تحزره بـ تشغلتين

- For multiclass classification then you need to have one output neuron per class, and you should use the **softmax** activation function (will ensure that the estimated probabilities are between 0 and 1 for each class)

→ الفرق بين multi label classification و multi class classification

لعندك (two class)

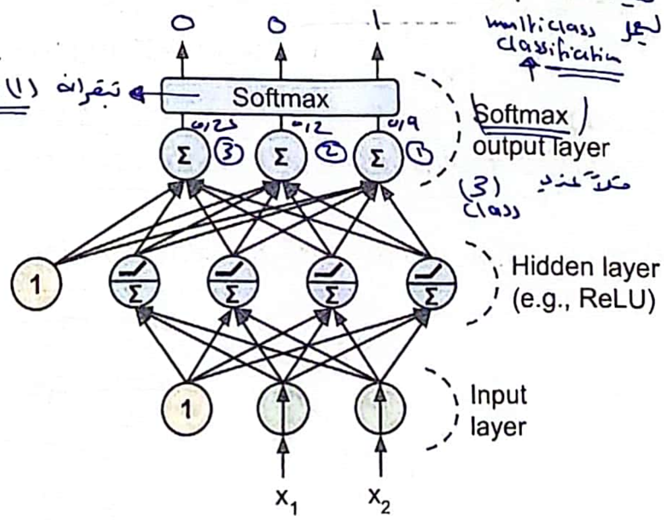
لو عندك كثير → 0 or 1

① ② ③ ④ multiclasse

بدي digit classifier لحزال
من (1) الى (4) بدي (4) (output neural)
(1) → الأول مسؤول عن الحزابة (1) او لا
(2) → الثاني — — — (2) او لا
⋮
إلخ

5. Classification MLPs

بستخدم MLP لحل multiclass classification



- For **classification**, the output layer uses the **softmax function.**
- The output of each neuron corresponds to the **estimated probability** of the corresponding class.

5. Classification MLPs

- Typical MLP architecture for **classification**:

Hyperparameter	Binary classification	Multilabel binary classification	Multiclass classification
Input and hidden layers	Same as regression	Same as regression	Same as regression
# output neurons	1	1 per label	1 per class → (10 neural بدل 10 class)
Output layer activation	Logistic (بين 0 و 1)	Logistic	Softmax
Loss function	Cross-Entropy (لو اكبر من 0.5 نعطيه (1) / لو اقل من 0.5 نعطيه (0))	Cross-Entropy	Cross-Entropy

Cross entropy loss is a metric used to measure how well a classification model in machine learning perform. The loss (or error) is measured as a number between 0 and 1, with 0 being a perfect model

0: الـ accuracy is a best

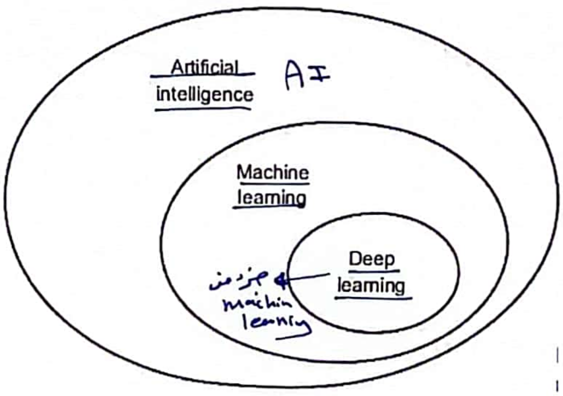
اقل قيمة افضل النتائج

6. Deep learning

(عملية الـ learning بتكون على اكثر من layer)

Deep learning is a *layered representations learning and hierarchical representations learning or multistage way to learn data representations.*

Deep learning *is a form of machine learning that uses artificial neural networks to learn*



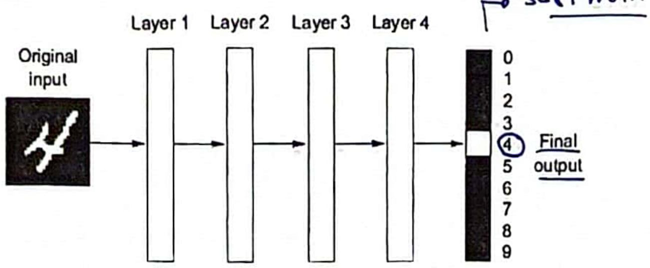
Ex) SVM → machine learning مو deep learning لكنه

MLP → machine learning و deep learning

6. Deep learning

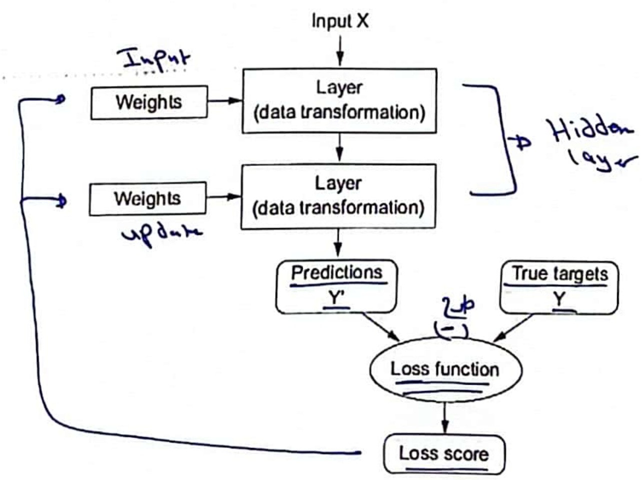
Example A deep neural network for digit classification

اذا اخذنا صورة ما بزبط امرها ب (one layer)



6. Deep learning

A loss function measures the quality of the network's output



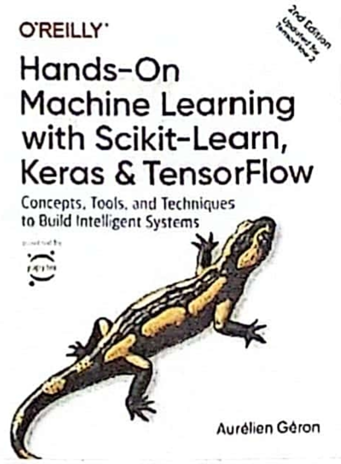
بحسب ال loss بعدين بعدّل كل (weight)

Artificial Neural Networks with Keras

Prof. Gheith Abandah

Edited by: Dr. Mohammad Al-hammouri

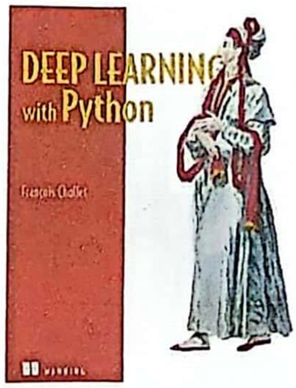
Reference



- Chapter 10: **Introduction to Artificial Neural Networks with Keras**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

Reference



- **Deep Learning with Python**, by François Chollet, Manning Pub. 2018
- **Introduction to Keras** by Francois Chollet, March 9th, 2018 (slides)

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

Introduction

- YouTube Video: *Keras Explained* from Siraj Raval

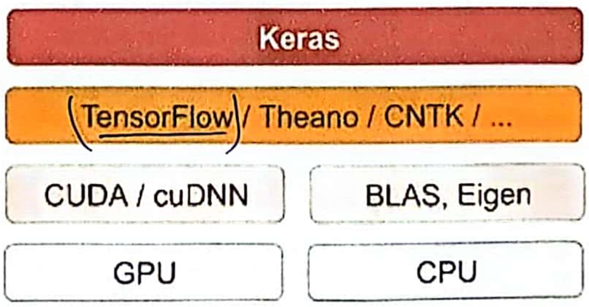
https://youtu.be/j_pJmXJwMLA

1. Introduction

(بضيف كلية الـ training كل الاخر
ثم (layers)

(use to build deep learning model)

- **Keras** is a high-level API (neural network library) to build and train deep learning models runs on top of TensorFlow.
- **TensorFlow** is an open-sourced end-to-end platform, a library for multiple machine learning tasks

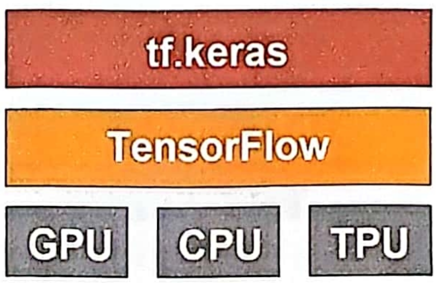


1. Introduction – Advantages (Keras)

- **User friendly**: Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors.
- **Modular and composable**: Keras models are made by connecting configurable building blocks together, with few restrictions.
- **Easy to extend**: Write custom building blocks to express new ideas for research. Create new layers, loss functions, and develop state-of-the-art models.

3. TensorFlow Keras

- Keras is the official high-level API of TensorFlow
- (موجودة داخله) tensorflow.keras (tf.keras) module
- Part of core TensorFlow since v1.4



3. TensorFlow Keras

- To install TensorFlow

```
$ pip install --upgrade tensorflow
```

- To import Keras from TensorFlow

```
>>> import tensorflow as tf
>>> from tensorflow.keras import Layers
>>> from tensorflow import keras
>>> tf.__version__
'2.1.0'
>>> keras.__version__
'2.2.4-tf'
```

- Dense
- Activations
- Dropout
- Conv1D, 2D, 3D
- Polling
- RNN, LSTM, GRU
- ...

tensflow

Ex)

4. Image Classifier Using the Sequential Model

(Keras)

- **Fashion MNIST** is similar to MNIST (70,000 grayscale images of 28×28 pixels each, with 10 classes).
- A simple linear model reaches about 92% accuracy on MNIST, but only about 83% on Fashion MNIST.

(Deep learning)

deep learning

(10 classes)

(data set)

4. Fashion MNIST

1. **Get and prepare the dataset.**
2. **Build sequential model** of layers that maps your inputs to your targets.
3. **Compile the model and configure the learning process** by choosing a loss function, an optimizer, and some metrics to monitor.
4. **Train the model** by calling the `fit()` method of your model.
5. **Evaluate and use** the model.

(MNist في Keras)

4.1 Get and Prepare the Dataset

70K Data → 60K train → 55K train, 5K Valid; 10KK test

```
import tensorflow as tf
from tensorflow import keras

# Get the Fashion MNIST
fashion_mnist = keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) =
        fashion_mnist.load_data()
```

جزء الـ load (text ,train) كله تقسيم

```
# Prepare the data train (55000), val (5000), test (10000)
X_valid = X_train_full[:5000] / 255.
X_train = X_train_full[5000:] / 255.
y_valid, y_train = y_train_full[:5000], y_train_full[5000:]
X_test = X_test / 255.
```

```
>>> X_train_full.shape
(60000, 28, 28)
>>> X_train_full.dtype
dtype('uint8')
```

→ (÷255) → normalization بدل تغيرته ف (0 لـ 1) → Pixel الغرفية منه (0-1) → كان الأكثر Pixel (255) اول (0) عاملتم كل 255 تعمل ل (1)

في شئ لسيس (deep learning)

← اول (5K) لـ valid من X-train

← بأخذ 5K النهاية (55K)

نفس الشيء لـ (y)

[Validation Value]

Validation في testing عزال

← تستخدم الـ (Validation)

لـ شان تحسن اداء الـ model

يعني هنا (training data) الـ model عامد بتعلم

بحساب الـ (Coefficient) كلها وكل (Round) في training

test الـ model على (Validation data) عشان نتأكد

بعدها يجعل اذا تتعلم صح

(during testing) ثم (during training)

← الـ Validation data ← حجمها قليل

4.2 Build the Model

The default is no activation function, i.e., linear layer.

```
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
```

① Seq. model ② layer First

layers الـ NN في

Flatten الـ كـ Input layer ليس

(عدد الـ feature = shap) الـ

(28 x 28) feature = shap الـ

لو عندي 20 feature (20 = shap) الـ

عدد الـ output في NN = عدد classes

```
>>> model.summary()
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
dense_3 (Dense)              (None, 300)               235500
dense_4 (Dense)              (None, 100)               30100
dense_5 (Dense)              (None, 10)                1010
=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
```


الـ Hidden layer الـ optimum (1-4)

عدد N في Hidden الاول 300

عدد N في Hidden الثاني = 100

two hidden

لو في binary classification عدد الـ (NN) في output layer = (1)

في multiclass classific. نحتاج كبير عدد الـ output الـ نحتاج class

كل output الـ (10) بيجيهم Input من الـ Hidden (100) 100x10 (+bias)

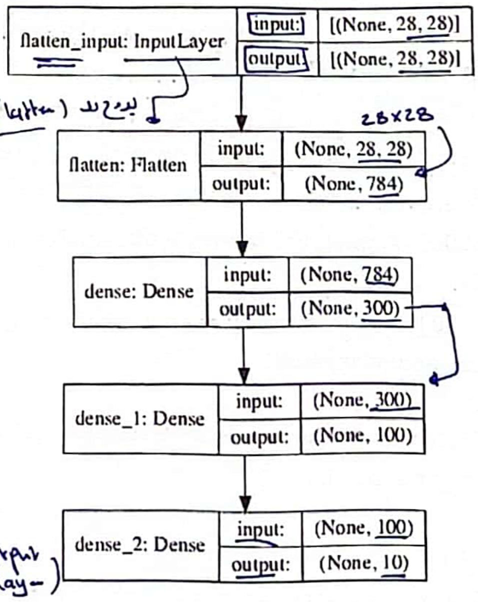
10 output لعطيني 10 class

الـ Activation Fun. في output layer: في حال multiclass classification نستخدم الـ (Softmax); اذا عندي binary classification نستخدم الـ (Step Function) او (logistic Fun.) تقطعني 0 1

الـ Activation Fun. داخل Hidden layer ← عادة نستخدم الـ (Relu)

4.2 Build the Model

```
# Plot the model
keras.utils.plot_model(
    model,
    "my_model.png",
    show_shapes=True
)
```



(Input layer) في الطلوب (N) مهم تعرف كم عدد

(output layer) في الطلوب (N) وكم عدد

4.3 Compile the Model

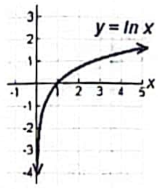
loss Fun. ⇒ ~~accuracy~~ error

```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])

# For sparse labels (0-9):
loss = "sparse_categorical_crossentropy"
# For one-hot labels:
loss = "categorical_crossentropy"
# For binary labels:
loss = "binary_crossentropy"
# For regression:
loss = "mean_squared_error"
```

Stochastic Gradient Descent

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$



4.4 Train the Model

```
# Train the model
history = model.fit(X_train, y_train, epochs=30,
                    validation_data=(X_valid, y_valid))
```

Train on 55000 samples, validate on 5000 samples

Epoch 1/30

55000/55000 [==============================] - 2s 44us/sample - loss: 0.7226 - accuracy: 0.7641 - val_loss: 0.5073 - val_accuracy: 0.8320

Epoch 2/30

55000/55000 [==============================] - 2s 39us/sample - loss: 0.4844 - accuracy: 0.8321 - val_loss: 0.4541 - val_accuracy: 0.8478

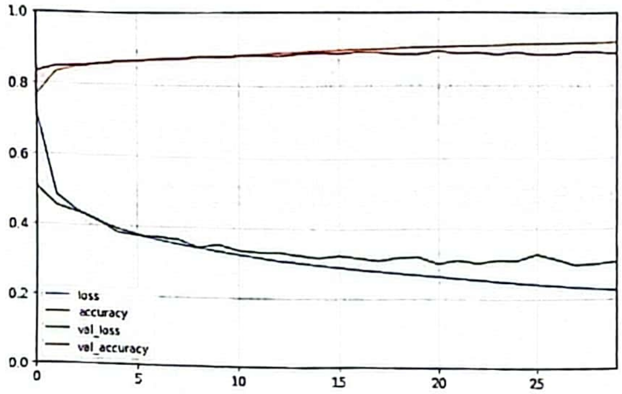
...

Epoch 30/30

55000/55000 [==============================] - 2s 39us/sample - loss: 0.2256 - accuracy: 0.9195 - val_loss: 0.3049 - val_accuracy: 0.8882

4.4 Train the Model

```
import pandas as pd
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
save_fig("keras_learning_curves_plot")
plt.show()
```



4.5 Evaluate and Use the Model

```
model.evaluate(X_test, y_test)
10000/10000 [==============================] - 0s 21us/sample - loss: 0.3378 -
accuracy: 0.8781
[0.33780701770782473, 0.8781]
```

ف testing

```
X_new = X_test[:3]
y_proba = model.predict(X_new)
y_proba.round(2)
array([[0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.01, 0.  , 0.99],
       [0.  , 0.  , 0.99, 0.  , 0.01, 0.  , 0.  , 0.  , 0.  , 0.  ],
       [0.  , 1.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ]],
      dtype=float32)
```

بدي اول ٣ بالسطر

(10 classes) (output soft max) (ح تتصنف لآخر class)

```
model.predict_classes(X_new)
array([9, 2, 1])
```

5. Example - MNIST

Ex
حليه

1. **Define your training data**: input tensors and target tensors.
2. **Define a network** of layers (or **model**) that maps your inputs to your targets.
3. **Configure the learning process** by choosing a loss function, an optimizer, and some metrics to monitor.
4. **Iterate on your training data** by calling the `fit()` method of your model.

5. Example – Prepare the data

```
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) =
    mnist.load_data()
#(60000, 28, 28), (60000), #(10000, 28, 28), (10000)
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255
```

train test 70K Dataset

لتقسيم ال Data train test

(normali.) ال Input كونه Images عمله reshape

```
from keras.utils import to_categorical  #one hot
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

target value جزء train.

target value جزء ال test

بدل ال one column صار كل عدد في target value

0 1 9 7 | 1 | 0 | 0 | 0 ...

3's representation is:

[0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

* ال (validation) اضافات لل

Additional Parameter 1 في ال Fit

(valid.) ال تضاف

(مش ضرورية لكن مفيدة)

5. Example – Define and configure the network

```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

Hidden first

(Input layer) (Input ... Implicitly)

Feature or Weights

10 = output

use for multiclass classification

Input layer

(Flatten)

5. Example – Training and evaluation

```
network.fit(train_images, train_labels, epochs=5, batch_size=128)
```

```
Epoch 1/5
60000/60000 [==============================] - 2s - loss: 0.2577 - acc: 0.9245
Epoch 2/5
60000/60000 [==============================] - 1s - loss: 0.1042 - acc: 0.9690
Epoch 3/5
60000/60000 [==============================] - 1s - loss: 0.0687 - acc: 0.9793
Epoch 4/5
60000/60000 [==============================] - 1s - loss: 0.0508 - acc: 0.9848
Epoch 5/5
60000/60000 [==============================] - 1s - loss: 0.0382 - acc: 0.9890
```

```
test_loss, test_acc = network.evaluate(test_images, test_labels)
```

```
9536/10000 [===========================>..] - ETA: 0s
```

$\boxed{\texttt{test\_acc: 0.9777}}$ ← Keras

(liner model) 92% simple model

6. Regression Using the Sequential Model

- Solve the **California housing** problem using a regression neural network.
- Scikit-Learn has `fetch_california_housing()` function to load the data
- This dataset contains **only numerical features** and there are **no missing values.**

6.1 Get and Prepare the Dataset

هنا من انا نقل (Regression) على housing "Data Set" (deep learning model)

```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

housing = fetch_california_housing()

X_train_full, X_test, y_train_full, y_test =
      train_test_split(housing.data, housing.target, random_state=42)
X_train, X_valid, y_train, y_valid = train_test_split(X_train_full,
                     y_train_full, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_valid = scaler.transform(X_valid)
X_test = scaler.transform(X_test)
```

بعمل Fetching لـ (California dataset) الموجودة في Sklearn وتخزنه

The default is 75% : 25%

اخذت جزء الـ train
جزء الـ test
جزء الـ (Validation)

(Data normalization الـ Standardization عامل) (deep learning) في (train) لعملية

بدي ابني الـ model عندي (Sequential model)

الـ (seq.) لعندي في عندي (layers) الترتيب تجمع (seq.)
لعندي أول layer بدي الـ Input تاع تاني layer والخ (Output)
هنا two layer اخر اتنين الـ (Hidden layer) (Output layer)
كل الـ layer السابقة تابع لللاحق

6.2 Build and Compile the Model

ال Hidden layer فيه (30 nural) كل واحد فيهم بيجي Input !?

Since the dataset is quite noisy, we just use a single hidden layer with fewer neurons than before, to avoid overfitting

```
# Building by passing a list of layers when creating
#  the Sequential model
model = keras.models.Sequential([
    keras.layers.Dense(30, activation="relu",
        input_shape=X_train.shape[1:]),
    keras.layers.Dense(1)])

# Compile with creating an optimizer object
model.compile(loss="mean_squared_error",
    optimizer=keras.optimizers.SGD(lr=1e-3))
```

two layer

Hidden layer

(output layer)

Regression problem

(MSE)

Implicitly is defind

(multiclass classification)

Input layer

(Data set)

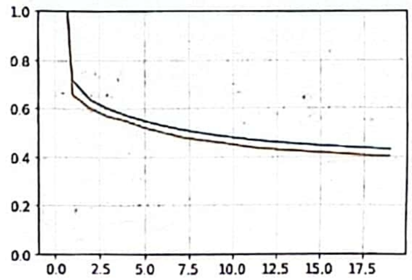
(30 nural) (one Hidden layer)

(Input layer) (Hidden layer) (deep layer)

6.3 Train and Evaluate the Model

(fit) (accuracy)

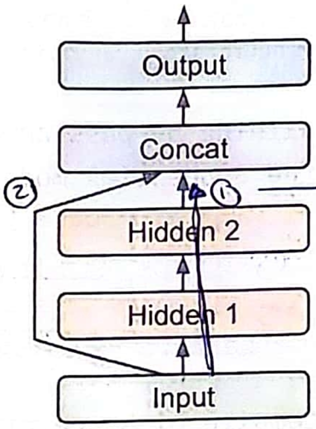
```
history = model.fit(X_train, y_train, epochs=20,
        validation_data=(X_valid, y_valid))
```



```
mse_test = model.evaluate(X_test, y_test)
5160/5160 [==============================] - 0s
15us/sample - loss: 0.421
```

Building Complex Models Using the Functional API

- Keras functional API can be used to build arbitrary **static graph topologies**.
- **wide and deep** network that learns both deep patterns (using the deep path) and simple rules (through the short path).
- Regular MLP forces all the data to flow through the full stack of layers
  - Simple patterns in the data may end up being distorted by this sequence of transformations.



mlp: multi layer perceptron or (seq. model)

Complex Model: جزء من Input مايمر ب (Hidden layer)

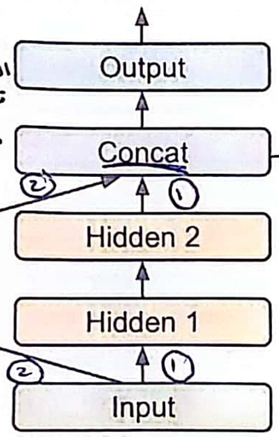
[Deep learning + wide learning]

Complex Models Using the Functional API (Example)

```
input = keras.layers.Input(shape=X_train.shape[1:])
hidden1 = keras.layers.Dense(30, activation="relu")(input)
hidden2 = keras.layers.Dense(30, activation="relu")(hidden1)
concat = keras.layers.Concatenate()([input, hidden2])
output = keras.layers.Dense(1)(concat)
model = keras.models.Model(inputs=[input], outputs=[output])
```

→ Input layer

Ex) = [10, 8]



(output + Concat + two hidden layer + Input)

Using the Functional API (Multi-input)

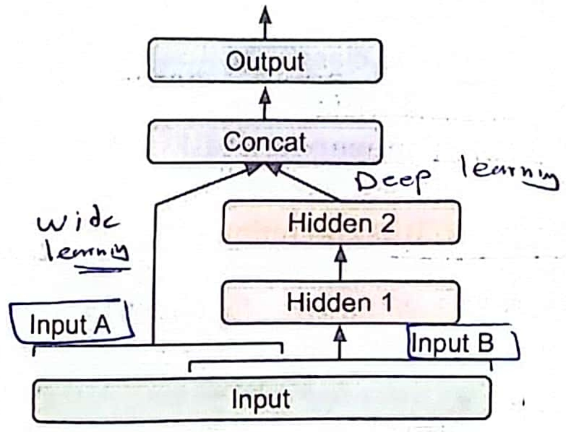
(multiple output) +ممكن يكون

Multi-input: You can send a subset of the features through the wide path and a different subset (possibly overlapping) through the deep path.

المقصود فيه يلي بروح مباشرة لل Concat

المقصود فيه عن طريق ال (hidden layer)

wide / deep ← path لترسل اي Input انا بقسم ال *



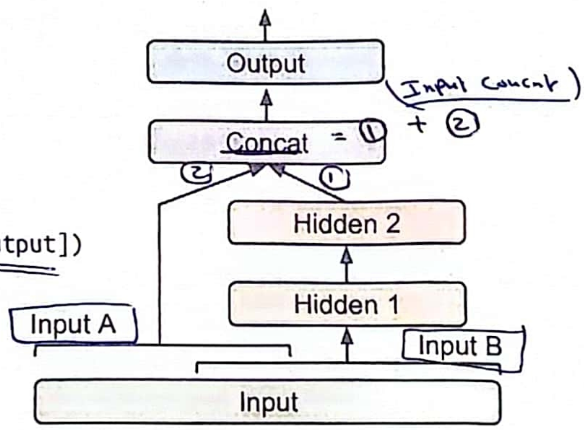
Using the Functional API (Multi-input)

نبني ⇐ (Complex model) عشان كلام الهم (parameter \ Input) أو

(5) Feature

```
input_A = keras.layers.Input(shape=[5])
input_B = keras.layers.Input(shape=[6])
hidden1 = keras.layers.Dense(30, activation="relu")(input_B)
hidden2 = keras.layers.Dense(30, activation="relu")(hidden1)
concat = keras.layers.concatenate([input_A, hidden2])
output = keras.layers.Dense(1)(concat)
model = keras.models.Model(inputs=[input_A, input_B], outputs=[output])
```

بعد ال model

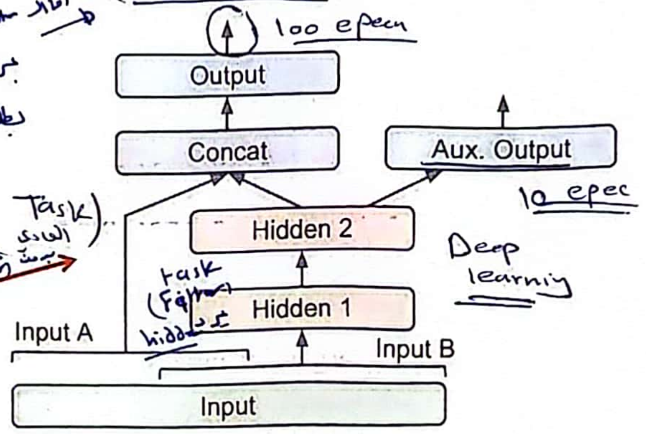


(multiple output) شو الفائدة من
ال (Aux. output) لـ Feature بدي اعرفه اذا موجود لحص

Using the Functional API (Multiple Outputs)

لما يكون عنا multiple Input

Multiple Outputs



(العادية Feature) 100 epoch

10 epoc

Deep learning

- To locate and classify the main object in a picture.
- Multiple independent tasks to perform based on the same data.
- Regularization technique (to ensure that the deep network learns something useful on its own, ex: Aux output).

goal (reduce overfitting)

Regularization technique (i.e., a training constraint whose objective is to reduce overfitting and thus improve the model's ability to generalize)

31

(overfitting) model

(two output)

7.1 Auxiliary Output for Regularization

```
# Build the model
input_A = keras.layers.Input(shape=[5], name="wide_input")
input_B = keras.layers.Input(shape=[6], name="deep_input")

hidden1 = keras.layers.Dense(30, activation="relu")(input_B)
hidden2 = keras.layers.Dense(30, activation="relu")(hidden1)

concat = keras.layers.concatenate([input_A, hidden2])

output = keras.layers.Dense(1, name="main_output")(concat)

aux_output = keras.layers.Dense(1, name="aux_output")(hidden2)

model = keras.models.Model(inputs=[input_A, input_B],
                           outputs=[output, aux_output])
```

32

7.1 Auxiliary Output for Regularization

[:, :5] X-train A

كل ال row (0,1,2,3,4) من البداية لحد ال Col (ا) (Feature)

X-train B نرجع كل row لحد ال Column من عند ال Feature (2) لحد النهاية

بحضرال (Data) (X-train) العادية مقسمة لـ

```
# Split the input
X_train_A, X_train_B = X_train[:, :5], X_train[:, 2:]
X_valid_A, X_valid_B = X_valid[:, :5], X_valid[:, 2:]
X_test_A,  X_test_B  = X_test[:, :5],  X_test[:, 2:]

# Take some test samples
X_new_A, X_new_B = X_test_A[:3], X_test_B[:3]
```

الهدف انه يكون عندي multiple Input انا بأجزء نفس (Data set) لجزئين بس الفكرة مايجوز من كل Feature جزء من Feature Input A و Input B ممكن يكون في نوع overlab لبعض ال Column يكون مشترك بين (two input)

نعمل نفس الشي لل Valid وال test

7.1 Auxiliary Output for Regularization

(نعمل fit و Compile و evaluate زي العادي)

بالانخذ عرض يكون ال (two input) A B } جزئين Valid Set وال test نجد جزئين

ده ليس يكون عندك multiple input بل two diff. Valid Set + two diff. test set

```
# Compile, train, evaluate, and predict
model.compile(loss=["mse", "mse"], loss_weights=[0.9, 0.1],
       optimizer=keras.optimizers.SGD(lr=1e-3))

history = model.fit([X_train_A, X_train_B], [y_train, y_train], epochs=20,
       validation_data=([X_valid_A, X_valid_B], [y_valid, y_valid]))

total_loss, main_loss, aux_loss = model.evaluate([X_test_A, X_test_B],
                                                  [y_test, y_test])

y_pred_main, y_pred_aux = model.predict([X_new_A, X_new_B])
```

Save and Restore the Model

• After training a model save it to a file.

```
model.save("my_keras_model.h5")
```

• In the production program, load the trained model (load the model and use it to make predictions)

```
model = keras.models.load_model("my_keras_model.h5")
```

• What if training lasts several hours?

you should not only save your model at the end of training, but also save checkpoints at regular intervals during training

35

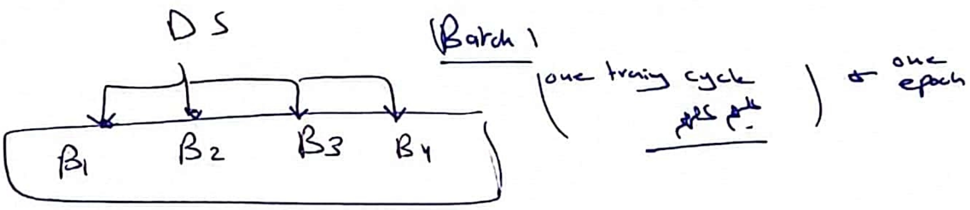
8. Using Callbacks

• The **fit()** method accepts a **callbacks** argument that lets you specify a list of objects that Keras will call during training

- at the start and end of **training**
- at the start and end of each **epoch**
- before and after processing each **batch**

• There are many callbacks available in the **keras.callbacks** package. See

https://keras.io/callbacks/



36

8.1 Saving Best Model

- **Save your best model** when its performance on the validation set is the best so far.

ليسيڤ ال model عن طريق (Call back)

```
checkpoint_cb = keras.callbacks.ModelCheckpoint(
    "my_keras_model.h5", save_best_only=True)
history = model.fit(X_train, y_train, epochs=10,
          validation_data=(X_valid, y_valid),
          callbacks=[checkpoint_cb])

# rollback to best model
model = keras.models.load_model("my_keras_model.h5")
mse_test = model.evaluate(X_test, y_test)
```

(file اسم ال)

ب accuracy — لا نحفظ ليسيڤ — اذا ما رح improvement بال accuracy

(Array)

ال سيڤنا model قبل الآخر جزء

** لا يجيب من File افضل model افضل accuracy

يعني انت ما رح تعمل عملية training ب test لتشوف
من training عن طريق ال valid اذا انت ماشفت تحسين او تحسن في performance
لحالة عملية ال training تتوقف عن طريق ال Feature
بلي اسمها Early stopping

Early Stopping

يسمى ب لانه بتسمح ال regularization يكون عندنا اكبر من output

- Early stopping is a regularization technique used to prevent overfitting by monitoring the model's performance on a validation set during training.

يعني ليبقى ال model بمرحلة معينة يطلع Acc output

- When the performance stops improving for a specified number of epochs, the training is stopped.

بلي تشتغل بليها Early stopping

- This helps to ensure that the model does not train too long and start to overfit the training data.

لحتى نختصر الوقت لنحله يكمل كل ال epoch ويطلع ال output الرئيسي

سترة ال Ex

لما ادرب ال model عن epoch كثيرة بلي يجرحني model
حتى دا اذا تعدال ال epoch الكثير تعطيني نتائج افضل overfitting
فـ ال Early stopping بتخليك توقف

لـ تشتغل عن طريق Feature تسمى patience

Ex عملت ال patience = 5 ، لجرب ال model انه لحقق نتائج
لـ 5 مرات متتالية ، اذا صار ان 5 وما حقق نعمل Early stop

Ex بدي ادرب ال model على
100 epoch ب iteration ال epoch
رقم 50 وجد انه ما في تحسن في
model بدي يطلع ال model
عن طريق ال Aux output
ونختصر الوقت بدل ما يكمل ال 100
ويعطيني ال output ، اساسي
لذلك تسمى ال Early stopping ب regularization technique فيها اكثر من output

38

8.2 Early Stopping

له هدف goal لا يوفر وقت + يحل مشكلة ال overfit
لما انا استفيد ال model وبدي ارجعه

- The **patience** argument in the EarlyStopping callback of Keras specifies the number of epochs with no improvement after which training will be stopped. Essentially, it sets a tolerance for how many epochs the model can continue training without seeing an improvement in the monitored metric (e.g., validation loss).

ما يكمل للآخر عملية ال training ليس يكمل لآخر ما يلاقي Early stopping

```
[early_stopping_cb]= keras.callbacks.EarlyStopping(
            patience=10, restore_best_weights=True)
```

لما يجرب 10 مرات انه ما في تحسن في performance بعدها يوقف فقط

```
# Train the model with early stopping
history = model.fit(X_train, y_train, epochs=100,
            validation_data=(X_valid, y_valid),
            callbacks=[checkpoint_cb, early_stopping_cb])
```

model العادي

the best model

لـ رح يخزن فيها نتائج ال model
عن اكثر مرحلة بلي هي the best model
وال model العادي

39