



JorgeMunnozAguado/TALLER\_PYTHON



## Parte 1

- Introducción al lenguaje.
- Variables y tipos de datos.
- Bucles.
- Condiciones.
- Funciones.

## Parte 2

- Orientación a objetos.
- Creación de entorno.
- Numpy
- Matplotlib
- Sklearn
- Keras / Tensorflow
- Matplotlib
- Pandas



# python<sup>TM</sup>

## Parte 1

# Introducción al lenguaje

- ¿Qué es?
- Versiones
- Instalación
- Jupyter Notebook

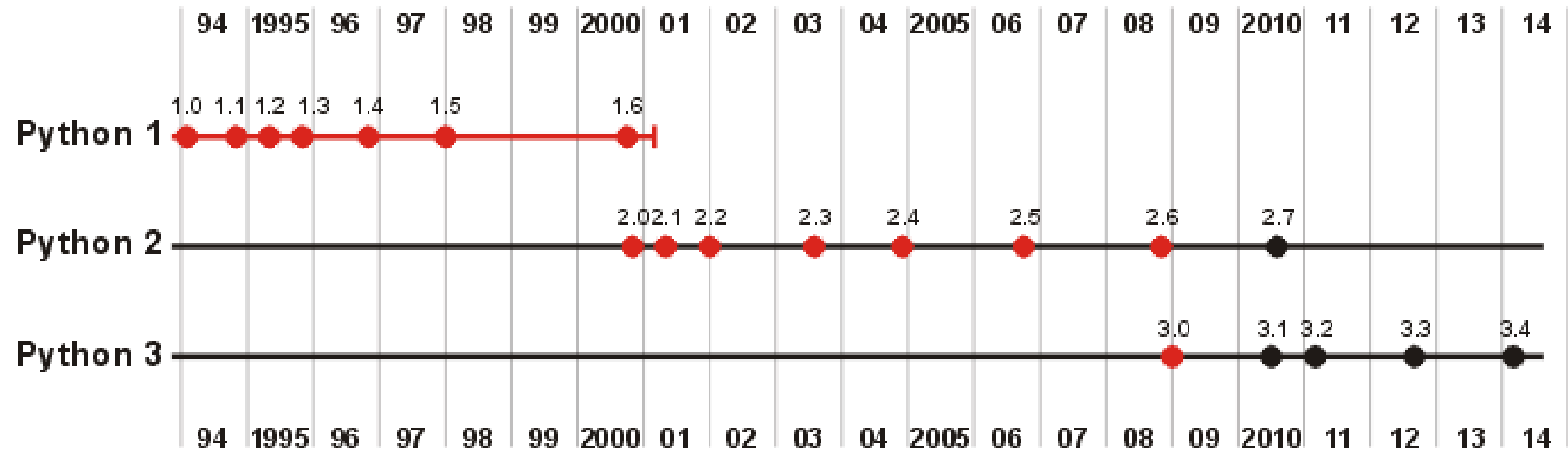
# ¿Qué es?

- Lenguaje de programación
  - > Interpretado
- Multiplataforma
- Lenguaje no tipado
- Orientado a objetos

.py

.pyc

# Versiones



# Uso e Instalación

```
sudo apt update  
sudo apt install python3
```

Ubuntu

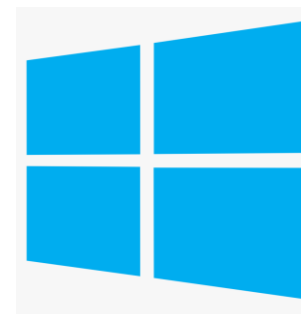


```
sudo apt update  
pip install jupyter
```

Jupyter Notebook



Windows





# Jupyter Notebook

## Memoria

```
In [1]: from skimage import transform as tf

import matplotlib.pyplot as plt
import numpy as np
import os

classes = {'a':0, 'e':0, 'i':0, 'o':0, 'u':0, 'A':0, 'E':0, 'I':0, 'O':0, 'U':0}

dirname = "out/"
```

## Datos

En este apartado analizaremos los datos, las clases.

```
In [2]: images = []

for filename in os.listdir(dirname):
    images.append( plt.imread(dirname + filename) )
    classes[filename[-5]] += 1
```

En la siguiente imagen se observan imágenes de ejemplo de cada clase.

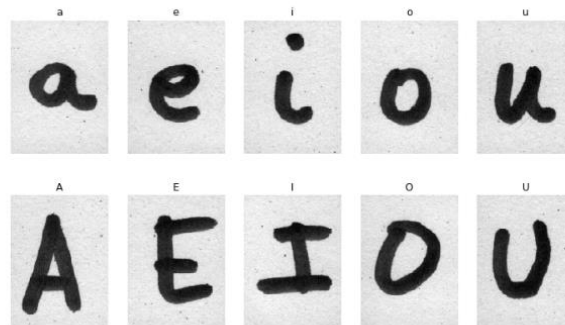
```
In [3]: fig, ax = plt.subplots(nrows=2, ncols=5, figsize=(12, 7))

for i, cls in enumerate(classes.keys()):
    image = plt.imread(dirname + '0000' + str(i) + '_' + cls + '.png')

    if i >= 5:
        x = 1
        y = i - 5
    else:
        x = 0
        y = i

    ax[x, y].imshow(image, cmap="gray")
    ax[x, y].axis('off')
    ax[x, y].set_title(cls)

plt.show()
```



Observamos que el número de ejemplos para cada clase es el mismo, en concreto de 110.

# Variables y tipos de datos

- Inicialización de variables
- Operaciones
- Listas
- Strings
- Diccionarios
- (Ficheros)

```
2  
3     variable = 1001  
4     variable = 10.01  
5     variable = 10000000000000000000000  
6     variable = "Hola Mundo"  
7     variable = True  
8
```

# Operaciones

```
9
10  n1 = 10
11  n2 = 2
12
13  r1 = n1 + n2
14  r2 = n1 - n2
15  r3 = n1 * n2
16  r4 = n1 / n2
17  r5 = n1 % n2
18  r6 = n1 ** n2
19
```

Operador aritméticos

```
19
20  n1 = 10
21  n2 = 2
22
23  r1 = n1 == n2
24  r2 = n1 != n2
25  r3 = n1 > n2
26  r4 = n1 < n2
27  r5 = n1 => n2
28  r6 = n1 =< n2
29
```

Operador lógicos

```
20
21  n1 += 1
22  n1 = n1 + 1
23
```

# Listas

```
49
50 lista = []
51 # []
52
53 lista.append(1)
54 lista.append(2)
55
56 # [1, 2]
57
58 lista2 = [3, 4]
59
60 lista_final = lista + lista2
61 # [1, 2, 3, 4]
62
```

```
63
64 lista = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
65
66 lista[0]
67 # 0
68
69 lista[5]
70 # 5
71
72 lista[-1]
73 # 9
74
75 lista[-5]
76 # 5
77
78 lista[0:3]
79 # [0, 1, 2]
80
81 lista[0:6]
82 # [0, 1, 2, 3, 4, 5]
83
84 lista[0:6:2]
85 # [0, 2, 4]
86
```

# Strings

```
30
31 "cadena de caracteres"
32 'cadena de caracteres'
33
```

```
56
57 "hola " + 4
58 # TypeError: can only concatenate str (not "int") to str
59
60 "hola " + str(4)
61 # 'hola 4'
62
```

```
57
58 "Hola".upper()
59 # "HOLA"
60 "Hola".lower()
61 # "hola"
62
63 "Hola".split('l')
64 # ['Ho', 'a']
65
```

```
34
35 l1 = "hola "
36 l2 = "mundo!"
37
38 r1 = l1 + l2
39 # "hola mundo!"
40
41 r2 = l1 == l2
42 r3 = l1 != l2
43
44
45 l1[0]
46 # "h"
47
48 l1[-1]
49 # " "
50
```

# Diccionarios

```
99
100 dic = {}
101
102 dic['a'] = 0
103 dic['b'] = 1
104
105 # {'a': 0, 'b': 1}
106
107 dic + {'c': 3}
108 # TypeError: unsupported operand type(s) for +: 'dict' and 'dict'
109
110 dic['a']
111 # 0
112
113 dic['c']
114 # KeyError: 'c'
115
```

# (Ficheros)

```
255
256  f = open('texto.txt', 'r')
257
258  contenido = f.read()
259
260  f.close()
261
```



# Bucles

- Iteradores y bucles sencillos
- Enumerar
- Diccionarios

# Iteradores y bucles sencillos

```
156  
157     for(int i = 0; i < len(lista); i++) {  
158  
159         print(lista[i]);  
160     }  
161
```

```
124     for elemento in lista:  
125  
126         ...print(elemento)  
127
```

# Iteradores y bucles sencillos

```
210
211 ▼ while True:
212     |
213     |     print("infinito")
214
```

# Iteradores y bucles sencillos

```
140
141  for elemento in range(0, 10):
142
143      print(elemento)
144
145  # 0
146  # 1
147  # 2
148  # 3
149  # 4
150  # 5
151  # 6
152  # 7
153  # 8
154  # 9
155
```

```
121
122  lista = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
123
124  for elemento in lista:
125
126      print(elemento)
127
128
129  # 0
130  # 1
131  # 2
132  # 3
133  # 4
134  # 5
135  # 6
136  # 7
137  # 8
138  # 9
139
```

# Enumerar

```
163
164  lista = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
165
166  for indice, elemento in enumerate(lista):
167
168      print(indice, ":", elemento)
169
170
171  # 0 : 0
172  # 1 : 1
173  # 2 : 2
174  # 3 : 3
175  # 4 : 4
176  # 5 : 5
177  # 6 : 6
178  # 7 : 7
179  # 8 : 8
180  # 9 : 9
181
```

# Diccionarios

```
182
183
184 dic = {'a':0, 'b':1, 'c':2, 'd':3, 'e':4, 'f':5}
185
186 # dic.values() -> 1, 2, 3, 4, 5, ...
187 # dic.keys()   -> 'a', 'b', 'c', ... (POR DEFECTO)
188 # dic.items()  -> {'a':0, 'b':1, ...}
189
190
191 for clave, valor in dic.items():
192     print(clave, ":", valor)
193
194
195 # a : 0
196 # b : 1
197 # c : 2
198 # d : 3
199 # e : 4
200 # f : 5
201
```

# Condiciones

If / else

# If / else

```
216
217 valor = 10
218
219 if valor == 1:
220     print("Vale 1!")
221
222 elif valor != 10:
223     print("No es el valor 10")
224
225 elif valor % 2 == 0:
226     print("Es múltiplo de 2")
227
228 else:
229     print("No cumple las anteriores condiciones.")
230
```

```
234
235 if valor == 10 or valor == 20:
236     print("Vale 10 o 20.")
237
238 elif valor > 10 and valor < 20:
239     print("Está entre 10 y 20.")
240
```



# Funciones

# Funciones

```
235
236 def funcion(atr1, atr2, atr3, atr=True):
237     '''
238     Esta es la documentacion de la funcion.
239     '''
240
241     return ret1, ret2
242
```

```
244
245 ret1, _ = funcion(atr1, atr2, atr3)
246
247 ret1, ret2 = funcion(atr1, atr2, atr3, atr=10)
248
```

## Parte 1

- Introducción al lenguaje.
- Variables y tipos de datos.
- Bucles.
- Condiciones.
- Funciones.

## Parte 2

- Orientación a objetos.
- Creación de entorno.
- Numpy
- Matplotlib
- Sklearn
- Keras / Tensorflow
- Matplotlib
- Pandas



# Parte 2

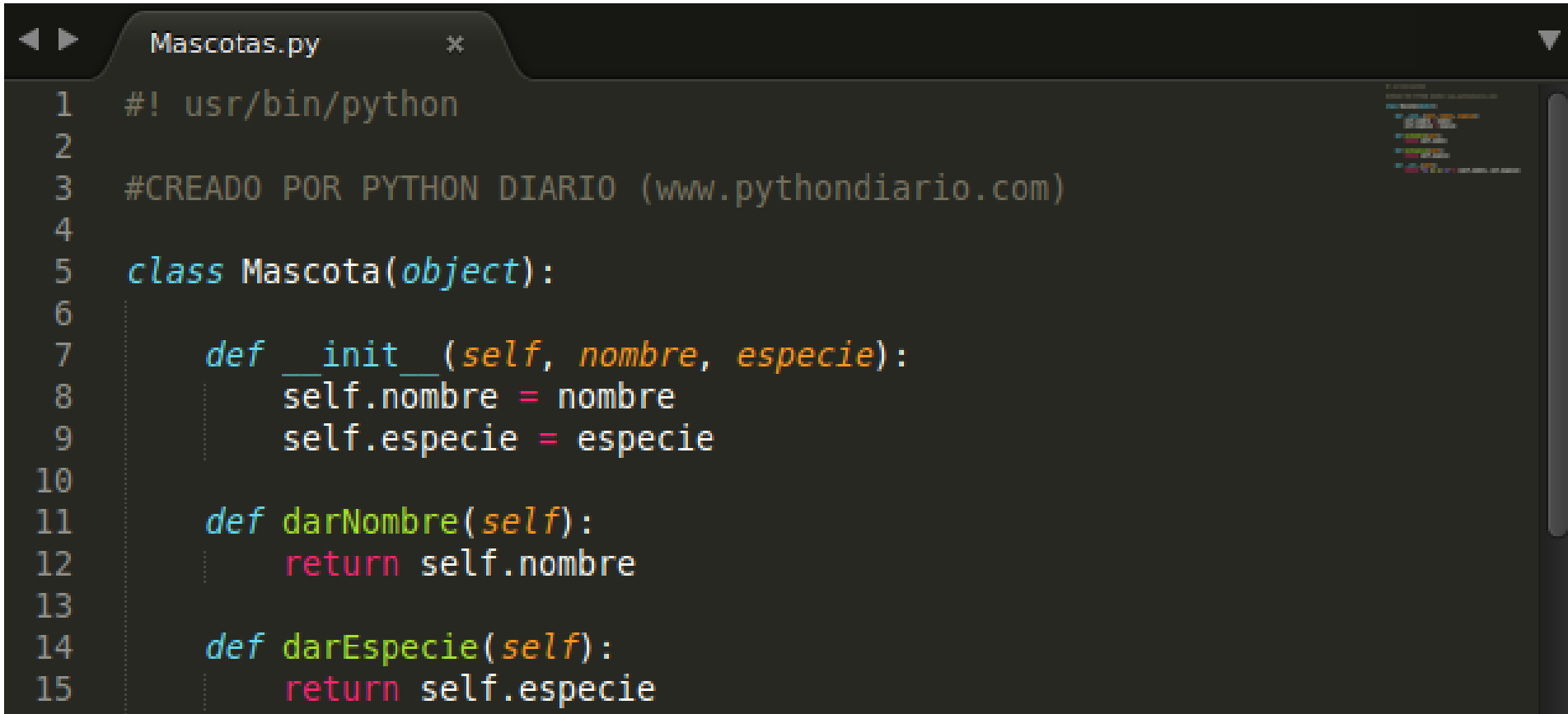
# POO (Programación Orientada a Objetos)

- Intento de imitación de la realidad
- Clases con ejemplos que tienen características y hacen cosas.
- Mejora de la limpieza y el poder de editar el código

# Classes

```
28
29 class Gato(Mascota):
30
31     def __init__(self, nombre, odia_perros):
32         Mascota.__init__(self, nombre, "Gato")
33         self.odia_perros = odia_perros
34
35     def odiaPerros(self):
36         return self.odia_perros
37
```

# Métodos especiales



```
1  #! usr/bin/python
2
3  #CREADO POR PYTHON DIARIO (www.pythondiario.com)
4
5  class Mascota(object):
6
7      def __init__(self, nombre, especie):
8          self.nombre = nombre
9          self.especie = especie
10
11      def darNombre(self):
12          return self.nombre
13
14      def darEspecie(self):
15          return self.especie
```

-¿Y si queremos enseñar el nombre y especie?

# Métodos abstractos

```
1  import abc
2
3
4  class A(abc.ABC):
5      @abc.abstractmethod
6      def method_1(self):
7          """
8          This abstract method should return a list
9          """
10         pass
11
12     def method_2(self):
13         list_output_method_1 = self.method_1()
14         for x in list_output_method_1:
15             pass
16 Expected 'collections.Iterable', got 'None' instead more... (%F1)
```



# Librerías / Paquetes

[Help](#)[Donate](#)[Log in](#)[Register](#)

**Find, install and publish Python packages  
with the Python Package Index**



Or [browse projects](#)



# Librerías / Paquetes

pip

```
pip install <paquete>
```

# Instalación

```
sudo apt-get virtualenv
```

**×** Máquina virtual

```
virtualenv env --python=python3
```

```
source env/bin/activate
```

```
deactivate
```

S0

Entorno virtual



# Numpy

$$A = \begin{matrix} & \begin{matrix} \text{columnas} & \downarrow 1 & \downarrow 2 & \downarrow 3 & & \downarrow n \end{matrix} & \\ \begin{matrix} \left( \begin{array}{ccccc} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{array} \right) \end{matrix} & \begin{matrix} \leftarrow 1 \\ \leftarrow 2 \\ \leftarrow 3 \\ \\ \leftarrow m \end{matrix} \end{matrix} \begin{matrix} \text{Filas} \\ \\ \\ \\ \end{matrix} \end{matrix} \longrightarrow$$



Imagen 1



Imagen 2



Máscara



Fusión sin piramides

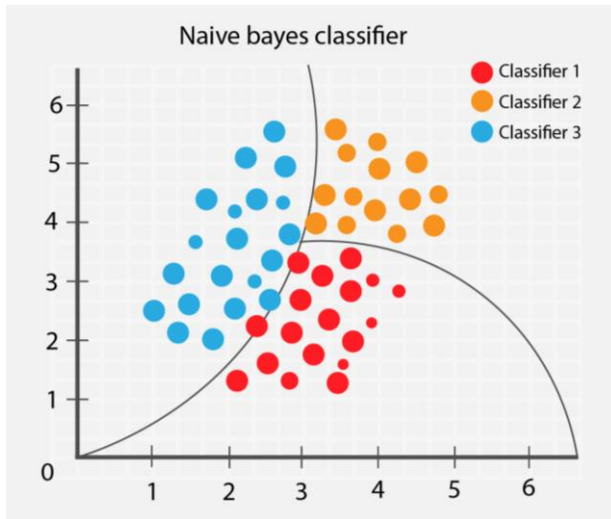


Fusión con piramides (4 niveles)

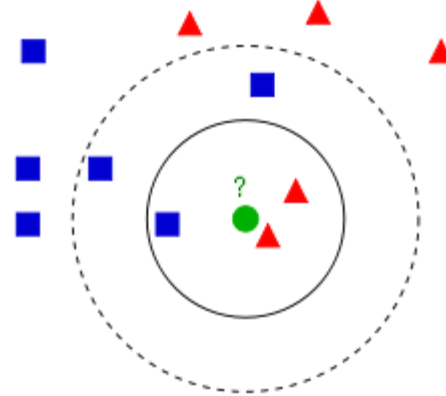


# Sklearn

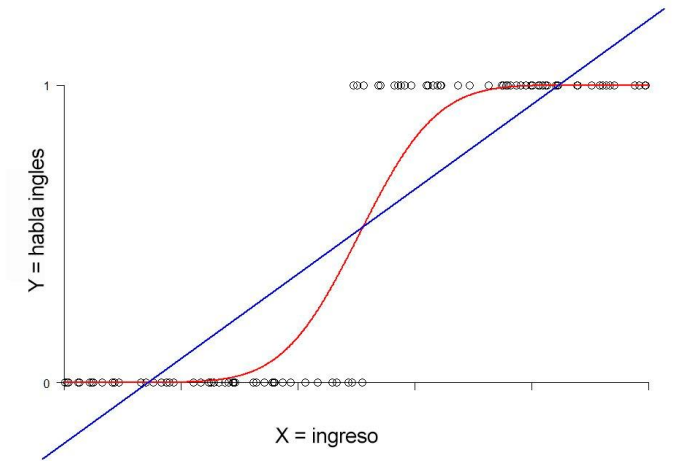
$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$



Clasificador Naïve Bayes

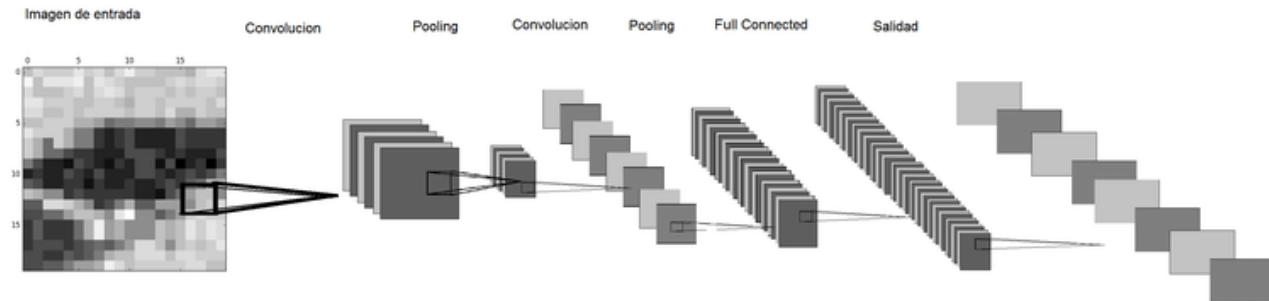
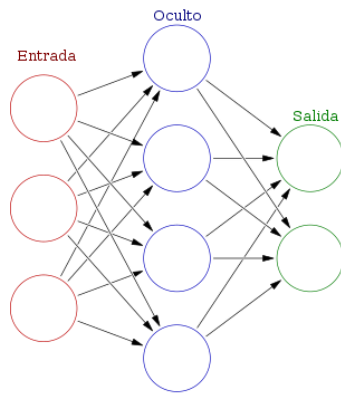


Clasificador K vecinos próximos

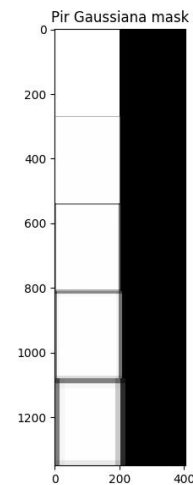
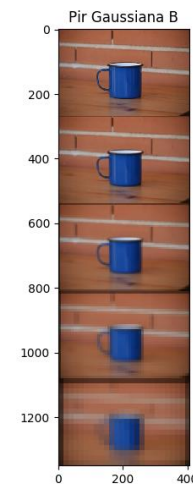
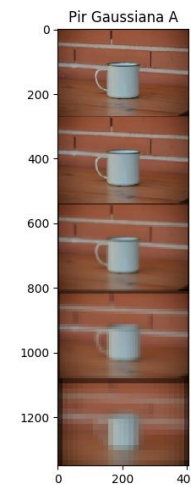
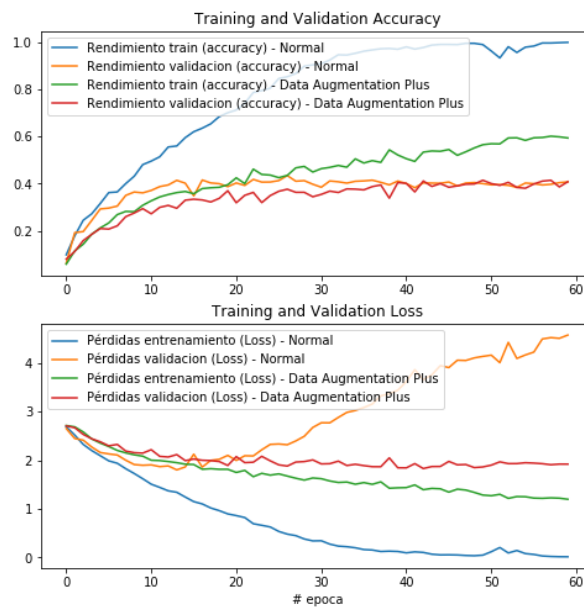
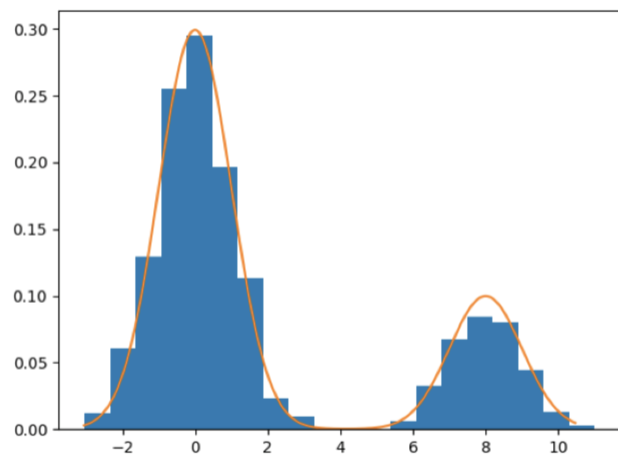


Clasificador de regresión logística

# Keras y TensorFlow



# Matplotlib





# Pandas

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

