Aprendizaje Automático Profundo

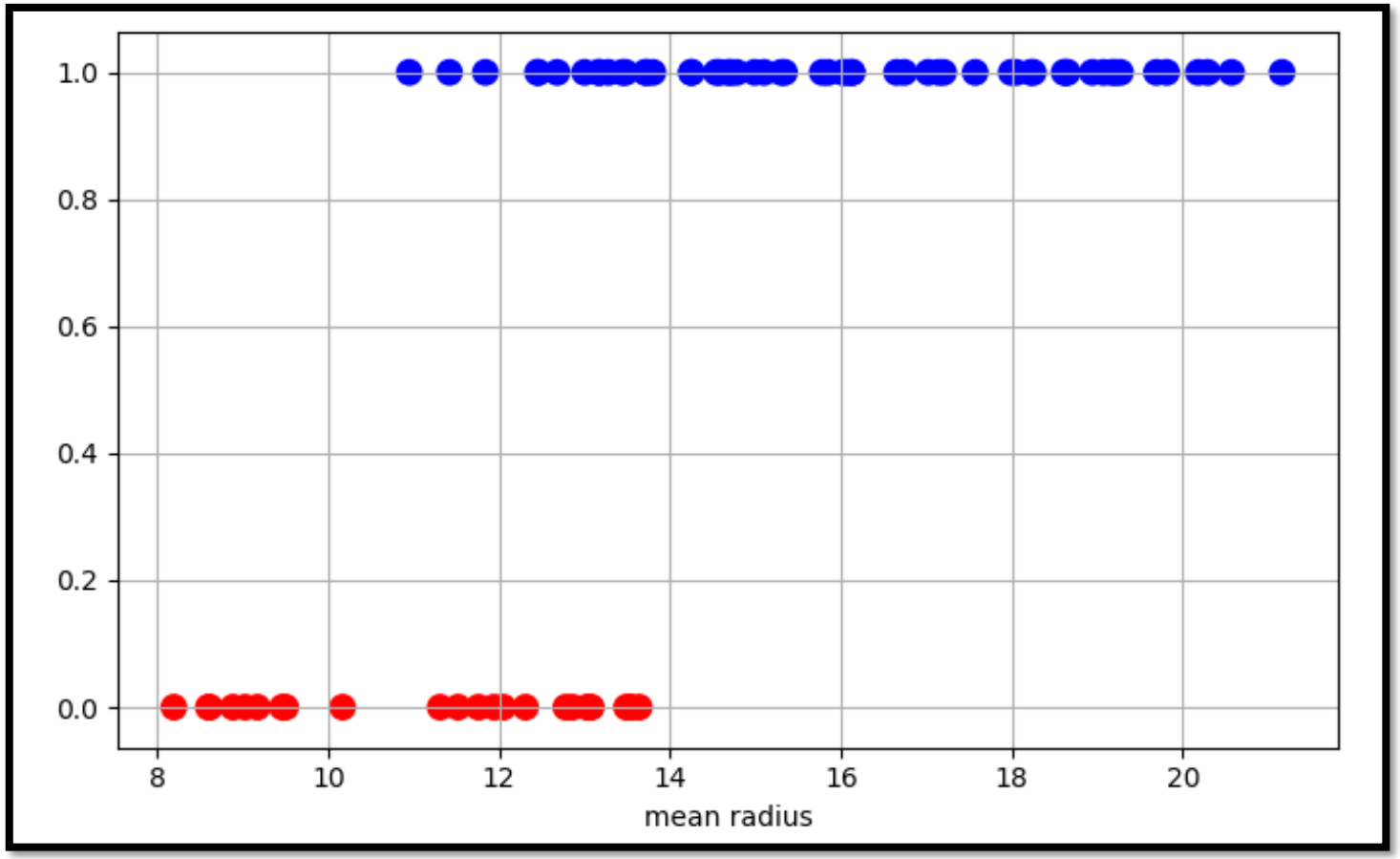
Clase 3 - 2019

Profs: Franco Ronchetti - Facundo Quiroga

REGRESIÓN LOGÍSTICA

Clasificación Binaria

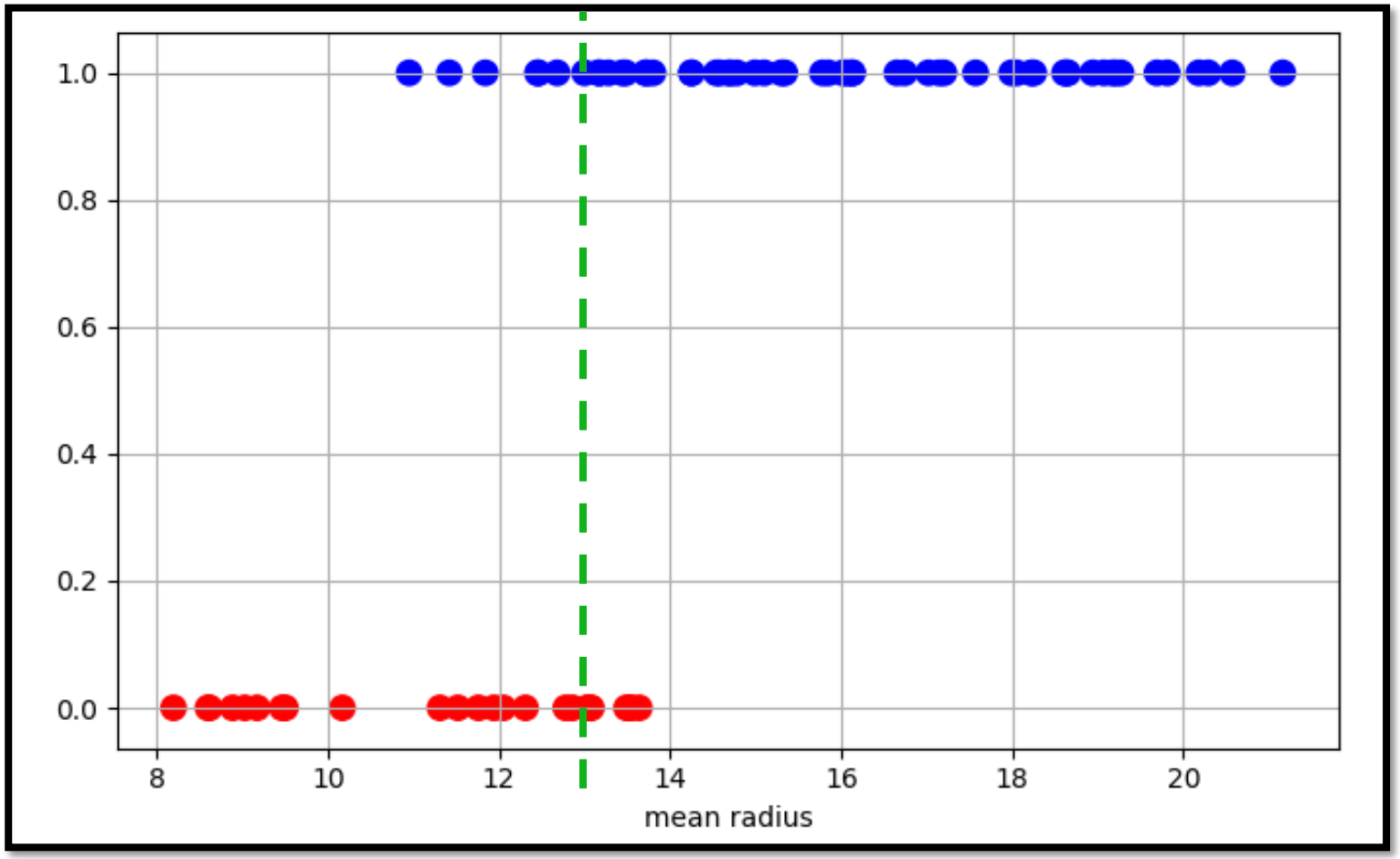
Supongamos que tenemos información sobre el tamaño varios tumores y si finalmente fueron benignos (0), o malignos (1).



Queremos crear un modelo que sea capaz de discriminar entre ambas clases: **clasificación binaria**.

Clasificación Binaria

¿Cómo podemos separar las clases?

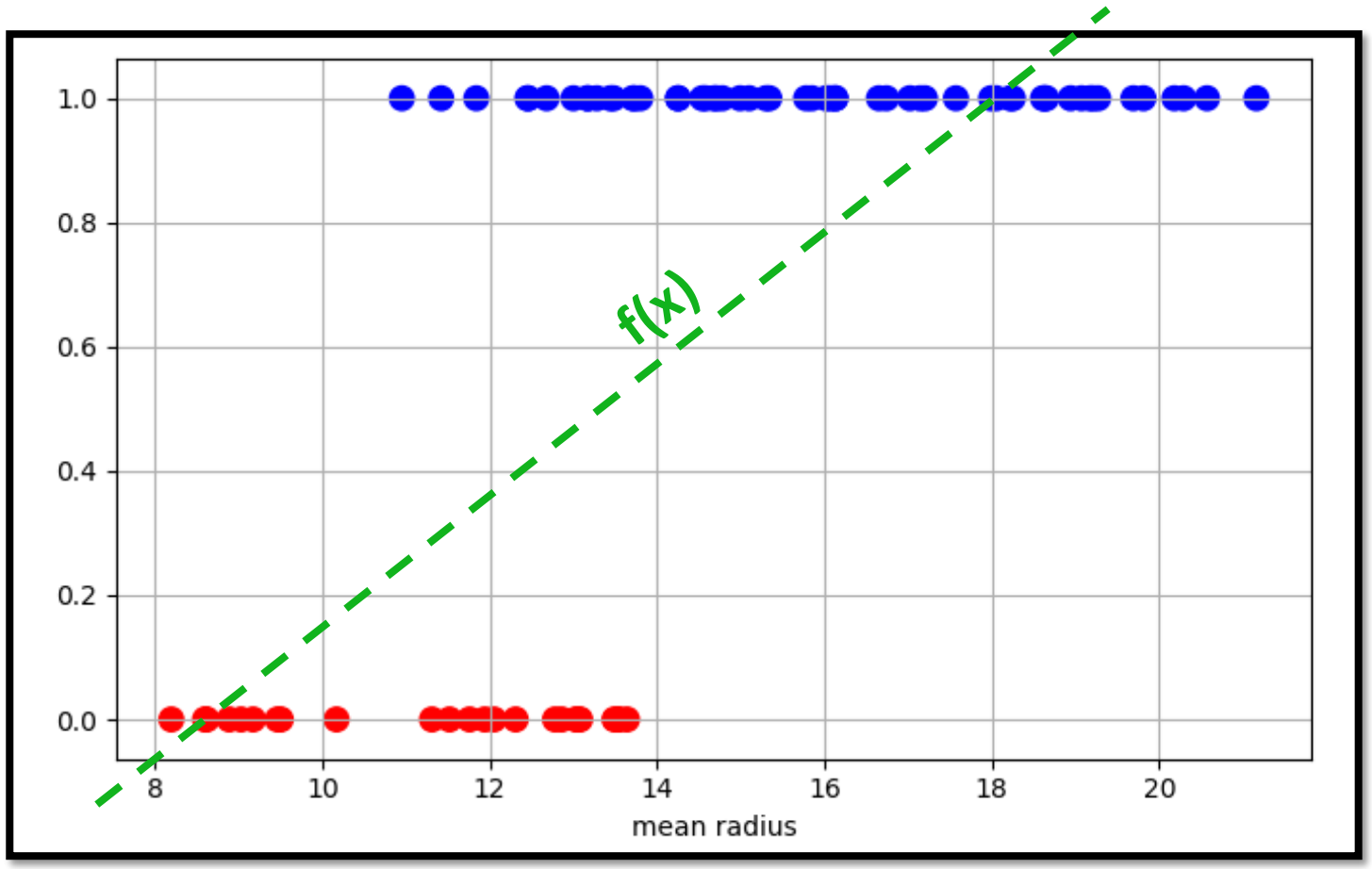


Ej: Clasificador *ad hoc*

$$f(x) = \begin{cases} 1 & si \;\; x > 13 \\ 0 & si \;\; x \leq 13 \end{cases}$$

Clasificación Binaria

Podríamos utilizar Regresión Lineal para ajustar una recta y luego decidir partiendo a la mitad el espacio.

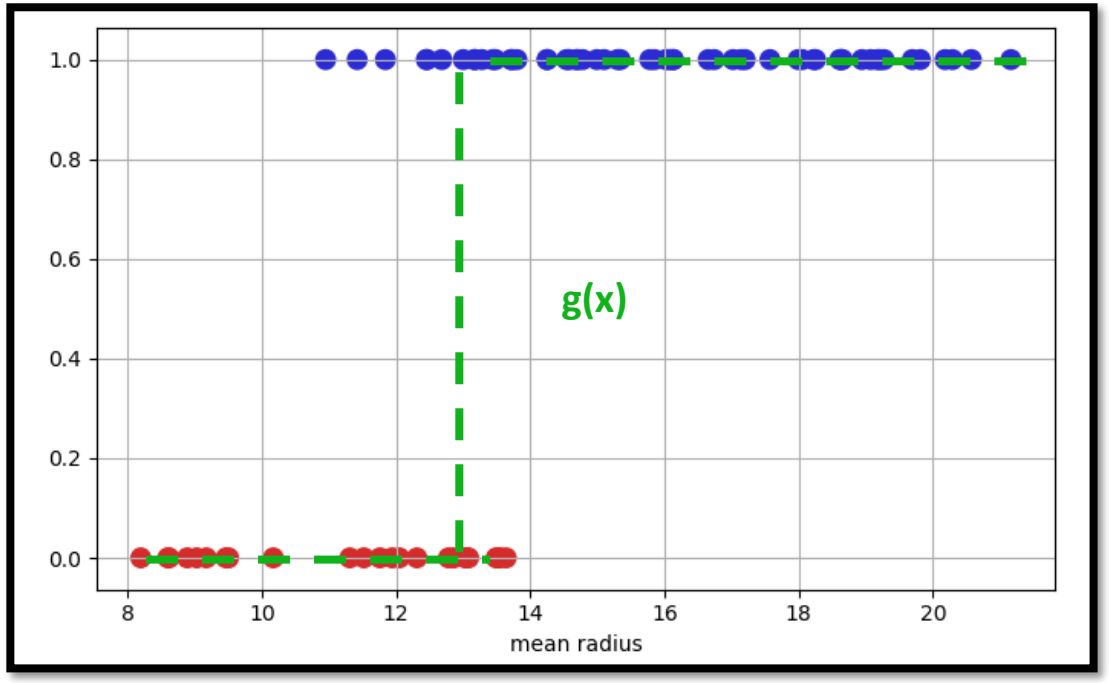


$$g(x) = \begin{cases} 1 & si\ f(x) > 0{,}5 \\ 0 & si\ f(x) \leq 0{,}5 \end{cases}$$

Clasificación Binaria

Tenemos dos problemas:

1. Cualquier cambio en la distribución de los datos corre mucho la línea de corte.
2. La función partida no es diferenciable.



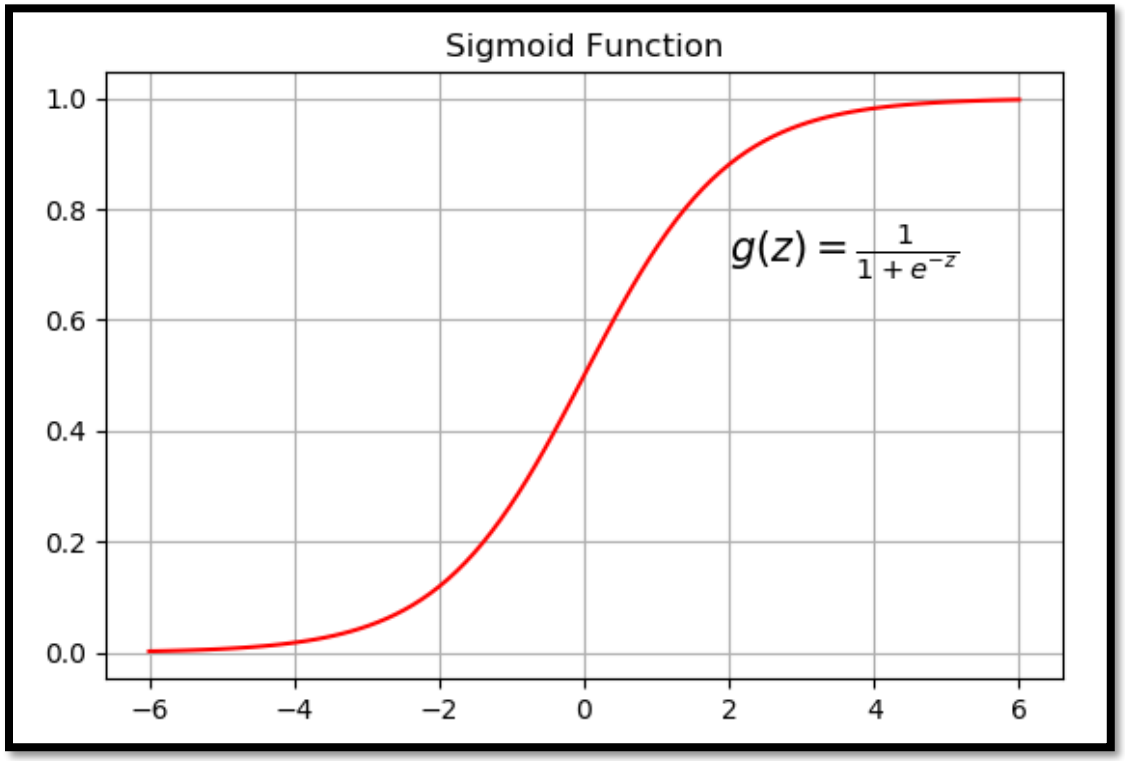
$$g(x) = \begin{cases} 1 & si\ f(x) > 0{,}5 \\ 0 & si\ f(x) \leq 0{,}5 \end{cases}$$

Regresión Logística

Un modo de solucionarlo es utilizando una función Sigmoidal, que tiene un rango de salida entre 0 y 1. Además, la función de Error es diferenciable.

$$g(z) = \frac{1}{1+e^{-z}}$$

$$g(f(x)) = \frac{1}{1+e^{-wx-b}}$$



g(z) tiene dos asíntotas horizontales:

- Tiende a 1 cuando x tiende a $+\infty$
- Tiende a 0 cuando x tiende a $-\infty$

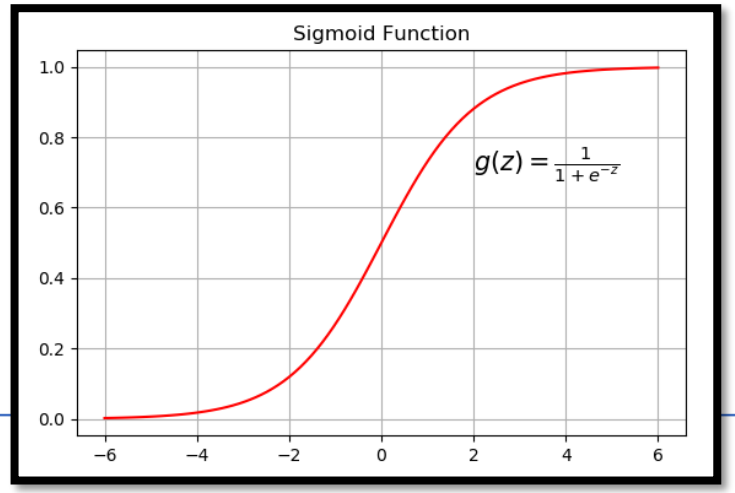
Regresión Logística

Lo que me está diciendo esta función hipótesis es la “probabilidad de que y=1”.

En el ejemplo de los tumores. Si para un ejemplo particular (supongamos x=14), la hipótesis me retorna 0.7, significa que hay un 70% de probabilidad de que sea un tumor maligno.

Formalmente: $g(x) = P(y=1 \mid x; w)$

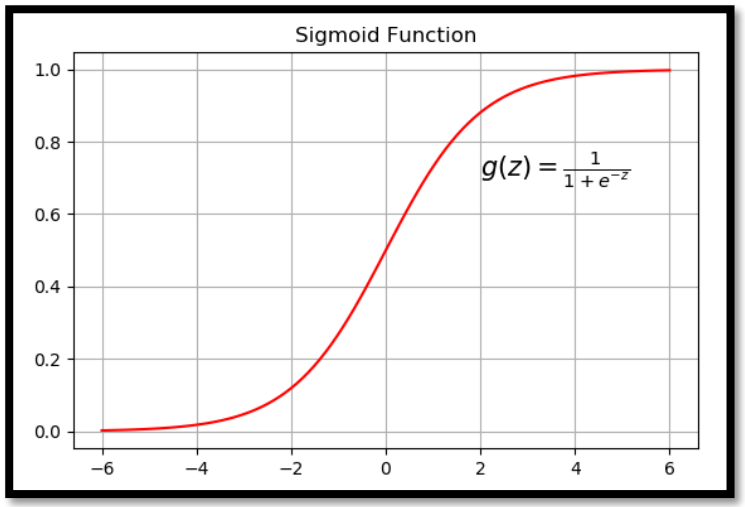
Por otro lado, $P(y=0 \mid x; w) = 1 - P(y=1 \mid x; w)$



Regresión Logística

Luego, podemos decir que

$$y = \begin{cases} 1 & si\ g(x) > 0{,}5 \\ 0 & si\ g(x) \leq 0{,}5 \end{cases} \quad \begin{matrix} \rightarrow \\ \rightarrow \end{matrix} \quad \begin{matrix} wx + b > 0 \\ wx + b \leq 0 \end{matrix}$$



Nos queda pendiente revisar la función de Error

Frontera de decisión

Supongamos, para nuestro problema de tumores, que

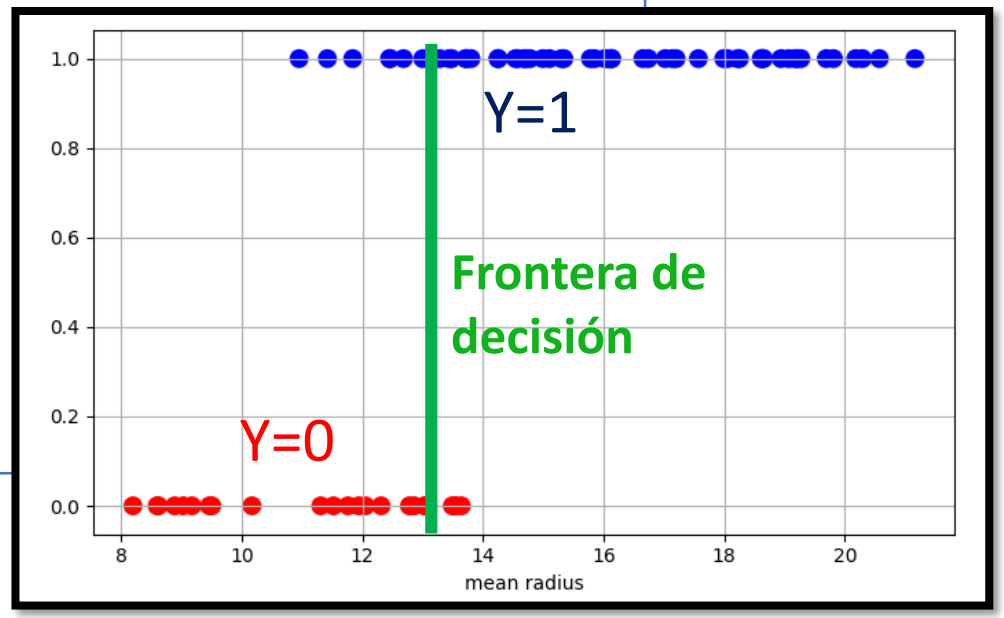
$\mathbf{w}= 0{,}3$ y $\mathbf{b}= -4$,

Entonces, $\mathbf{y = 1}$ si $0{,}3x - 4 > 0$

$$x > \frac{4}{0{,}3}$$

$\mathbf{y = 1}$ si $\mathbf{x > 13{,}33}$

Frontera de Decisión x=13,33



Frontera de decisión

En el caso de tener dos dimensiones (o más), la idea es igual:

Supongamos los siguientes puntos pertenecientes a dos clases

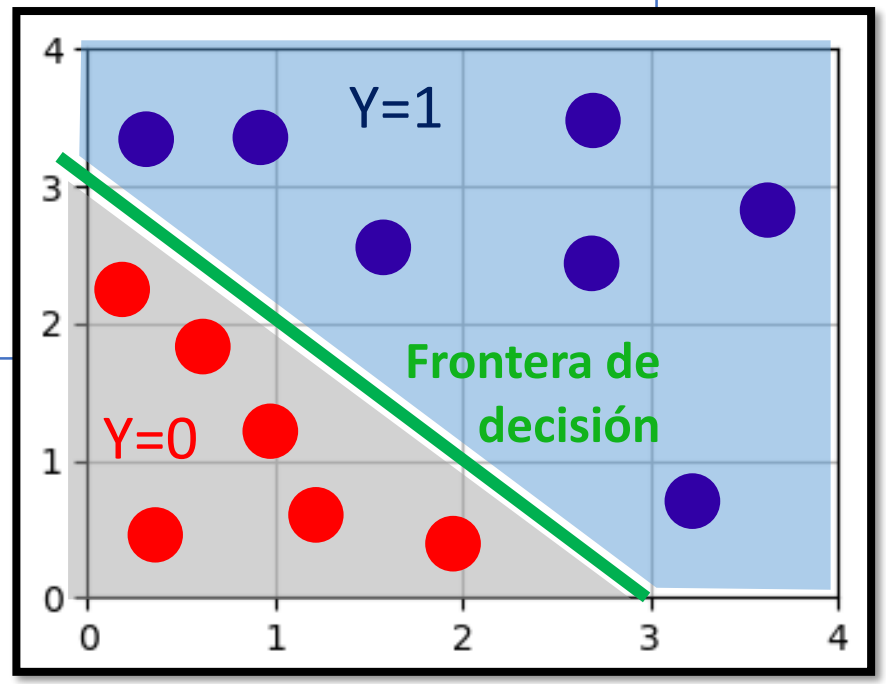
Mi hipótesis sería Hip(x1, x2)= g(w1x1 + w2x2 + b)

Supongamos

$\mathbf{w1} = 1, \mathbf{w2} = 1$ y $\mathbf{b} = -3$,

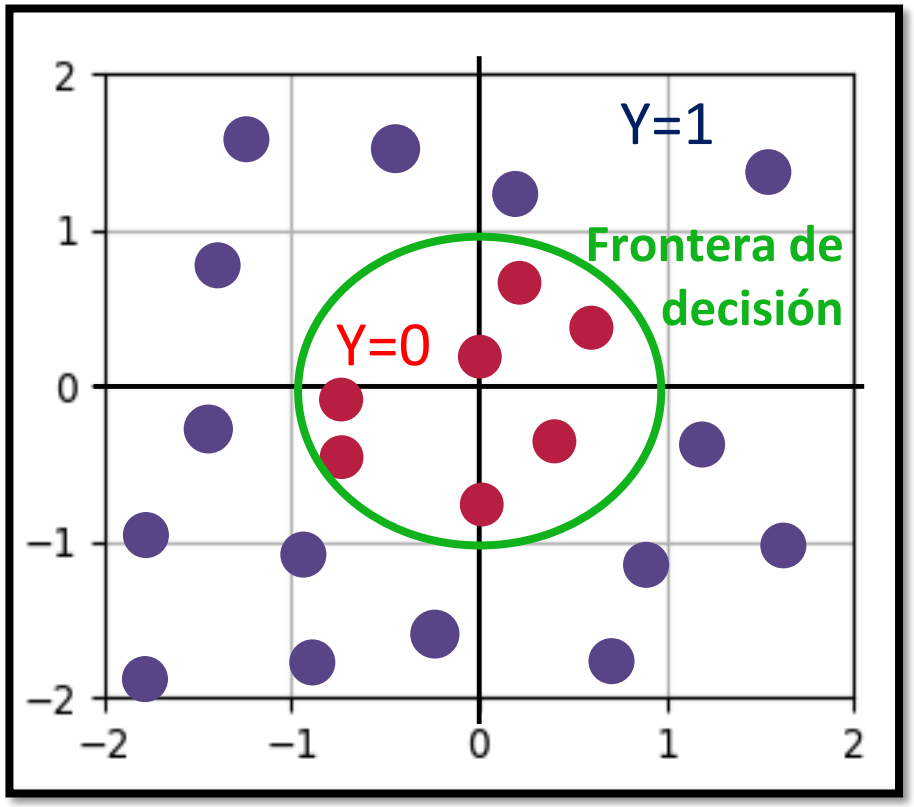
Entonces, $\mathbf{y = 1}$ **si** $x1 + x2 - 3 > 0$

$$\mathbf{x1 + x2 > 3}$$



Frontera de decisión no lineal

¿Cómo podemos resolver el caso de tener una distribución de las clases como la siguiente?



Podríamos utilizar un polinomio para ajustar mi hipótesis.

$$\mathbf{hip}(\mathbf{x}) = \mathbf{g}(\boldsymbol{w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + b})$$

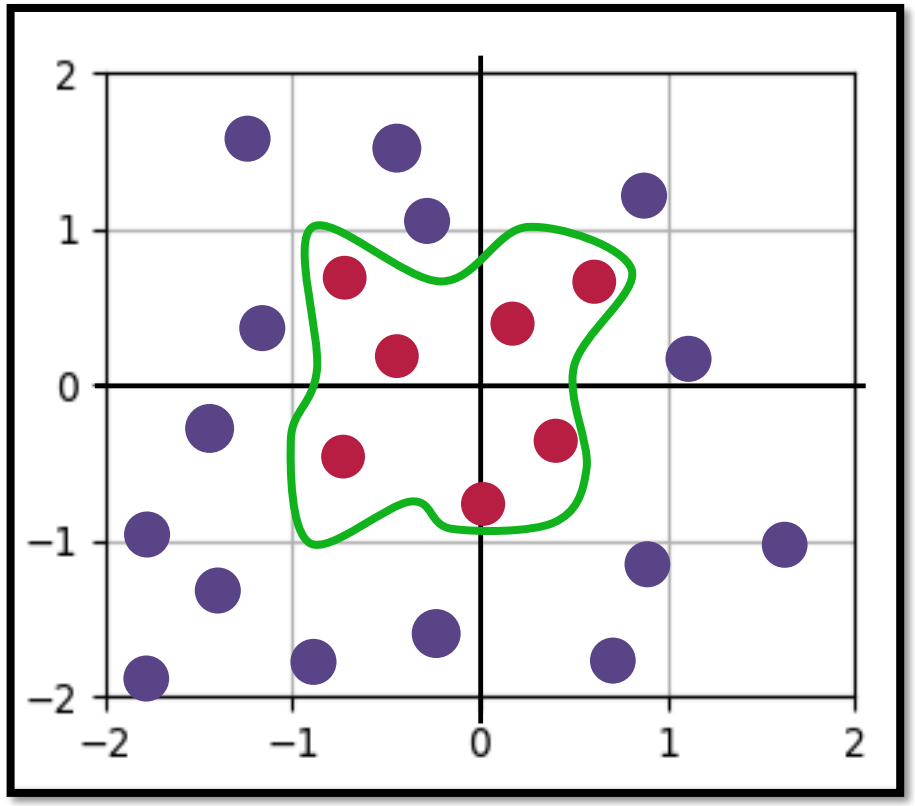
Supongamos $\boldsymbol{w_1 = 0,\ w_2 = 0,\ w_3 = 1, w_4 = 1,}$

$$\boldsymbol{b = -1,}$$

Entonces, $\mathbf{y = 1}$ **si** $x_1^2 + x_2^2 - 1 > 0$

$$x_1^2 + x_2^2 > 1$$

Frontera de decisión



Para ir pensando: ¿Qué pasaría si utilizamos un polinomio de mayor grado?

Podría generarse un Sobreajuste (*Overfitting*).

Regresión Logística. Error.

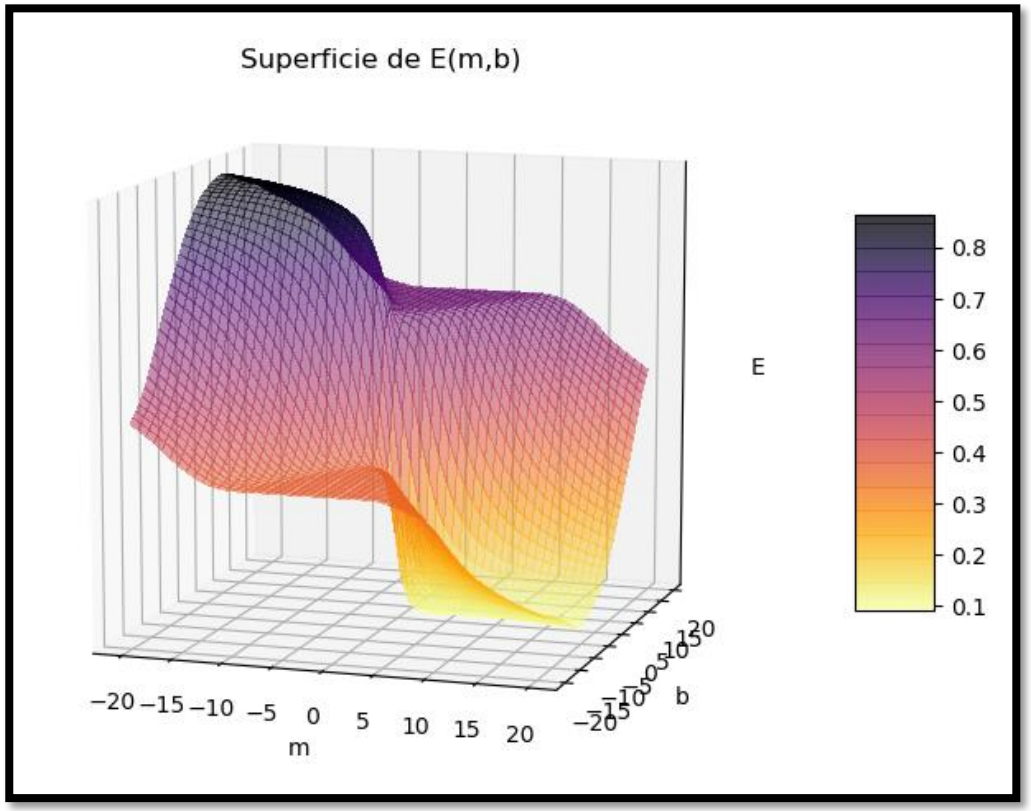
$$E = \frac{1}{n}\sum_{i}^{n}(g(mx_i + b) - y_i)^2$$

¿Cómo obtenemos los parámetros del modelo?

Podemos utilizar el Error cuadrático medio para entrenar la Regresión Logística. Pero tenemos un problema: la función no es convexa.

En cambio, se utiliza la **entropía cruzada.**

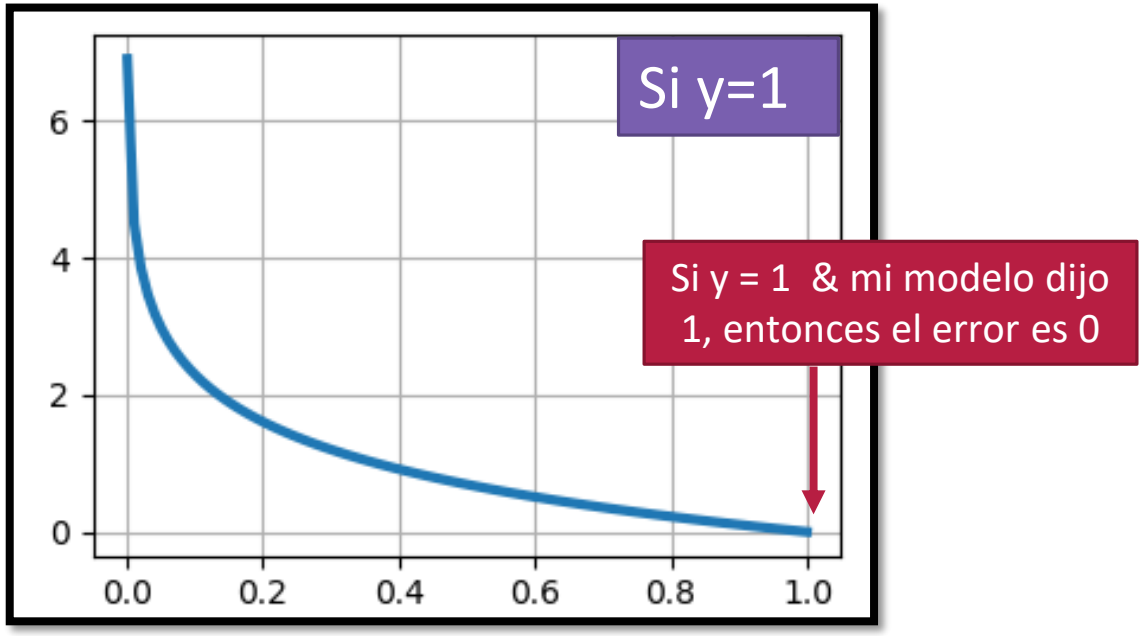
Nos permite medir distancias entre distribuciones de probabilidades.

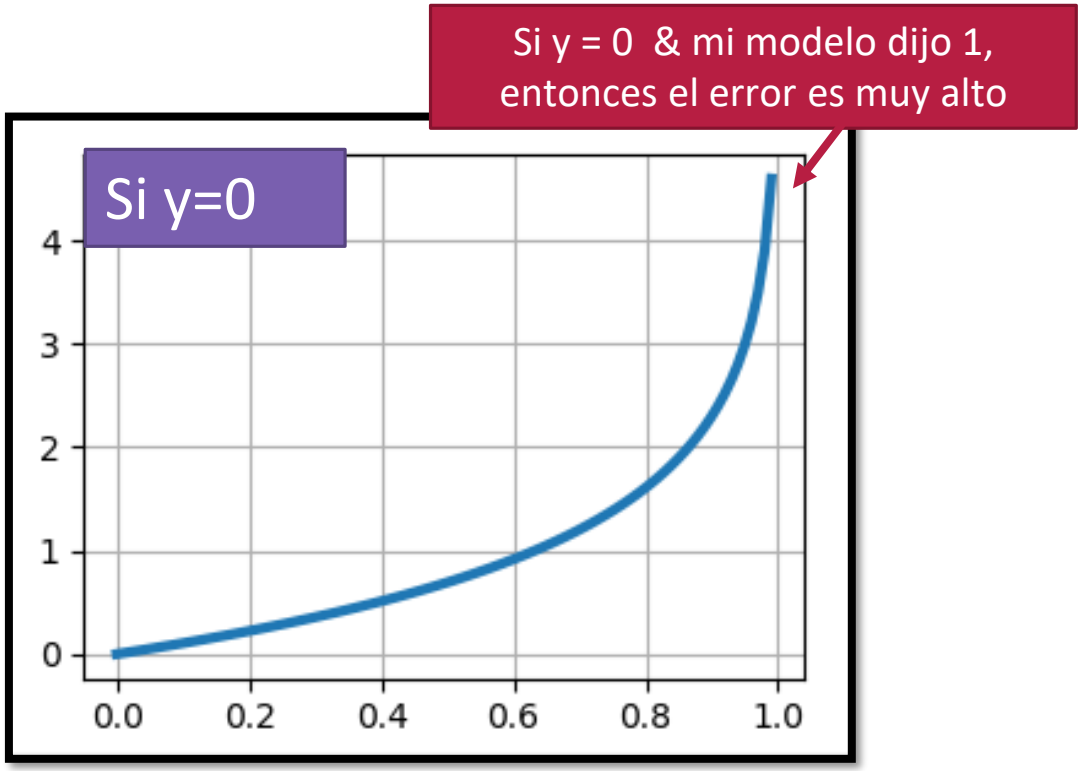


Regresión Logística. Entropía Cruzada.

$$E = \frac{1}{n}\sum_{i}^{n} E_i$$

$$E_i = \begin{cases} -\log(g(x_i)) & si\ y = 1 \\ -\log(1 - g(x_i)) & si\ y = 0 \end{cases}$$





Regresión Logística. Entropía Cruzada.

$$E = \frac{1}{n}\sum_{i}^{n} E_i$$

$$E_i = \begin{cases} -\log\big(g(x_i)\big) & si\ y = 1 \\ -\log\big(1 - g(x_i)\big) & si\ y = 0 \end{cases}$$

Podemos escribir la función partida en una sola línea:

$$\boldsymbol{E_i} = -y\ \log\big(g(x_i)\big) - \Big((1-y)\log\big(1 - g(x_i)\big)\Big)$$

Los coeficientes "y", "$(1-y)$", actúan como "if-else"

Regresión Logística. Error.

Para realizar descenso del gradiente, sólo es necesario computar las derivadas del error:

$$\frac{\delta Ei}{\delta b} = \mathrm{f(x)} - \mathrm{y}$$

$$\frac{\delta Ei}{\delta m} = (\mathrm{f(x)} - \mathrm{y})\mathrm{x}$$

Ejemplo 1D

DATOS

mean radius	target
13.73	1
14.54	1
14.68	1
16.13	1
19.81	1
13.54	0
13.08	0
9.504	0
15.34	1
21.16	1
16.65	1
17.14	1

```
w
array([[0.51737148]])

b
array([-7.81318005])
```

```
Loss (error): 0.33
```

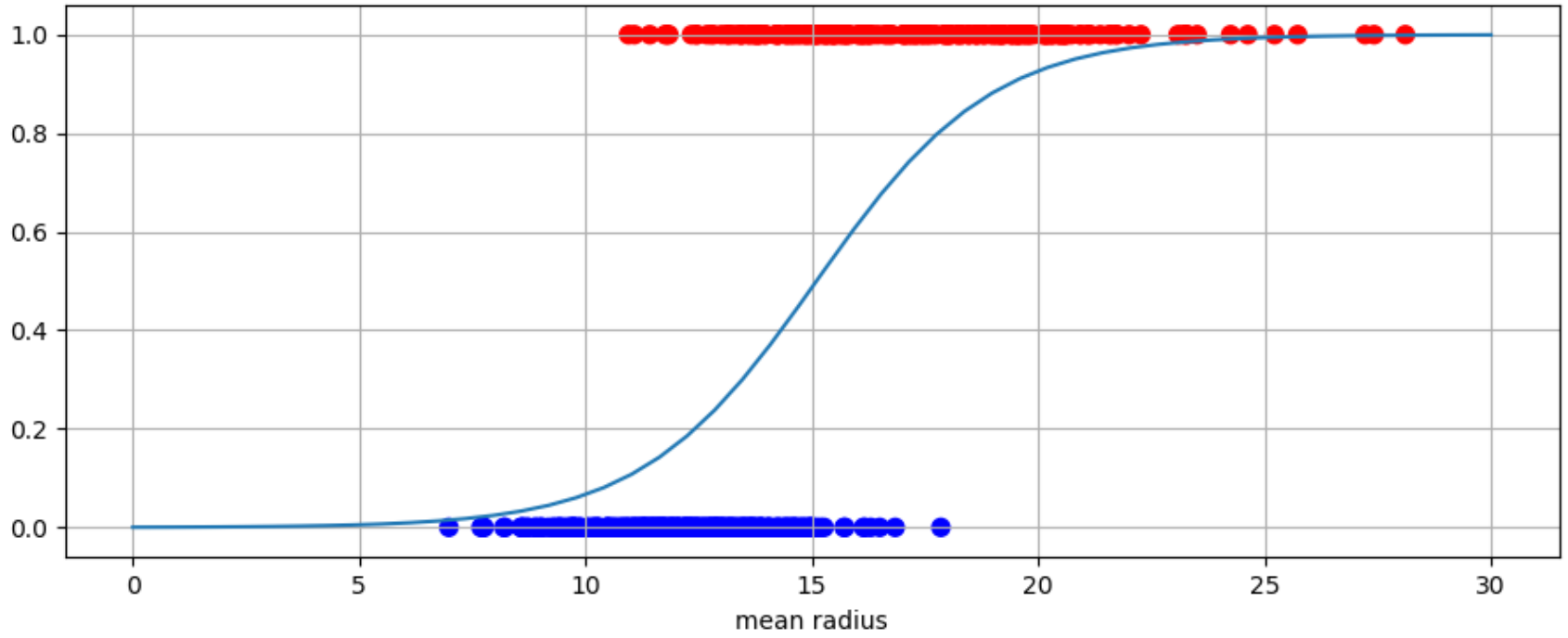
$y = 1 \quad si \quad si\ g(x) > 0{,}5 \quad \rightarrow \quad \boldsymbol{wx + b > 0}$

$\boldsymbol{0{,}52x - 7{,}8 > 0}$

$\boldsymbol{0{,}52x > 7{,}8/0{,}52}$

$\boldsymbol{x > 15}$

$g(x) = \frac{1}{1+e^{-wx-b}}$



Regresión Logística. Resumen.

- Mi modelo no es más que $\boldsymbol{w}$ y $\boldsymbol{b}$
- Luego, aplico G(f(x)) que me da una probabilidad de pertenencia a la clase 1 (función sigmoidea).
- Para clasificar, simplemente escalonamos esa función cortando en un umbral.
- Los parámetros se “aprendieron” minimizando el error definido por la entropía cruzada en base a los datos.

Ejemplo 2D

Ahora tenemos 2 variables para cada registro

$y = 1 \quad si \quad si\ g(x) > 0{,}5 \quad \rightarrow \quad \boldsymbol{w1x1 + w2x2 + b > 0}$

$\boldsymbol{0{,}47x1 + 0{,}05x2 - 8 > 0}$

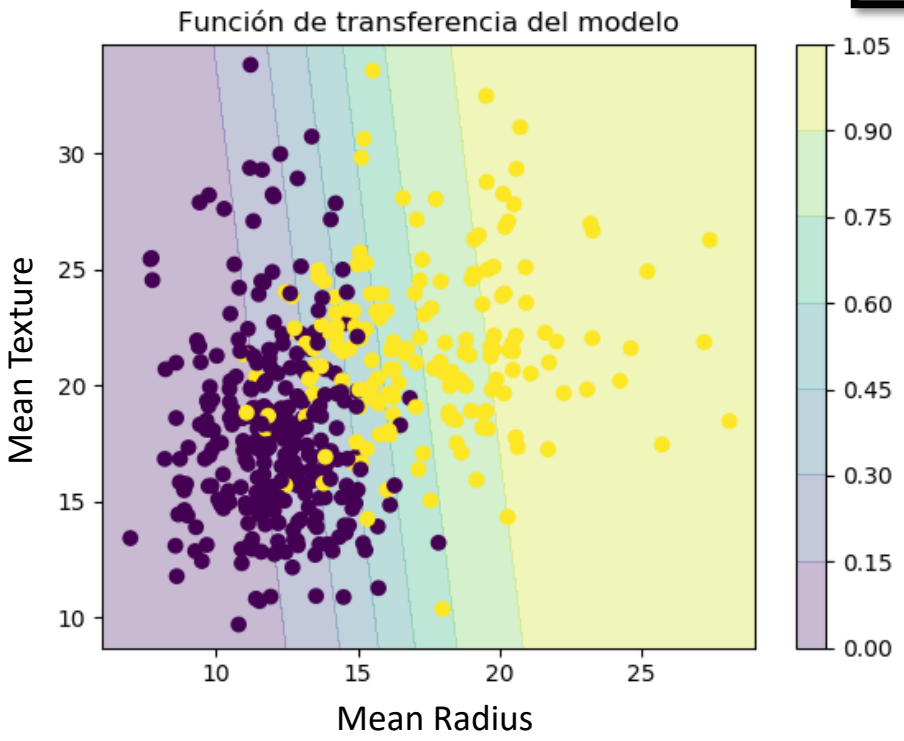
$\boldsymbol{0{,}47x1 + 0{,}05x2 > 8}$

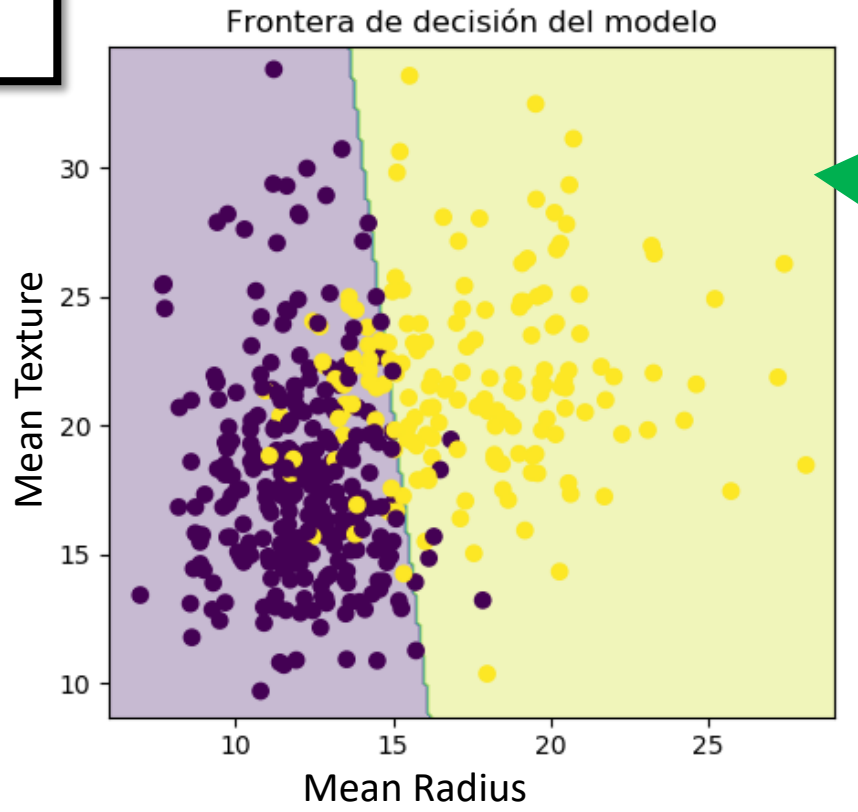
$\boldsymbol{x2 = 160 - 9{,}4x1}$

```
w
array([[0.47033752, 0.04635503]])

b
array([-8.0051825])
```

Loss (error): 0.32





Ejemplo ND

breast_cancer.csv

DataFrame	(569, 31)

```
df.columns

'mean radius', 'mean texture', 'mean perimeter', 'mean area',
'mean smoothness', 'mean compactness', 'mean concavity',
'mean concave points', 'mean symmetry', 'mean fractal dimension',
'radius error', 'texture error', 'perimeter error', 'area error',
'smoothness error', 'compactness error', 'concavity error',
'concave points error', 'symmetry error', 'fractal dimension error',
'worst radius', 'worst texture', 'worst perimeter', 'worst area',
'worst smoothness', 'worst compactness', 'worst concavity',
'worst concave points', 'worst symmetry', 'worst fractal dimension',
'target'],
```

MODELO ENTRENADO

```
w

[-1.26295960e+00, -2.25489139e-01, -2.43393550e-01,
  1.62175280e-02,  4.99734817e-02,  2.24574570e-01,
  3.23667429e-01,  1.51384120e-01,  7.94138472e-02,
  1.41031274e-02, -2.95021486e-02, -5.21128942e-01,
 -8.58250005e-02,  1.07684743e-01,  4.32656526e-03,
  3.19192271e-02,  5.31989759e-02,  2.04630873e-02,
  2.06397377e-02,  1.11784584e-03, -1.15229290e+00,
  3.73401255e-01,  2.85087901e-01,  1.36335625e-02,
  9.17199071e-02,  6.50232988e-01,  8.36191624e-01,
  2.98052807e-01,  2.56801257e-01,  6.01426341e-02]])

b
array([-0.22395408])
```

```
Loss (error): 0.10
```

Keras

Keras es una API de Redes Neuronales Profundas de alto nivel escrita en Python que utiliza como backend TensorFlow, CNTK, o Theano. Está orientado a poder realizar un rápido prototipado de modelos profundos.
https://keras.io/

```
from keras.models import Sequential
from keras.layers import Dense

#create model
model = Sequential()

#add model layers
model.add(Dense(2, input_shape=[N], activation= 'softmax'))
```

Permite crear modelo de varias capas

Capa "lineal" o "fully-connected". En este caso 2 neuronas con salida softmax especifican clasificación binaria.

Cantidad de neuronas de salida

Tamaño de la entrada= Cantidad de variables del dataset

Keras

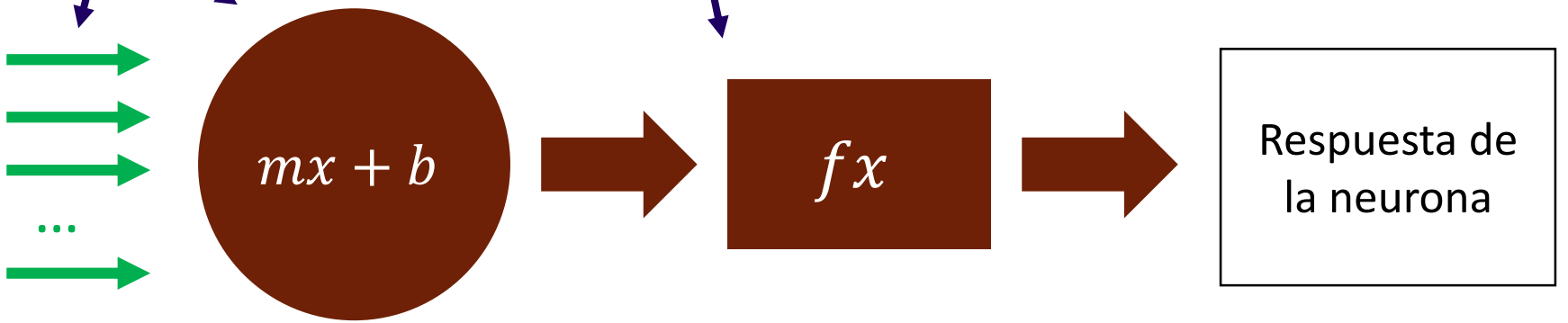
```
from keras.models import Sequential
from keras.layers import Dense

#create model
model = Sequential()

#add model layers
model.add(Dense(1, input_shape=[N], activation= None))
```

Al poner "None" como función de transferencia la neurona responde con la combinación lineal de la entrada.

Una sola neurona de salida (regresión simple)



Keras

```
# compilar el model
model.compile(optimizer= 'sgd', loss= 'mse')


# Imprimir resumen del modelo
model.summary()


# entrena el modelo
model.fit(X, Y, epochs= EPOCAS, batch_size=32)


# Calcula el score del modelo
model.evaluate(X, Y)

# Predecir la salida de un conjunto de datos
model.predict(X)
```

sgd= "descenso de gradiente estocástico".
mse= "Error Cuadrático Medio".

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1)                 5
=================================================================
Total params: 5
Trainable params: 5
Non-trainable params: 0
_________________________________________________________________
```

Epochs **es la cantidad de veces que se verá todo el conjunto de datos.**

Batch_size es la cantidad de datos que se utilizarán en cada iteración.